

# High-Accuracy Malware Classification with a Malware-Optimized Deep Learning Model

*Rikima Mitsuhashi*  
The University of Tokyo  
mitsuhashi@os.ecc.u-tokyo.ac.jp

*Takahiro Shinagawa*  
The University of Tokyo  
shina@ecc.u-tokyo.ac.jp

## Abstract

Malware threats are a serious problem for computer security, and the ability to detect and classify malware is critical for maintaining the security level of a computer. Recently, a number of researchers are investigating techniques for classifying malware families using malware visualization, which convert the binary structure of malware into grayscale images. Although there have been many reports that applied CNN to malware visualization image classification, it has not been revealed how to pick out a model that fits a given malware dataset and achieves higher classification accuracy. We propose a strategy to select a Deep learning model that fits the malware visualization images. Our strategy uses the fine-tuning method for the pre-trained CNN model and a dataset that solves the imbalance problem. We chose the VGG19 model based on the proposed strategy to classify the Maling dataset. Experimental results show that the classification accuracy is 99.72 %, which is higher than other previously proposed malware classification methods.

## 1 Introduction

Malware threats are a serious problem for computer security, and the ability to detect and classify malware is critical for maintaining the security level of a computer. Malware analysis has a static analysis approach and a dynamic analysis approach. Static analysis approaches disassemble malware code, reveal its execution logic and find patterns that trigger attack behavior. The dynamic analysis approach, on the other hand, runs malware in a virtual environment and obtains a report that traces the movement to identify the characteristics of the attack behavior. The static analysis approach provides a complete picture of the program structure by reading the malware code from start to finish. However, in general, Malware authors obfuscate program code by packing, which hinders this approach [26]. Although the dynamic analysis approach does not matter if the program code is obfuscated, typically only a single execution path is examined [29] and

this can lead to an incomplete comprehension of malware activity.

In addition to the approach described above, many researchers are investigating technique for classifying malware families using malware visualization images [26]. This technique converts the structure of the malware binary sample into a two-dimensional grayscale image and uses the image features for classification. As a result, even if the malware code is obfuscated, it can generate images without being obstructed [26]. Moreover, there's no need to execute malware programs.

In the field of image recognition, the Convolutional Neural Network (CNN) [14], which is recognized as one of the representative methods of Deep Learning [17], is widely applied. For example, CNN has been applied to the classification of fruit images [32], medical images [36], criminal investigation images [19], tread pattern images [20], and face images [15].

Besides, there have been many reports that applied CNN model to malware visualization image classification [26] [4] [27] [25] [13] [23]. Unfortunately, it has not been revealed the way how to pick out a CNN model that fits a given malware dataset and achieves higher classification accuracy.

We propose a strategy to select a Deep learning model that fits the malware visualization images. Our strategy uses the fine-tuning method for the pre-trained CNN model and a dataset that solves the imbalance problem. We have selected the VGG19 model based on the proposed strategy to classify the Maling dataset. Experimental results show that the classification accuracy is 99.72%, which is higher than other previously proposed malware classification methods.

Our contribution of the research can be organized as follows:

- We proposed a strategy to select a CNN model that fits the malware visualization images using fine-tuning method.
- We solved the problem of sample data imbalance between malware families using the undersampling technique.

- We selected the VGG16 fine-tuning model and achieved high accuracy of classifying malware family.

This paper is organized as following: Section 2 presents related work on malware detection and classification; Section 3 introduces design for strategy comparing the performance between CNN models using fine-tuning; Section 4 shows experimental settings; Section 5 provides experimental results; Section 6 concludes the paper.

## 2 Related Work

we present the related research regarding malware detection and classification including static analysis approach, dynamic analysis approach, malware visualization image in this section.

### 2.1 Static Analysis Approach

In relation to static analysis approach, several methods have been proposed for analyzing the code of malware. For example, Lo et al. [22] developed a static analysis tool called Malicious Code Filter (MCF). The tool used some program property features to determine if the program is malicious or not. Masud et al. [24] combined three types of features, binary n-grams, assembly instruction sequences, and dynamic link libraries (DLLs) to detect malicious executables. Then, a classifier for malware detection was built based on SVM and Boosted J48. Yakura et al. [38] used CNN attention maps to extract byte sequences that characterize malware families from malware binary data. Jung et al. [12] disassembled the malware binary file and extracted the bytecode. Next, the bytecode was converted to an image, and malware was classified based on the convolutional neural network. Sewak et al. [31] investigated a deep learning-based system for malware detection. They prepared one and three layers of Auto-Encoders and two, four and seven layers of Deep Neural Networks, and evaluated the results of each combination. Dam and Touili [5] proposed a tool called STAMAD. The tool automatically classified malware and benign programs using either extracting API graphs that represent malicious behavior, or machine learning based on SVM.

### 2.2 Dynamic Analysis Approach

Regarding the dynamic analysis approach, a number of methods have been proposed to execute malware and analyze its behavior. For example, Rhode et al. [28] proposed a model for detecting malicious files based on a few seconds of the initial action sequence executed by the malware. The set of API calls obtained by executing the malware PE file in the Cuckoo Sandbox was sent to the recurrent neural network for analysis. Xiaofeng et al. [37] proposed an architecture

for detecting malware by combining machine learning (random forest) and deep learning (LSTM) using an API call sequence. S.L and CD [34] proposed a CNN-based windows malware detector that detects and classifies malware using the behavior of Portable Executable (PE) files. The proposed method executed malware PE files in Sandbox, and obtains N-grams of API calls. Liu and Wang [21] implemented a malware detection system based on deep learning and API calls. They used the Cuckoo Sandbox to extract the API calling sequence of a malicious program and evaluated it with BLSTM. Li et al. [18] proposed a technique for detecting kernel-resident malware using the location of the page global directory and the instruction set of the processor. They implemented the technique in a tool called Fluorescence and showed that 200 virtual machines could be inspected in about an hour.

### 2.3 Generating hostile attacks against malware detection

In recent years, research results have been reported that generate not only malware detection methods but also attack methods that invalidate malware detection. Grosse et al. [8] has extended existing hostile sample creation algorithms to build effective attacks against malware detection models. This approach works in the discrete binary input domain. Rosenberg et al. [30] generated a hostile example of an RNN malware classifier based on API calls. They have shown that the proposed attack is feasible by implementing a black box attack.

### 2.4 Malware Visualization Approach

As for the method of malware classification using malware visualization image, many researchers have reported results using the Maling dataset. Nataraj et al. [26] selected GIST to extract image features and classified them using k-nearest neighbors. As a result, they obtained a classification accuracy of 97.18% . Nonetheless, the confusion matrix showed that there was some confusion between "C2Lop.P" and "C2Lop.gen! G". Then there was a similar confusion between "Swizzor.gen! I" and "Swizzor.gen! E". As the name implied, each malware family had similar characteristics. Therefore, a new approach was needed to classify these families. Kosmidis and Kalloniatis [16] classified malware images using machine learning techniques such as Decision Tree, Nearest Centroid, Stochastic Gradient, Perceptron, Multilayer Perceptron, and Random Forest. Of these techniques, the best result was Random Forest with 91.6% accuracy. Cui et al. [4] used CNN to extract image features and classify malware families. In the paper, the number of data belonging to each family is equalized before learning based on data augmentation. As a result of the experiment, the classification accuracy reached 94.5%. Furthermore, Precision and Recall

were almost the same. This indicates that not only malware families with large data samples were classified correctly, but also that malware families with small data were classified with a certain degree of accuracy.

In addition, various methods using deep learning models have been reported. Rezende et al. [27] prepared a ResNet50 model in which all convolutional layer parameters were transferred from a model previously trained on an ImageNet dataset. Experimental results indicate that malware families could be classified with an accuracy of 98.62%. Mourtaji et al. [25] report that the VGG16 model was used for classification with an accuracy of 97.02%. Kalash et al. [13] state that malware families were classified with an accuracy of 98.52% using M-CNN based on VGG16. Lo et al. [23] indicate that the Xception model achieved a classification accuracy of 99.03%. These four reports, nevertheless, do not sufficiently reveal how to pick out a model that fits a given malware dataset. Moreover, since these reports show neither confusion matrices nor evaluation metrics such as Precision and Recall, it is not clear how accurately these proposed models classified malware families containing only small data samples.

### 3 Strategy Design

This section reveals our strategy design.

Our strategy uses the fine-tuning method for the pre-trained CNN model and a dataset that solves the imbalance problem.

#### 3.1 CNN fine-tuning model

We classify malware visualization images using a convolutional neural network pre-trained using ImageNet. ImageNet [6] is a database that stores over 14 million color images and is intended for use by researchers. As shown in figure 1 [1], the classification accuracy of each CNN model has been clarified. As an example of a strategy for classifying malware visualized images, you can select a model with as high a classification accuracy as possible. However, note that this metric does not indicate general performance for all datasets. We believe that the compatibility of the model with the dataset is an important factor when choosing a model. Therefore, we decided to check the compatibility between the dataset and the deep learning model by evaluating the performance of learning the dataset that solved the imbalance problem.

Fine-tuning is widely used technique for model reuse, and it has several unfrozen layers in the pre-trained model [7]. Our strategy uses fine-tuning with unified conditions to compare the performance between CNN models. The condition is to freeze 80 % of the convolutional layers and train the remaining 20 % on image features. The main reason for setting this condition is that Tajbakhsh et al. [35]

Table 1: Deep Neural Network benchmarks.

Model	Year	FLOPs (M)	Params (M)	Accuracy (top-1)	Accuracy (top-5)
AlexNet	2012	955.21	61.10	56.52	79.07
VGG11	2014	8171.57	132.86	69.02	88.63
VGG13	2014	11895.04	133.05	69.93	89.25
VGG16	2014	16063.36	138.36	71.59	90.38
VGG19	2014	20231.68	143.67	72.38	90.88
ResNet18	2015	1836.82	11.69	69.76	89.08
ResNet34	2015	3692.78	21.80	73.31	91.42
ResNet50	2015	4154.96	25.56	76.13	92.86
ResNet101	2015	7892.77	44.55	77.37	93.55
ResNet152	2015	11636.60	60.19	78.31	94.05
Inception-v3	2015	5730.17	27.16	75.64	92.59
Inception-v4	2015	12561.10	42.68	80.08	94.89
SqueezeNet-v1	2016	865.78	1.25	58.09	80.42
SqueezeNet-v1.1	2016	377.80	1.24	58.18	80.62
DenseNet 121	2017	2928.89	7.98	74.43	91.97
DenseNet 169	2017	3473.88	14.15	75.60	92.81
DenseNet 201	2017	4435.03	20.01	76.87	93.37
DenseNet 161	2017	7902.37	28.68	77.14	93.56
Xception	2017	8494.59	22.86	78.89	94.29
MobileNetV2	2017	336.43	3.50	71.81	90.42
ShuffleNet-v2.05	2018	52.32	1.37	60.55	81.75
ShuffleNet-v2.1	2018	160.09	2.28	69.36	88.32
MnasNet	2018	649.51	4.38	61.95	84.73
PNASNet	2018	25945.87	86.06	82.74	95.99
NasNet	2018	24882.21	88.75	82.51	96.02
NasNet mobile	2018	667.75	5.29	74.08	91.74

evaluated the adaptation between fine-tuning and the dataset, and they found that neither shallow tuning nor deep tuning was the optimal select for a particular image dataset.

Subsequently, we discard the original fully-connected layer prepared for classifying 1000 classes from each fine-tuning models. Then we designed new fully-connected layers to classify the 25 classes and added them to each model. We also train the parameters of this new fully-connected layer by image features. Our proposed model architecture is shown in Figure 1. The fully-connected layers shown in Figure 1 are used in all our experiments.

#### 3.2 Approach to imbalanced dataset

In general, models generated by directly learning imbalanced data have poor classification accuracy for classes with a small number of samples. Therefore, there are many research reports on the problem of data imbalance. The class imbalance approaches can be divided into two main categories, 1) data-level, and 2) algorithm-level approaches [9]. The data-level approaches improve the distribution of data such as oversampling the minority classes or undersampling the majority classes. On the other hand, the algorithm level approaches adjust the classifier itself based on cost-oriented learning.

We use the Maling dataset [2] created by Nataraj et al. [26] to classify malware visualization images. Maling is a dataset of 9339 images, including 25 malware families. According to a report from the author, the malware based on this dataset is wild malware submitted to the Anubis

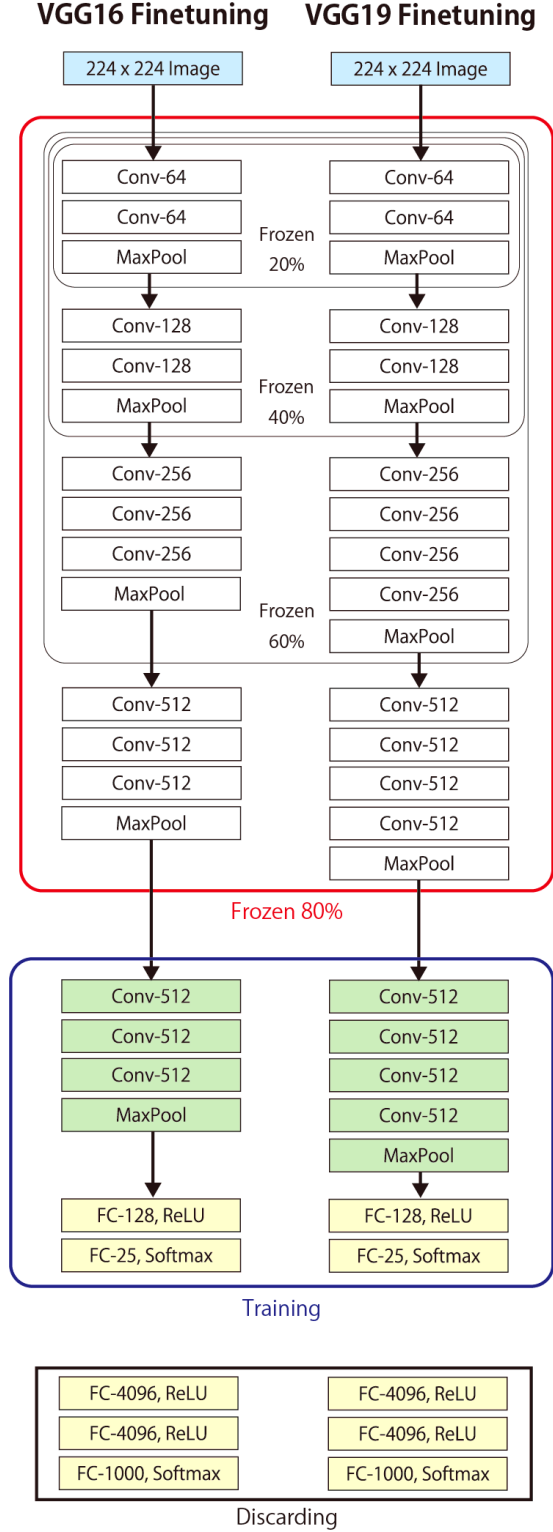


Figure 1: Example of our fine-tuning model and fully-connected layer.

analysis system. Each malware is labeled by Microsoft Security Essentials and classified into different malware families. In this paper, the number of malware image samples is imbalanced, as shown in Figure 2 [4]. As for the number of images belonging to each family, Allapple.A has 2949 images, while Skintrim.N, WinTrim.BX and Autorun.K have only 80, 97 and 106 images respectively. Among the several approaches to imbalanced data, we adopt the undersampling method. The main reason is that most families in the Malimg dataset have roughly similar sample sizes, except for the three families Allapple.A, Allapple.L, and Yuner.A.

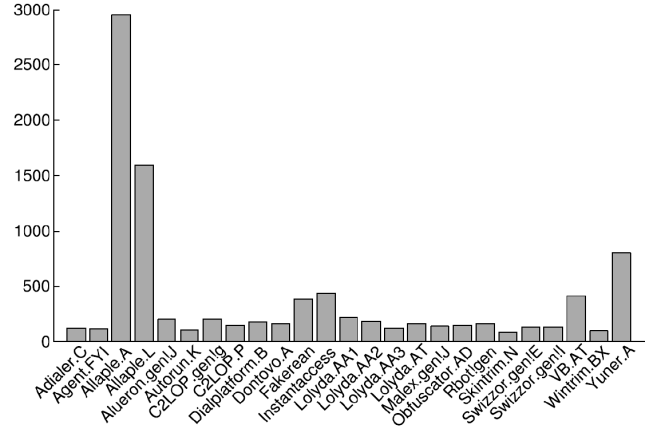


Figure 2: Number of malware images in different families.

## 4 Experimental Settings

This section describes our experimental settings. The contents are composed of Dataset, Evaluation metrics, and Evaluation environment.

### 4.1 Dataset

We undersample the Malimg dataset and set the maximum number of samples belonging to each malware family to 80 so that all classes have the same number of samples. Yet, as a result of undersampling, important features may be lost. so, we also prepare groups that set the maximum number of samples to 160, 240, and 320. Example of the number of image samples belonging to each class is as shown in the table 2. We kept 90% of the dataset as training data and the rest as test data in all experiments. Note that we did not use data augmentation in the training process in all experiments.

### 4.2 Evaluation metrics

We use the accuracy, precision, and recall for evaluation metrics. These evaluation metrics are adopted frequently in

Table 2: The number of samples for our experiment.

No.	Family	Original	Max320	Max240	Max160	Max80
1	Allaple.L	1591	320	240	160	80
2	Allaple.A	2949	320	240	160	80
3	Yuner.A	800	320	240	160	80
4	Lolyda.AA1	213	213	213	160	80
5	Lolyda.AA2	184	184	184	160	80
6	Lolyda.AA3	123	123	123	123	80
7	C2Lop.P	146	146	146	146	80
8	C2Lop.gen!g	200	200	200	160	80
9	Instantaccess	431	320	240	160	80
10	Swizzor.gen!l	132	132	132	132	80
11	Swizzor.gen!E	128	128	128	128	80
12	VB.AT	408	320	240	160	80
13	Fakerean	381	320	240	160	80
14	Alueron.gen!J	198	198	198	160	80
15	Malex.gen!J	136	136	136	136	80
16	Lolyda.AT	159	159	159	159	80
17	Adialer.C	122	122	122	122	80
18	WinTrim.BX	97	97	97	97	80
19	Dialplatform.B	177	177	177	160	80
20	Dontovo.A	162	162	162	160	80
21	Obfuscator.AD	142	142	142	142	80
22	Agent.FYI	116	116	116	116	80
23	Autorun.K	106	106	106	106	80
24	Rbot!gen	158	158	158	158	80
25	Skintrim.N	80	80	80	80	80
Total		9339	4699	4219	3565	2000

the research community to provide comprehensive assessments of imbalanced learning problems [4]. These metrics are defined as follows:

$$\begin{aligned}
precision &= TP/(TP + FP) \\
recall &= TP/(TP + FN) \\
accuracy &= (TP + TN)/(TP + TN + FP + FN)
\end{aligned}$$

where true positive (TP) and false positive (FP) are the number of samples correctly and wrongly classified as malware. Likewise, true negative (TN) and false negative (FN) are the number of samples correctly and wrongly classified as benign.

### 4.3 Evaluation environment

This experiment uses the environment shown in Table 3 and the hyperparameter shown in Table 4.

## 5 Experimental Result

In this section, we shows our experimental results. We first evaluated the models using a dataset that solved the data imbalance problem and Fine-tuning models pretrained on ImageNet. Next, we investigated the optimal Fine-tuning and undersampling for the model we selected, and showed the

Table 3: Experimental environment

CPU	Intel Core i9-9900K
Memory	32GB
GPU	Nvidia GeForce RTX 2080Ti
OS	Ubuntu 18.04 LTS 64bit
Code	Keras on top of TensorFlow

Table 4: The hyper parameters for CNN models

Loss Function	Categorical Cross Entropy
Optimizer	SGD
Learning Rate	1e-4
Momentum	0.9
Batch Size	32

results that classified the entire Maling dataset. Finally, we compared our experimental results with previous reports.

### 5.1 Selecting models that fits the dataset

We have prepared models pre-trained on ImageNet: NasNet [39], DenseNet201 [11], Xception [3], ResNet50 [10], VGG19 [33], VGG16 [33]. We fine-tuned each model and frozen 80% of the convolutional layers. The training and validation results are as shown in the figure 3 and figure 4.

Figure 3 shows that ACC and LOSS showed almost identical curves for all models. On the other hand, as shown figure 4, Val\_acc value increased and Val\_loss value decreased only in the VGG16 and VGG19 models. Of course, depending on the design of some parameters, it is possible that other models could improve the verification capabilities. However, in our strategy, these models have not been able to obtain sufficient validation capabilities using the Max80 dataset. Table shows the classification accuracy when the verification was performed. We selected VGG16 model because it was slightly more accurate than VGG19 model.

Table 5 shows the classification accuracy using 9339 images of the whole Maling dataset. Because the classification accuracy between VGG16 and VGG19 was very small, we chose both models at this stage.

### 5.2 Investigating optimal undersampling and fine-tuning

We investigated the optimal fine-tuning of selected VGG16 and VGG19 models. We prepared Max80, Max160, Max240 and Max320 dataset for the investigation. First, We trained and validated Max80, Max160, Max240, Max320 using VGG19 frozen 80%. The validation performance results were shown in Figure 5 and Figure 6. Comparing Figure 5



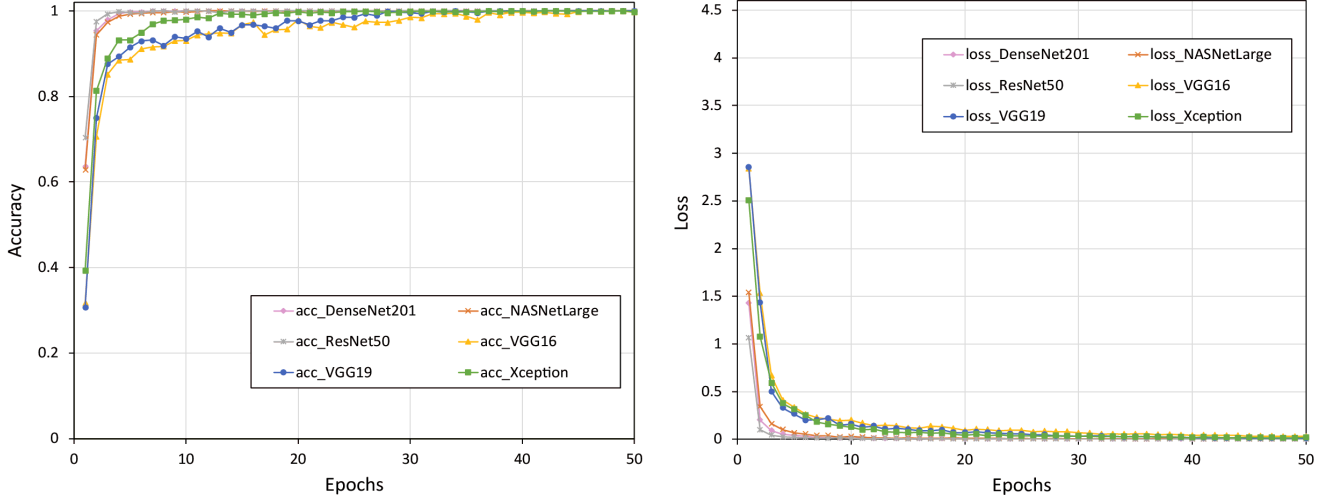


Figure 3: Training performance of each models.

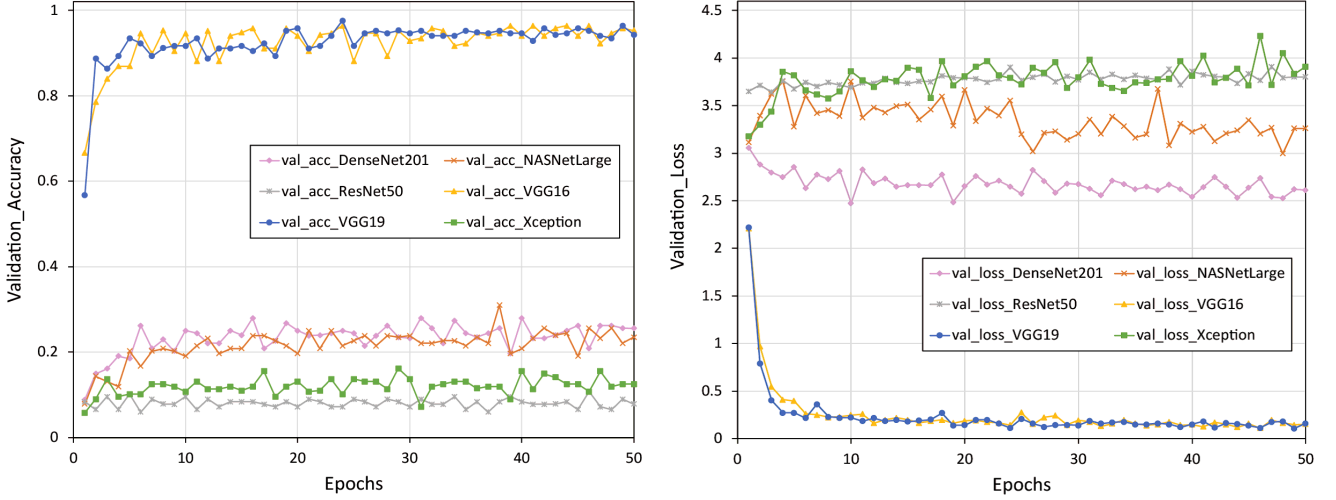


Figure 4: Validation performance of each models.

and Figure 6, the former with a larger number of samples had higher Accuracy and a lower Loss. This result shows that reducing the sample data extremely by undersampling reduces the classification accuracy. Referring to Figure 6 (a), since Max240 and Max320 drew almost the same curve, we conjecture that these numbers would be the optimal value for undersampling. In both Figure 5 (b) and Figure 6 (b), the loss is dropping steadily, which means that there is no tendency for over-fitting.

Next, We trained and validated Max240 and Max320 using frozen 80%, frozen 60%, frozen 40% and frozen 20%, and the top of the results are as shown in Table 6. Figure 7 and Figure 8 show the confusion matrices generated by the VGG19 and VGG16 models. All classes are observed to be able to classify malware without major errors.

### 5.3 Comparison with other reports

We compared the classification accuracy of our chosen model with that of other reports for malware visualization image classification. Table 7 shows that the classification accuracy of our selected model is higher than any previously reported method using Maling dataset and that we can effectively classify the Maling dataset.

## 6 Conclusion

We proposed a strategy to select a Deep learning model that fits the malware visualization images. First, we solved the problem of sample data imbalance between malware families using the undersampling technique. Second, we selected

Table 5: Classification results of 9339 images by the model frozen 80% trained by Max80

Model	Accuracy	Precision(micro)	Precision(macro)	Recall(micro)	Recall(macro)
VGG19	98.83	98.83	97.53	98.83	97.63
VGG16	98.82	98.82	97.55	98.82	97.46

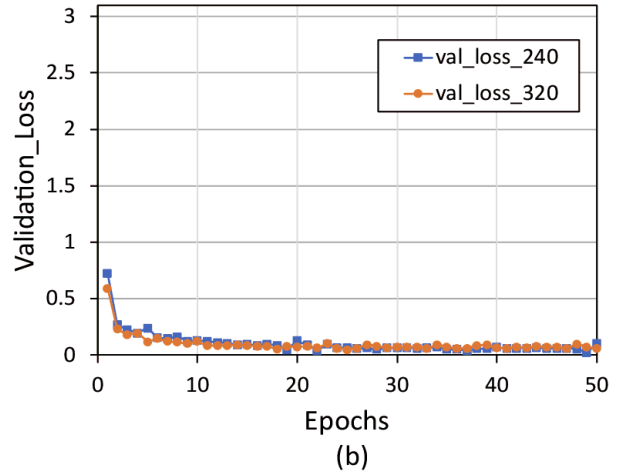
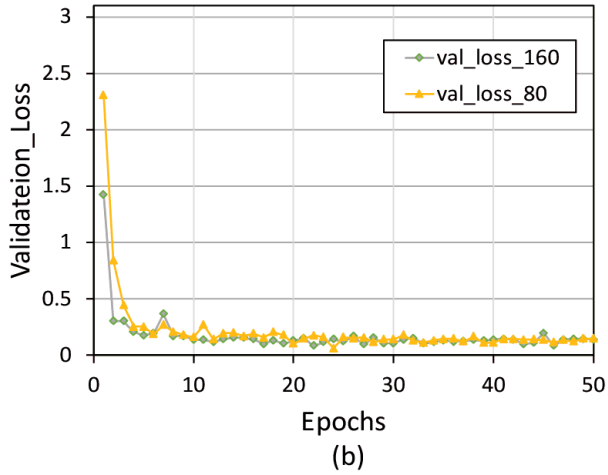
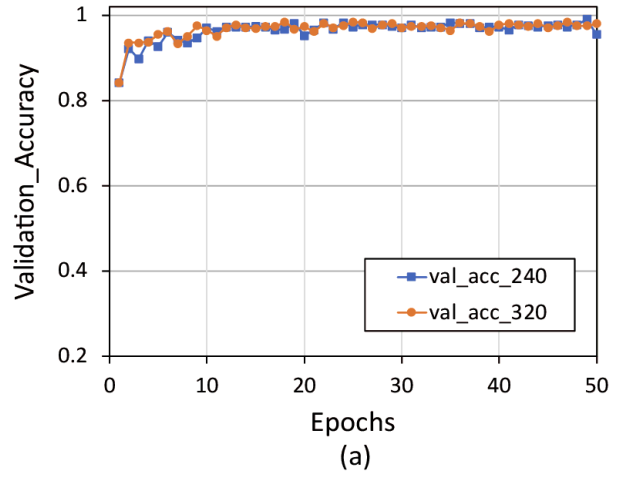
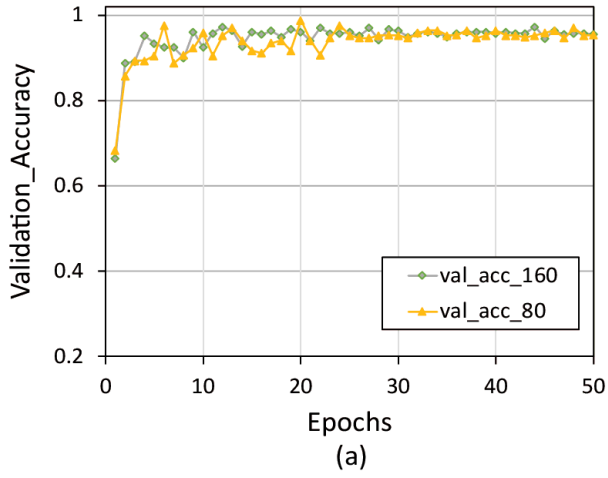


Figure 5: Validation performance of Max80 and Max160.

Figure 6: Validation performance of Max240 and Max320.

Table 6: Classification results of 9339 malware images by VGG16 and VGG19

Model	Accuracy	Precision (micro)	Precision (macro)	Recall (micro)	Recall (macro)
VGG19 Frozen60% trained by Max320	<b><u>99.72</u></b>	99.72	<b><u>99.47</u></b>	99.72	<b><u>99.76</u></b>
VGG16 Frozen40% trained by Max320	99.72	99.72	99.44	99.72	99.74
VGG16 Frozen40% trained by Max240	99.68	99.68	99.31	99.66	99.56
VGG19 Frozen60% trained by Max240	99.65	99.65	99.21	99.65	99.58

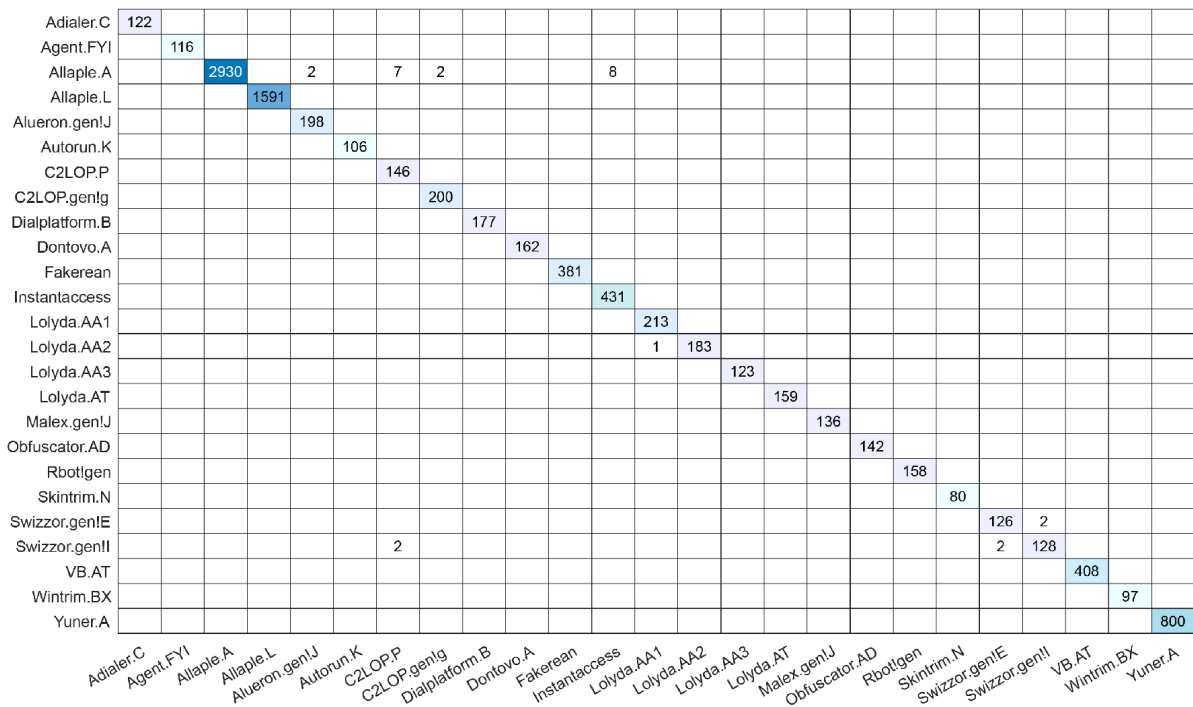


Figure 7: Confusion matrix: VGG19 Frozen60% model trained by Max320 with 99.72% accuracy (**Best**).



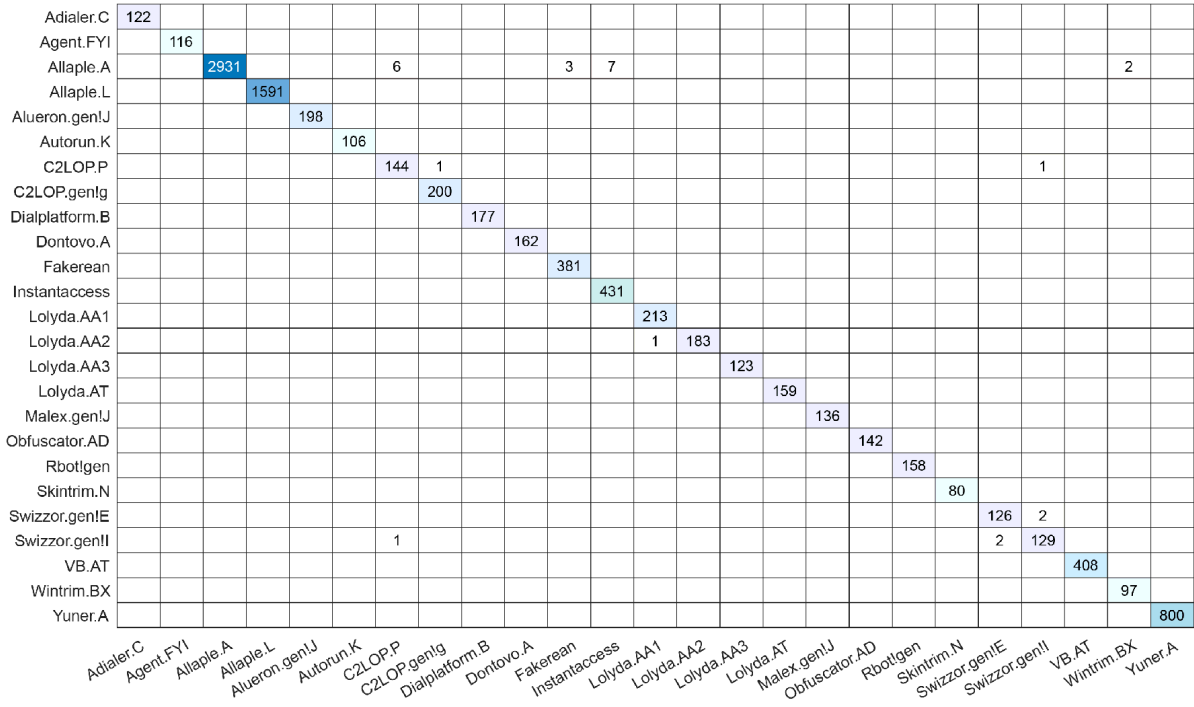


Figure 8: Confusion matrix: VGG16 Frozen40% model trained by Max320 with 99.72% accuracy.

Table 7: Comparison of accuracy for Maling dataset

Methods	Accuracy
CNN [4]	94.50%
GIST + K-nearest neighbors [26]	97.18%
M-CNN (VGG16) [13]	98.52%
ResNet50 [27]	98.62%
Xception [23]	99.03%
VGG19 (ours)	<b>99.72%</b>

a CNN model that fits the malware visualization images using fine-tuning method. Finally, we selected the VGG16 fine-tuning model and achieved high accuracy of classifying malware family. In the future, there is an approach to improve the proposed strategy by preparing a dataset with a larger malware sample.

## References

- [1] Mario Almeida, Stefanos Laskaridis, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. EmBench: Quantifying Performance Variations of Deep Neural Networks across Modern Commodity Devices. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, EMDL '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.
- [2] Sarvam Blog. Maling Dataset, 2014. <https://sarvamblog.blogspot.com/2014/08/>.
- [3] Francois Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

- [4] Zhihua Cui, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Transactions on Industrial Informatics*, 14(7):3187–3196, July 2018.
- [5] Khanh Huu The Dam and Tayssir Touili. STAMAD: A Static MALware Detector. In *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES '19*, New York, NY, USA, 2019. Association for Computing Machinery.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.
- [7] Chollet François. *Deep learning with Python*. Manning Publications Company, 2017.
- [8] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.
- [9] Munawar Hayat, Salman H. Khan, Waqas Zamir, Jianbing Shen, and Ling Shao. Max-margin Class Imbalanced Learning with Gaussian Affinity. *CoRR*, abs/1901.07711, 2019.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [11] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [12] Byungho Jung, Taeguen Kim, and Eul Gyu Im. Malware Classification Using Byte Sequence Information. In *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, RACS '18*, page 143–148, New York, NY, USA, 2018. Association for Computing Machinery.
- [13] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil D. B. Bruce, Yang Wang, and Farkhund Iqbal. Malware Classification with Deep Convolutional Neural Networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, Feb 2018.
- [14] Andrej Karpathy. *Stanford university cs231n: Convolutional neural networks for visual recognition*. 2018. <http://cs231n.stanford.edu/syllabus.html>.
- [15] Jeyeon Kim, Hoon Jo, Moonsoo Ra, and Whoi-Yul Kim. Fine-tuning Approach to NIR Face Recognition. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2337–2341, May 2019.
- [16] Konstantinos Kosmidis and Christos Kalloniatis. Machine Learning and Images for Malware Detection and Classification. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics, PCI 2017*, New York, NY, USA, 2017. Association for Computing Machinery.
- [17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature* 512, pages 436–444, May 2015.
- [18] Richard Li, Min Du, David Johnson, Robert Ricci, Jacobus Van der Merwe, and Eric Eide. Fluorescence: Detecting Kernel-Resident Malware in Clouds. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, pages 367–382, Chaoyang District, Beijing, September 2019. USENIX Association.
- [19] Ying Liu, Yanan Peng, Daxiang Li, Jiulun Fan, and Yun Li. Crime Scene Investigation Image Retrieval with Fusion CNN Features Based on Transfer Learning. In *Proceedings of the 3rd International Conference on Multimedia and Image Processing, ICMIP 2018*, page 68–72, New York, NY, USA, 2018. Association for Computing Machinery.
- [20] Ying Liu, S. Zhang, Daxiang Li, Jiulun Fan, and Weihua Liu. An Effective Tread Pattern Image Classification Algorithm Based on Transfer Learning. In *Proceedings of the 3rd International Conference on Multimedia Systems and Signal Processing, ICMSSP '18*, page 51–55, New York, NY, USA, 2018. Association for Computing Machinery.
- [21] Yingying Liu and Yiwei Wang. A Robust Malware Detection System Using Deep Learning on API Calls. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1456–1460, March 2019.
- [22] Raymond W Lo, Karl N Levitt, and Ronald A Olsson. *MCF: A malicious code filter*, volume 14. Elsevier, 1995.
- [23] Wai Weng Lo, Xu Yang, and Yapeng Wang. An Xception Convolutional Neural Network for Malware Classification with Transfer Learning. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, June 2019.
- [24] Mohammad M. Masud, Latifur Khan, and Bhavani Thuraishingham. *A scalable multi-level feature extraction*

technique to detect malicious executables, volume 10. Springer, 2008.

- [25] Youness Mourtaji, Mohammed Bouhorma, and Daniyal Alghazzawi. Intelligent Framework for Malware Detection with Convolutional Neural Network. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, NISS19, New York, NY, USA, 2019. Association for Computing Machinery.
- [26] Lakshmanan Nataraj, S. Karthikeyan, Grégoire Jacob, and Bangalore S. Manjunath. Malware Images: Visualization and Automatic Classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, New York, NY, USA, 2011. Association for Computing Machinery.
- [27] Edmar Rezende, Guilherme Ruppert, Tiago Carvalho, Fabio Ramos, and Paulo de Geus. Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1011–1014, Dec 2017.
- [28] Matilda. Rhode, Pete. Burnap, and Kevin. Jones. Early-stage malware prediction using recurrent neural networks. *Computers & Security*, 77:578–594, May 2018.
- [29] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. *Automatic analysis of malware behavior using machine learning*, volume 19. IOS Press, 2011.
- [30] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 490–510, Cham, 2018. Springer International Publishing.
- [31] Mohit Sewak, Sanjay K. Sahay, and Hemant Rathore. An Investigation of a Deep Learning Based Malware Detection System. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, New York, NY, USA, 2018. Association for Computing Machinery.
- [32] Raheel Siddiqi. Effectiveness of Transfer Learning and Fine Tuning in Automated Fruit Image Classification. In *Proceedings of the 2019 3rd International Conference on Deep Learning Technologies*, ICDLT 2019, page 91–100, New York, NY, USA, 2019. Association for Computing Machinery.
- [33] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.
- [34] Shiva Darshan S.L and Jaidhar CD. Windows malware detector using convolutional neural network based on visualization images. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019.
- [35] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016.
- [36] Weibin Wang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, Qingqing Chen, Dong Liang, Lanfen Lin, Hongjie Hu, and Qiaowei Zhang. Classification of Focal Liver Lesions Using Deep Learning with Fine-Tuning. In *Proceedings of the 2018 International Conference on Digital Medicine and Image Processing*, DMIP '18, page 56–60, New York, NY, USA, 2018. Association for Computing Machinery.
- [37] Lu Xiaofeng, Zhou Xiao, Jiang Fangshuo, Yi Shengwei, and Sha Jing. *ASSCA: API based sequence and statistics features combined malware detection architecture*, volume 129. Elsevier, 2018.
- [38] Hiromu Yakura, Shinnosuke Shinozaki, Reon Nishimura, Yoshihiro Oyama, and Jun Sakuma. Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, page 127–134, New York, NY, USA, 2018. Association for Computing Machinery.
- [39] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.