mathematics

Article

FastText-Based Local Feature Visualization Algorithm for Merged Image-Based Malware Classification Framework for Cyber Security and Cyber Defense

Sejun Jang, Shuyu Li and Yunsick Sung *

Department of Multimedia Engineering, Dongguk University-Seoul, Seoul 04620, Korea; sejun@dongguk.edu (S.J.); lishuyu@dongguk.edu (S.L.)
* Correspondence: sung@dongguk.edu

Received: 18 February 2020; Accepted: 18 March 2020; Published: 24 March 2020

Abstract: The importance of cybersecurity has recently been increasing. A malware coder writes malware into normal executable files. A computer is more likely to be infected by malware when users have easy access to various executables. Malware is considered as the starting point for cyber-attacks; thus, the timely detection, classification and blocking of malware are important. Malware visualization is a method for detecting or classifying malware. A global image is visualized through binaries extracted from malware. The overall structure and behavior of malware are considered when global images are utilized. However, the visualization of obfuscated malware is tough, owing to the difficulties encountered when extracting local features. This paper proposes a merged image-based malware classification framework that includes local feature visualization, global image-based local feature visualization, and global and local image merging methods. This study introduces a fastText-based local feature visualization method: First, local features such as opcodes and API function names are extracted from the malware; second, important local features in each malware family are selected via the term frequency inverse document frequency algorithm; third, the fastText model embeds the selected local features; finally, the embedded local features are visualized through a normalization process. Malware classification based on the proposed method using the Microsoft Malware Classification Challenge dataset was experimentally verified. The accuracy of the proposed method was approximately 99.65%, which is 2.18% higher than that of another contemporary global image-based approach.

Keywords: cyber security; deep learning; malware classification; malware visualization

1. Introduction

Technologies in various fields, such as autonomous control [1], music [2] and multimedia content, are rapidly advancing [3–5]. Through these advancements, information access has progressively become easy, which also exposes users to cyber threats. A malware is any malicious software designed to harm computers or computer networks. Accessing a file that contains malware poses a direct threat to personal information; therefore, malware files are blocked before execution. Every type of malware acts differently, depending on the family it belongs to, and thus, countermeasures for them are also different; this necessitates the classification of malware into different families.

For detecting malware, there are majorly two types of methods: signature-based and heuristic-based; and the latter addresses the shortcomings of the former [6,7]. Heuristic-based malware detection involves malware scanning to detect features suspected of malicious behavior. Towards this end, many dynamic and static analysis methods have been developed [8–11]. A dynamic analysis method detects malicious behavior by executing the malware itself in an isolated virtual environment [12], whereas a static analysis method detects malicious behavior by identifying the overall structure without executing the malware [13,14].

Previously, a method for visualizing malware has been proposed through static analysis [15,16]. A global image can detect malware mutants as the overall structure is maintained, whereas small changes in malware are captured. Global images generated from malware belonging to the same family are similar; making them suitable for classifying malware. However, a global image cannot capture the actual behavior of obfuscated malware. A method for combining the global image and local features was proposed to increase classification accuracy by considering actual malware behavior [17]. The application programming interface (API) and dynamic link library (DLL) information is utilized as a feature from the text section of the bytes file to extract local features. The global image of malware combined with local features can be used for accurate malware classification. However, there is a need for a method to extract any local features of obfuscated malware, which is considered a difficult task.

This paper proposes a merged, image-based malware classification framework (MMCF) to classify malware. MMCF includes local image visualization, global image-based local feature visualization, and global image merging methods. This paper describes the local feature visualization technique for MMCF. The local feature visualization method embeds the local features extracted from malware, and generates local images based on the embedding results. As per our knowledge, there is no known method for generating local images based on the embedding results of local features in malware detection and classification. By creating a local image based on the embedding results, the relationship between all the local features can be represented in a single image. The generated local image has unique features for each family, because it selects important local features for each family of malware. The contribution of this study is as follows:

- FastText model extensibility: The fastText model is used to embed local features, and it aids malware classification.
- New local feature visualization: The local features of malware based on the embedding results are visualized. Both relationship and order between local features in a single image can be considered by visualizing the extracted local features based on embedding.
- Generating a local image for MMCF: A local image with a simple pattern is generated for MMCF. The generated local image helps in applying a local feature visualization method based on the global image proposed by MMCF.

The rest of the paper has the following structure: Section 2 covers related works. Section 3 introduces the local feature visualization method for MMCF. Section 4 derives the experimental procedures and results. Section 5 discusses the results. Section 6 presents the conclusion.

2. Related Works

2.1. Portable Executable

Portable executable (PE) is a file format for executable files utilized by the Windows operating system [18]. PE is the data structure that encapsulates the information required by the windows loader that manages codes, and includes dynamic library references for linking and API import tables. The PE file format could be of different types, such as *.exe file or a *.dll file. A PE file consists of several headers and sections to map file to a memory. Specifically, the text section contains the program code, and the data section contains the global variables. Each section is mapped to a different memory.

2.2. Global Image-Based Malware Detection or Classification

Nataraj et al. proposed a method for visualizing malware to address the shortcomings of static and dynamic analyses [19]. They divided the binary information extracted from malware into an 8-bit vector and used it as one pixel. Because the representation range of 8-bit vectors is between 0 and 255, it is suitable for grayscale images. They extracted texture features from the generated image and used KNN (K-Nearest Neighbors) as a classifier, achieving good performance.

Kesav Kancherla et al. converted the binary value extracted from executable files into an 8-bit vector, and used it as the pixel intensity [20]. They detected malware using a support vector machine (SVM) as a classifier by extracting the intensity, wavelet and three Gabor features from the generated image. They detected and classified malware based on various features using different feature extraction algorithms.

2.3. Local Feature-Based Malware Detection or Classification

Sang Ni et al. proposed a malware classification method using SimHash and Convolutional neural network (MCSC), which combined malware visualization and deep learning [21]. They extracted an opcode sequence from malware files and encoded it using SimHash. They then converted the SimHash result of the extracted opcodes into a grayscale image using the pixel, and verified it through a convolutional neural network (CNN) using 10,805 malware samples.

Jianwen Fu et al. generated a global image using the global features of malware along with local features [17]. They used entropy values, byte values and relative sizes of all the sections for each section from the malware-infected PE files to generate the global image; they extracted the texture and color features from this global image using the gray-level co-occurrence matrix (GLCM) and color moment. They extracted local features from the code and data sections of the malware, and accurately classified malware through the random forest (RF) method by combining global and local features.

2.4. Comparison between Prior Works and Our Proposed Method

Table 1 summarizes the difference between conventional malware detection and classification methods and MMCF. Nataraj and Kesav Kancherla generated a global image using binary information of malware; they detected and classified malware based on the features of the generated global image [19,20]. When detecting and classifying obfuscated malware only with global images, it is difficult to attain accuracy, as the actual malware behavior is not considered. We accurately classify non-obfuscated and obfuscated malware by visualizing the opcodes and API function names that represent malware behavior.

Table 1. Comparison between the proposed method and prior methods.

	Lakshmanan Nataraj [19]	**Keshav [20]**	**Sang Ni [21]**	**Jian Fu [17]**	**MMCF (Proposed)**
Image Variable	Global	Global	Local	Global	Global, Local
Global Feature	Texture	Wavelet Transforms, Gabor Filter	-	GLCM, Color Moments	-
Local Feature	-	-	Opcode	DLL, API	Opcode, API
Model	KNN	SVM	CNN	RF	GAN, CNN

Sing Ni et al. visualized local features using the opcode sequence extracted through static analysis to classify malware into different families [21]. However, there is a problem with their method—the relationships among opcodes cannot be represented when the extracted opcodes are encoded with SimHash. Our method embeds opcodes and API function names through the fastText model and visualizes them, thereby overcoming these shortcomings to a significant extent.

Jianwen Fu et al. classified malware by generating a global image with the global features of malware and extracting local features from the code and data sections of malware [17]. Such extraction of local features from obfuscated malware through static analysis remains to be a difficult task. We therefore propose MMCF, which visualizes local features based on the global images generated by the binary information of non-obfuscated and obfuscated malware; it classifies malware by merging global and local images.

3. Merged Image-Based Malware Classification Framework

3.1. Overview

The proposed method includes the input, preprocessing, and training and classification phases, as illustrated in Figure 1. The input phase extracts ASM and bytes files from a database with a disassembler.

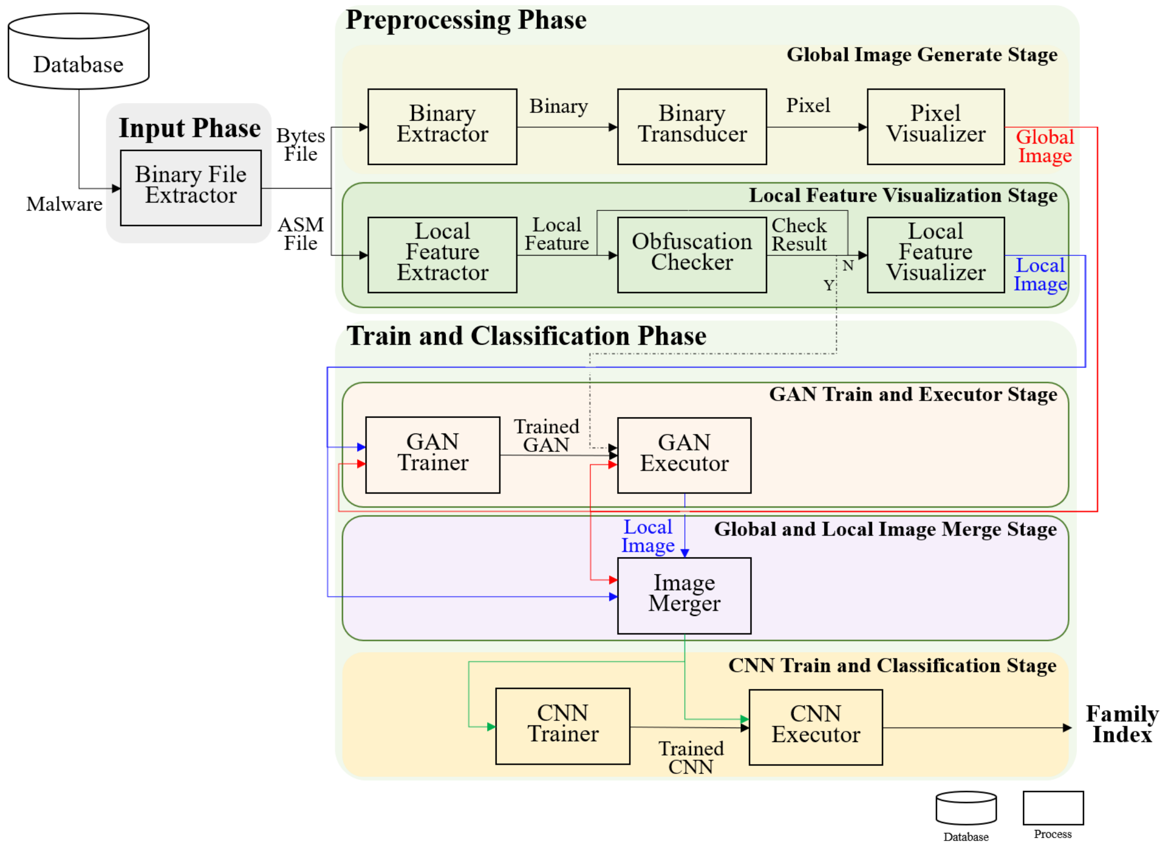

Figure 1. Overview of the merged, image-based malware classification framework (MMCF).

The preprocessing phase includes global image generation and local feature visualization. A global image is generated using the binary information extracted from the bytes file through a binary extractor as pixels. The local features are extracted from ASM files through a local feature extractor, and are visualized. The extracted local features are input into an obfuscation checker to determine whether they are obfuscated. If malware is obfuscated, the local features are entered in a GAN executor. If malware has not been obfuscated, the local features are visualized through the local feature visualizer and then a local image is generated. The phases are as follows:

1. Global and local images are input into a GAN trainer. A global image of the obfuscated malware is input into a GAN executor that outputs a local image of the obfuscated malware.
2. The generated global and local images are merged through an image merger.
3. The merged image is input into a CNN trainer to train the CNN. The CNN executor classifies malware into different families by receiving the merged image and the trained CNN.

The training and classification phase that utilizes a generative adversarial network (GAN) and CNN includes (1) GAN training and execution, (2) global and local image merge and (3) CNN training and classification stages.

3.2. Input Phase

D refers to a database, f refers to the malware family number, and i refers to the malware sample number. Database D consists of malware samples $[m_{f,1}, m_{f,2}, \ldots\ m_{f,i}, \ldots, m_{f,|m_f|}]$. The malware sample $m_{f,i}$ is input into a binary extractor that outputs the ASM file $m^A_{f,i}$ and the bytes file $m^B_{f,i}$.

3.3. Preprocessing Phase

G^L_f refers to a local image set, and G^L_f consists of local images $[g^L_{f,1}, g^L_{f,2}, \ldots, g^L_{f,i}, \ldots, g^L_{f,|g_f|}]$. A local image $g^L_{f,i}$ is an image generated by the processes detailed in Figure 2 using a local feature p_j extracted from the ASM file $m^A_{f,i}$. The text section refers to the section with the program code in the PE file. A feature extractor receives an ASM file $m^A_{f,i}$ and extracts a local feature p_j from the text section of the ASM file $m^A_{f,i}$ based on a predefined list. The local feature p_j is composed of opcodes and API function names. A feature selector receives the local feature p_j and outputs the selected local feature p'_j based on the term frequency inverse document frequency (TFIDF) algorithm [15]. The top Y local features are derived in the ascending order of TFIDF of the local feature p_j for each family. The selected local feature p'_j is derived after removing the same local features and the local features belonging to all families. The fastText model represents words with a similar meaning, among the words inputted through distributed representation, as similar vector values [22]. A fastText trainer learns by receiving the local feature p_j and outputs the trained fastText model t. The fastText executor outputs the embedded local feature p^*_j by receiving the local feature p_j and the trained fastText model t.

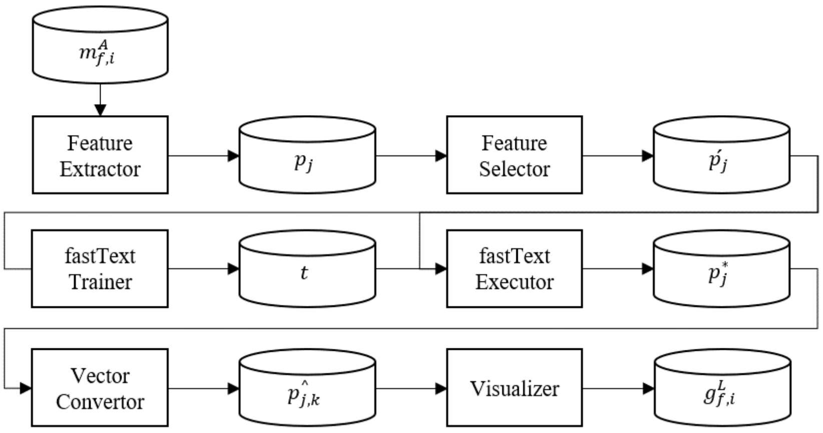

Figure 2. Local feature visualization process.

Algorithm 1 is a local feature visualization algorithm. The local feature visualization function outputs a local image $g^L_{f,i}$ by receiving the non-obfuscated malware $m^U_{f,i}$ and the embedded local feature p^*_j. The range from α to β is the pixel range of a grayscale image. $\Omega_{f,i}$ is a two-dimensional matrix for generating a local image. To generate a local image, the element $p^*_{j,k}$ of the embedded local feature p^*_j is used. The element $p^*_{j,k}$ is a real number ranging from the minimum value $MIN\left(p^*_{j,k}\right)$ to the maximum value $MAX\left(p^*_{j,k}\right)$. The element $p^*_{j,k}$ of the embedded local feature p^*_j is extracted to generate a local image. The range from the minimum value $MIN\left(p^*_{j,k}\right)$ of the element $p^*_{j,k}$ to the maximum value $MAX\left(p^*_{j,k}\right)$ of the element $p^*_{j,k}$ is normalized from α to β.

Because the normalized local feature $\hat{p}_{j,k}$ consists of pixels ranging from α to β of a grayscale image, a local image is generated by using the pixels. The size of the normalized local feature $\hat{p}_{j,k}$ is the same as the size M of the local feature $p^*_{j,k}$ embedded through the fastText model t. The row size of the

2-D matrix $\Omega_{f,i}$ is $SIZE\left(p_j\right)$, which is the size of the local feature p_j extracted from the non-obfuscated malware $m^U_{f,i}$. The column size of the 2-D matrix $\Omega_{f,i}$ is M, which is the size of the embedded local feature p^*_j.

Algorithm 1. Local Feature Visualization Algorithm

```
1.  FUNCTION LocalFeatureVisualization ($m^U_{f,i}$, $p^*_j$)
2.  OUTPUT
3.  $g^L_{f,i}$//Local Image
4.  BEGIN
5.  $\alpha$ ←Local image minimum pixel value
6.  $\beta$ ←Local image maximum pixel value
7.  $\Omega_{f,i}$ ←2-Dimension matrix for local image $g^L_{f,i}$
8.  FOR Zero to $SIZE\left(p_j\right)$
9.  $p^*_{j,k}$←Extract element of $p^*_j$
10. $\hat{p}_{j,k} \leftarrow \frac{p^*_{j,k}-\alpha}{\beta-\alpha} * 255$
11. END FOR
12. FOR Zero to $m \leftarrow SIZE\left(m^U_{f,i}\right)$
13. FOR Zero to $n \leftarrow SIZE\left(\hat{p}_j\right)$
14. $\Omega_{f,i} \leftarrow \hat{p}_{j,k}$
15. END FOR
16. END FOR
17. END
```

4. Experiments

An experiment was conducted to verify the proposed method by performing the local feature visualization process, and deriving its results and malware classification results through MMCF.

4.1. Dataset and Experimental Environment

The dataset used to verify the proposed method is the Microsoft Malware Classification Challenge (BIG 2015) [23]. The BIG 2015 dataset is divided into: (1) training data with label information; and (2) test data without label information. The training and test data consist of ASM files and bytes files extracted from malicious samples through IDA Pro. The datasets composed of 9 families includes 10,868 types of malware with a size of 500 GB. Table 2 details the names and numbers of malware used in the experiment. Ramnit is a worm-type malware, and its total count was 1541, of which 28 were obfuscated. The total number of Lollipop malware was 2478, of which 8 were obfuscated. Vundo, Tracur, Obfuscator.ACY, and Gatak are Trojan-type malware, and their total count was 3467, of which 544 were obfuscated. Kelihos_ver3 and Kelihos_ver1 are botnet-type malware, and their total count was 3340, of which 17 were obfuscated. Simda is backdoor-type malware. In the experiment, the 10,868 ASM files and 10,868 bytes files were used among the training data with label information, because the proposed method could not verify data without label information. A total of 90% of the training data was used for training and 10% for testing.

Table 2. Malware used in the experiment.

Family Index	Family Name	Non-Obfuscated Malware	Obfuscated Malware	Total Number
1	Ramnit	1513	28	1541
2	Lollipop	2470	8	2478
3	Kelihos_ver3	2936	6	2942
4	Vundo	447	28	475
5	Simda	34	8	42
6	Tracur	294	457	751
7	Kelihos_ver1	387	11	398
8	Obfuscator.ACY	1170	58	1228
9	Gatak	1012	1	1013

Table 3 lists the parameters used in the experiment; batchsize is the number of images inputted at a time, and imageshape is the size of an image. The 32 × 32 local images outputted from the GAN model are reshaped into 256 × 128 images. The learning rate is represented by learningrate, and epoch is the learning number. Similarly, filter_size is the size of the filter, G_h0 is the size of the first layer of the generator, G_h1 is the size of the first CNN layer of the generator, G_h1 is the size of the second CNN layer of the generator, G_h3 is the size of the output layer of the generator, D_h0 is the size of the first CNN layer of the discriminator, D_h1 is the size of the second CNN layer of the discriminator, D_h1 is the size of the third CNN layer of the discriminator, D_h3 is the size of the output layer of the discriminator, conv1 is the size of the first CNN layer, conv2 is the size of the second CNN layer, conv3 is the size of the third CNN layer, fc1 is the size of the first FC layer and fc2 is the size of the second FC layer.

Table 3. Generative adversarial network (GAN) and convolutional neural network (CNN) parameters.

No	GAN		CNN	
	Parameter	Value	Parameter	Value
1	batchsize	32	batchsize	64
2	imageshape	[256, 256]	imageshape	[32, 32, 1]
3	learningrate	1×10^{-4}	learningrate	0.0002
4	epoch	50	epoch	10
5	filter_size	3 × 3	filter_size	5 × 5
6	conv1	128 × 128 × 32	G_h0	4 × 4 × 128
7	conv2	64 × 64 × 32	G_h1	8 × 8 × 64
8	conv3	32 × 32 × 64	G_h2	16 × 16 × 32
9	fc1	128	G_h3	32 × 32 × 1
10	fc2	9	D_h0	16 × 16 × 32
11	-		D_h1	8 × 8 × 64
12			D_h2	4 × 4 × 128
13			D_h3	64 × 1

4.2. Local Feature Visualization Results

Table 4 lists the selected local features in descending order of the TFIDF values, spanning the results of embedding and normalization and of local feature visualization for non-obfuscated and obfuscated malware. The same local features were selected from the top 1–3 of the selected local features for each family, but different local features were selected from the top 4. The results are derived by extracting the top X opcodes and API function names for each family of malware, and removing the duplicates. These results prove that different families of malware act differently. The top 45, 50, 55, 60 and 65 with high TFIDF values are selected to visualize the local image.

Table 4. Term frequency inverse document frequency (TFIDF) algorithm results.

	45	**50**	**55**	**60**	**65**
1	aad	aad	aad	aad	aad
2	aam	aam	aam	aam	aam
3	adc	adc	adc	adc	adc
4	arpl	arpl	addss	addss	addss
5	bswap	bswap	arpl	arpl	arpl
6	div	div	bound	bound	bound
			...		
92	-	-	-	-	xchg
Total Number	67	70	76	85	92

Table 5 summarizes the extracted opcodes and API function names, embedding results and normalization results. Because the embedded opcodes and element values of API function names are the most widely distributed between −1 and 1, pixels are defined by normalizing the values between −1 and 1 of the elements to the values between 0 and 255. If the element value is less than −1, a pixel is defined as 0, and if it is greater than 1, it is defined as 255. By normalizing the embedded results to values between 0 to 255, which is the pixel range of a grayscale image, embedded results were included in a single image; the image is one of malware, which is included in the relationship between opcodes and API function names.

Table 5. Embedding and normalization results.

No	**Extracted Local Feature**	**Embedding Results**	**Normalization Results**
1	in	[−3.93824339, −1.38366544, −4.90960407, ... , 3.91097903]	[0, 0,0, ... , 255]
2	call	[$1.96547285 \times 10^{-1}$, $2.34460935 \times 10^{-1}$, $-8.65871787 \times 10^{-1}$, ... , $8.48949492 \times 10^{-1}$]	[152, 157, 17, ... , 19]
3	jmp	[0.28522536, −2.9717305, −1.2783291, ... , −0.60219544]	[163, 0, 0, ... , 50]
		...	
1369	SetBitmapDimensionEx	[−0.05661994, 0.22015877, −0.37183163, ... , −0.1277246]	[120, 155, 80, ... , 111]

Figure 3 presents the results of opcode and API function name visualization of each family of malware. The opcodes and API function names of each family of malware, which have been imaged through the proposed method, exhibit unique patterns. The trained GAN receives a global image of obfuscated malware, and it outputs a local image. The outputted local image is normalized to the 256 × 128 size. Although unique patterns of the local image of obfuscated malware generated by the GAN model are identified, families 2 and 8 exhibited similar patterns. The local image is ideal for classifying malware into different families because unique patterns have been derived from each family of malware.

4.3. Malware Classification through MMCF

Figure 4 presents the loss values of the CNN to classify obfuscated and non-obfuscated malware in each family. The loss value represents the difference between the predicted and actual values. The loss value of the CNN started with 2.6 at the first iteration, became 0.268 at the fifth iteration, and converged to 0.0543 at the 2701st iteration.

Figure 5 details the learning accuracy of the CNN for classifying obfuscated and non-obfuscated malware in each family. Accuracy is the rate at which the malware family predicted by the CNN is included in the actual malware family. The learning accuracy that started with 15.6% at the first iteration reached 84.4% at the 270th iteration, and converged to 100% at the 2701 iteration.

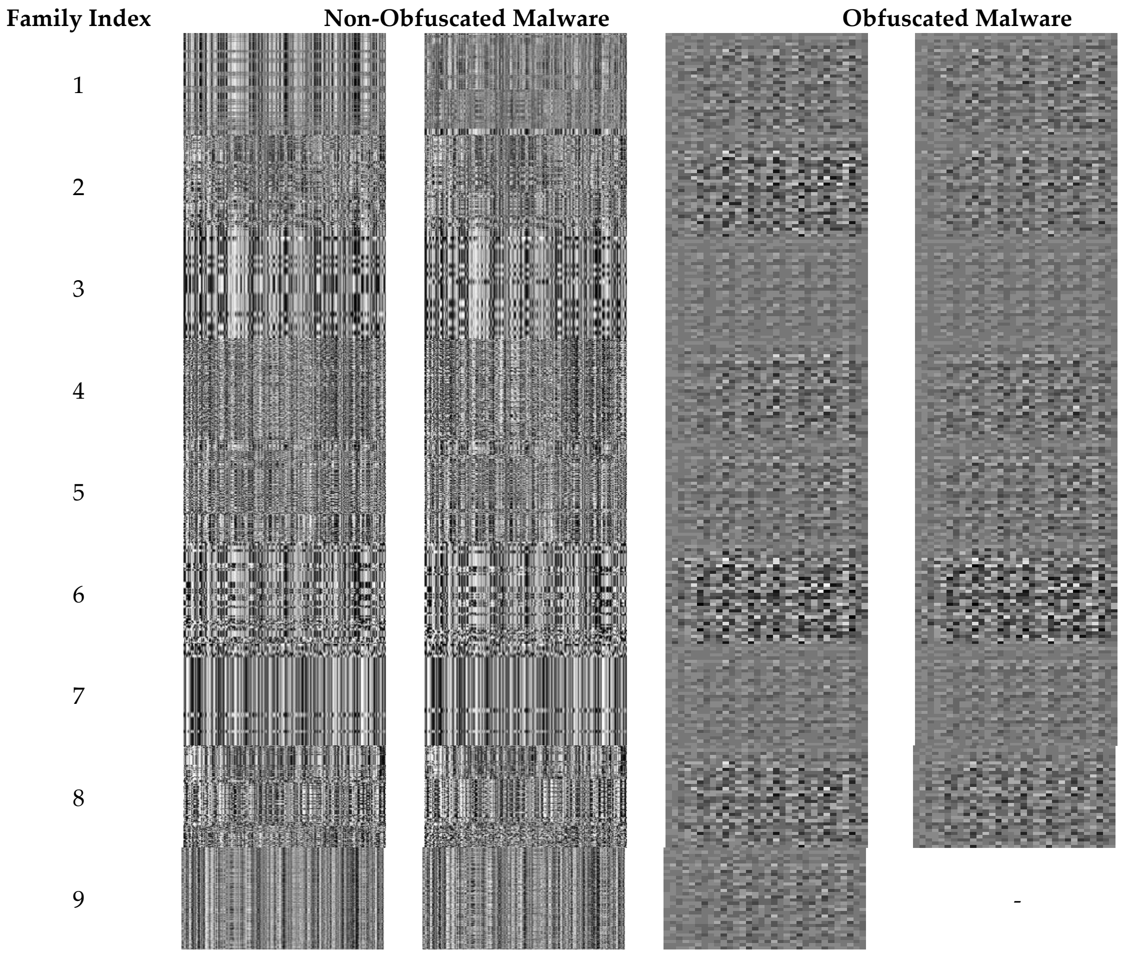

Figure 3. Local feature visualization results.

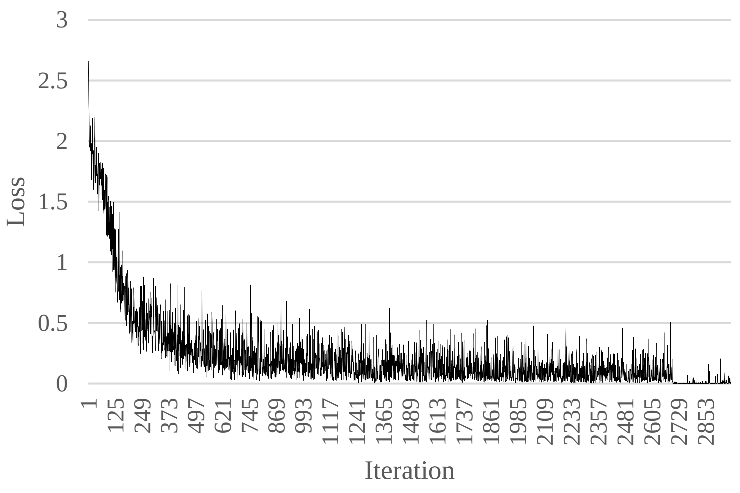

Figure 4. Learning loss of the CNN.

Table 6 lists the accuracy of malware classification achieved by the proposed method at 99.65% accuracy for each family. The method proposed by Jianwen Fu [17] classifies malware into different families; a global image is generated using a global feature and a local feature extracted from malware. The method is similar to the proposed method, in that it uses local features with images of malware. The performance of the method proposed in this study was 2.18% better than that of the method proposed by Jianwen Fu [17]. The method proposed by Sang Ni [21] classifies malware into different families; local images are created using local features extracted from malware. The performance of the method proposed in this study was 0.39% better than that of the method proposed by Sang Ni [21]. Furthermore, in comparison with the global image-based malware detection and classification method proposed by Nataraj [19] and Kancherla [20], the method proposed in this study achieved a 1.6% higher accuracy.

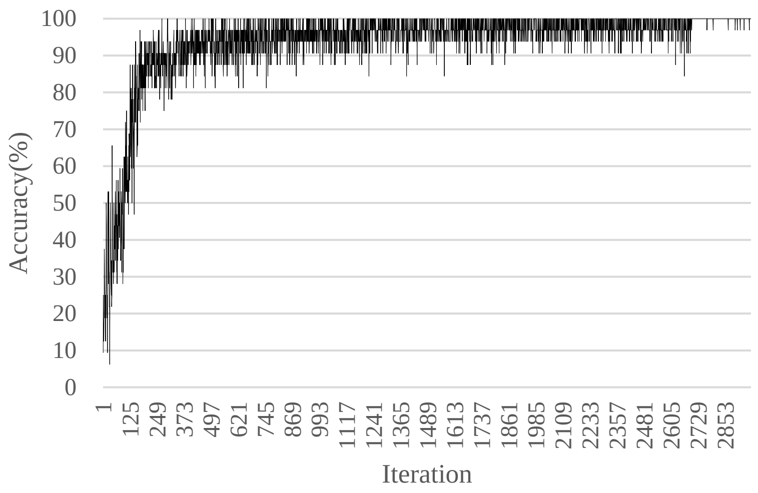

Figure 5. Learning accuracy of the CNN.

Table 6. Comparison between the proposed method and methods introduced in related works.

	Accuracy (%)	**TFIDF (Top)**	**Used Image**
Proposed Method	99.39	45	Merge Image
Proposed Method	99.47	50	Merge Image
Proposed Method	99.65	55	Merge Image
Proposed Method	99.56	60	Merge Image
Proposed Method	99.39	65	Merge Image
Jianwen Fu et al. [17]	97.47	-	Global Image, Local Feature
Sang Ni et al. [21]	99.26	-	Local Image
Nataraj et al. [19]	98.00	-	Global Image
Kancherla et al. [20]	95.95	-	Global Image

5. Discussion

5.1. Non-Obfuscated and Obfuscated Malware Classification Results Achieved by the Proposed MMCF

The non-obfuscated malware classified all the 1024 types of test malware for each family, resulting in 100% accuracy. The obfuscated malware classified 124 out of 128 types for each family, resulting in 96.87% accuracy. The local image of the obfuscated malware based on the global image was generated through the GAN model. However, the generated local image was inaccurate in comparison with the local image of the non-obfuscated malware. The classification accuracy of the obfuscated malware was lower than that of non-obfuscated malware, because the unique patterns of each family were not clearly displayed. Table 7 summarizes the confusion matrix that classifies obfuscated malware into different families. Yellow color in Table 7 means the numbers of accurate classification and red color means that of inaccurate classification.

Table 7. Confusion matrix that classifies obfuscated malware. (Yellow shading means the numbers of accurate classification and red shading means that of inaccurate classification.)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	5	0	0	0	0	0	0	0	0
2	0	4	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	1	0
4	0	0	0	6	0	0	0	0	0
5	0	0	0	0	2	0	0	0	0
6	0	0	0	0	0	86	0	0	0
7	0	0	0	0	0	0	4	0	0
8	0	0	0	0	0	1	1	16	0
9	0	0	0	0	0	1	0	0	0

5.2. *Comparison between the Results of the Proposed MMCF and Those of Other Research*

The malware classification technique based on the global image and local features (text) proposed by Jian Fu et al. [17] is similar to the method proposed in this study. However, MMCF, using the proposed global and local images, yields a 2.18% higher accuracy. The local image-based malware classification method proposed by Sang Ni et al. [21] used the same dataset as the proposed MMCF. Comparing MMCF (with the global and local images) to the method proposed by Sang Ni (with local images) [21], the result of the former is 0.39% better than that of the latter. On one hand, Sang Ni et al. [21] experimented using only 10,805 of the 10,868 BIG 2015 datasets. On the other hand, MMCF was experimented using all the 10,868 BIG 2015 datasets. Even though Sang Ni et al. [21] experimented with limited datasets, MMCF yielded a higher accuracy.

5.3. *Computational Complexity*

The proposed MMCF has a strong advantage, in that it delivers higher accuracy than other methods proposed in recent research studies; however, it has the disadvantage of increased computational complexity. When detecting or classifying malware in real-time, computational complexity is one of the important considerations. However, this study does not consider computational complexity, given that the proposed MMCF focuses on demonstrating the availability of fastText model, which is one of embedding models frequently utilized in natural language processing to express the co-relationship of malware local features by a single image. To further explore the computational complexity of MMCF, the reduction of the computational complexity of generating a local image using embedded models will be studied in future.

5.4. *Security of Big Data*

Big data contains a wide variety of data. Information protection for big data is necessary, especially when it contains personal information or important company information. My T. Thai et al. describe the applications of big data and social networks, and the protection of privacy and security [24].

If a system that handles big data is infected with malware, the root authority of the system is hijacked. Personal information or confidential information contained in the database is stolen easily if the hacker gains root authority of the system that controls the database. Therefore, the ability to detect and classify malware before their execution is important, because malwares are considered as the starting point of cyberattacks.

6. Conclusions

MMCF offers three methods—local feature visualization, global image-based local feature imaging and global and local image merging methods. This paper described the local feature visualization method. First, the ASM and bytes files were extracted from a database. Second, the local features of each family of malware were selected, based on the TFIDF algorithm. Third, the selected local features were embedded through fastText. Fourth, the embedding results were normalized for each local feature to use them as pixels. Fifth, a local image was generated using the normalized results. Sixth, based on the generated local and global images, malware were classified into different families.

The performance of the proposed method was experimentally verified as follows: First, the selected local feature results based on TFIDF were derived. Second, the embedding results through fastText and the normalized results through the embedding results were derived. Third, the local feature visualization results of obfuscated and non-obfuscated malware based on the normalized results were derived. In comparison with the method proposed by Jianwen Fu, which is the most similar to the proposed method, the proposed method achieved approximately 2.18% higher performance.

Future work will focus on improving the local image of obfuscated malware visualized based on a GAN. Because the derived local image is blurry, it is necessary to obtain a simpler pattern to generate an image based on the GAN. Methods for reducing the number of local features extracted from malware or selecting meaningful local features will also be studied. Also, MMCF will improve the ability to detect files as malicious/benign, and classify malicious files (malware) into each family.

Author Contributions: Conceptualization, S.L., S.J., and Y.S.; Methodology, S.L., J.S., and Y.S.; Software, S.L., S.J., and Y.S.; Validation, S.L., S.J., and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program (2019-0-01585) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwak, J.; Park, J.; Sung, Y. Affective Social Big Data Generation Algorithm for Autonomous Controls by CRNN-based end-to-end Controls. *Multimed. Tools Appl.* **2019**, *78*, 27175–27192. [CrossRef]
2. Li, S.; Jang, S.; Sung, Y. Automatic Melody Composition Using Enhanced GAN. *Mathematics* **2019**, *7*, 883–893. [CrossRef]
3. Sung, Y.; Kwak, J.; Park, J. Decision Tree Generation Algorithm for Image-based Video Conferencing. *J. Intern. Technol.* **2019**, *20*, 1535–1545.
4. Stai, E.; Kafetzoglou, S.; Tsiropoulou, E.E.; Papavassiliou, S. A Holistic Approach for Personalization, Relevance Feedback & Recommendation in Enriched Multimedia Content. *Multimed. Tools Appl.* **2018**, *77*, 283–326.
5. Balabanović, M.; Shoham, Y. Fab: Content-Based, Collaborative Recommendation. *Commun. ACM* **1997**, *40*, 66–72. [CrossRef]
6. Sangaiah, A.K.; Medhane, D.V.; Bian, G.B.; Ghoneim, A.; Alrashoud, M.; Hossain, M.S. Energy-Aware Green Adversary Model for Cyber Physical Security in Industrial System. *IEEE Trans. Ind. Inform.* **2020**, *16*, 3322–3329. [CrossRef]
7. Sangaiah, A.K.; Hosseinabadi, A.A.R.; Sadeghilalimi, M.; Zhang, W. Energy Consumption in Point-Coverage Wireless Sensor Networks via Bat Algorithm. *IEEE Access* **2019**, *7*, 180258–180269. [CrossRef]
8. Bilar, D. Opcodes as Predictor for Malware. *Int. J. Electron. Secur. Digit. Forensics* **2007**, *1*, 156–168. [CrossRef]
9. Albladi, S.; Weir, G.R. User Characteristics that Influence Judgment of Social Engineering Attacks in Social Networks. *Hum. Centric Comput. Inf. Sci.* **2018**, *8*, 1–24.
10. Gandotra, E.; Bansal, D.; Sofat, S. Malware Analysis and Classification: A Survey. *J. Inf. Secur.* **2014**, *5*, 56–64. [CrossRef]
11. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection. *Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
12. Souri, A.; Hosseini, R. A State-of-the-Art Survey of Malware Detection Approaches using Data Mining Techniques. *Hum. Centric Comput. Inf. Sci.* **2018**, *8*, 1–22. [CrossRef]
13. Homayoun, S.; Dehghantanha, A.; Ahmadzadeh, M.; Hashemi, S.; Khayami, R. Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence. *IEEE Trans. Emerg. Top. Comput.* **2017**. to appear. [CrossRef]

14. Zhao, B.; Han, J.; Meng, X. A Malware Detection Sysstem Based on Intermediate Language. In Proceedings of the 2017 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, China, 11–13 November 2017.
15. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of Ransomware Families with Machine Learning based on N-gram of Opcodes. *Futur. Gener. Comput. Syst.* **2019**, *90*, 211–221. [CrossRef]
16. Kim, J.; Kim, H.; Kim, I.K. Cyber Genome Technology for Countering Malware. *Electron. Telecommun. Trends* **2015**, *30*, 118–128.
17. Fu, J.; Xue, J.; Wang, Y.; Liu, Z.; Shan, C. Malware Visualization for Fine-grained Classification. *IEEE Access* **2018**, *6*, 1–14. [CrossRef]
18. Bai, J.; Wang, J.; Zou, G. A Malware Detection Scheme Based on Mining Format Information. *Sci. World J.* **2014**, *2014*, 1–12. [CrossRef] [PubMed]
19. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. *Malware Images: Visualization and Automatic Classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec '11)*; Association for Computing Machinery: New York, NY, USA, 20 July 2011.
20. Kancherla, K.; Mukkamala, S. Image Visualization based Malware Detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16–17 April 2013.
21. Ni, S.; Qian, Q.; Zhang, R. Malware Identification Using Visualization Images and Deep Learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
22. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
23. Kalash, M.; Rochan, M.; Mohammed, N.; Bruce, N.D.; Wang, Y.; Iqbal, F. Malware Classification with Deep Convolutional Neural Networks. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018.
24. Thai, M.T.; Wu, W.; Xiong, H. *Big Data in Complex and Social Networks*, 1st ed.; Taylor & Francis Inc: Portland, OR, USA, 2016.