Computer Networks 171 (2020) 107138

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture

Danish Vasan[a,b], Mamoun Alazab[c], Sobia Wassan[d], Hamad Naeem[e], Babak Safaei[f], Qin Zheng[a,*]

[a] School of Software Engineering, Tsinghua University, Beijing 100084, China
[b] Department of Computer Science, Isra University Hyderabad, Sindh, Pakistan
[c] College of Engineering, IT and Environment, Charles Darwin University (CDU), Australia
[d] School of Business (Business Administration), Nanjing University, Jiangsu 210000, China
[e] School of Computer Science, Neijiang Normal University, Neijiang, Sichuan, P.R China, 641100
[f] Department of Mechanical Engineering, Eastern Mediterranean University, G. Magosa, TRNC Mersin 10, Turkey

ARTICLE INFO

Keywords:
Cybersecurity
Malware
Image-based malware detection
Convolutional neural network
Transfer learned
Fine-tuned
Deep Learning
Obfuscation
IoT-Android Mobile

ABSTRACT

The volume, type, and sophistication of malware is increasing. Deep convolutional neural networks (CNNs) have lately proven their effectiveness in malware binary detection through image classification. In this paper, we propose a novel classifier to detect variants of malware families and improve malware detection using CNN-based deep learning architecture, called IMCFN (Image-based Malware Classification using Fine-tuned Convolutional Neural Network Architecture). Differing from existing solutions, we propose a new method for multiclass classification problems. Our proposed method converts the raw malware binaries into color images that are used by the fine-tuned CNN architecture to detect and identify malware families. Our method previously trained with the ImageNet dataset (≥10 million) and utilized the data augmentation to handle the imbalance dataset during the fine-tuning process. For evaluations, an extensive experiment was conducted using 2 datasets: Malimg malware dataset (9,435 samples), and IoT- android mobile dataset (14,733 malware and 2,486 benign samples). Empirical evidence has shown that the IMCFN stands out among the deep learning models including other CNN models with an accuracy of 98.82% in Malimg malware dataset and more than 97.35% for IoT-android mobile dataset. Furthermore, it demonstrates that colored malware dataset performed better in terms of accuracy than grayscale malware images. We compared the performance of IMCFN with the three architectures VGG16, ResNet50 and Google's InceptionV3. We found that our method can effectively detect hidden code, obfuscated malware and malware family variants with little run-time. Our method is resilient to straight forward obfuscation technique commonly used by hackers to disguise malware such as encryption and packing.

1. Introduction

The exponential increase in malware attacks has become one of the major threats to Internet security. A recent threat report from Symantec indicated that their 123 million sensors record thousands of malicious threat events per seconds on daily basis [1]. The presence of malware in the internet of things (IoT) and mobile devices increased. According to the latest threat report from Kaspersky Lab in 2019, remove the number of users that encountered Android malware more than tripled to 1.7 million globally. Connectivity between an IoT device and a personal computer is established through a cloud service. The complex IoT hardware and software environments, provides more opportunities for potential adversaries to attacks. According to the cyber financial threat report of 2019, most of the users in China, Brazil, Vietnam, India, Russia, Germany, and the US were attacked by banking malware. Almost 889,452 users of Kaspersky Lab solutions were attacked by banking Trojans in 2018, an increase of 16 percent compared to 2017 when over 767,000 users were hit [2].

Malware (***mal***icious soft***ware***) is one of the most serious security threats on the Internet today. Malware refers to any kind of malicious codes that affect the integrity, confidentiality and the functionality of the digital system [3]. Malware is separated into various classes by their functionalities i.e., Viruses, Worms, Trojans, and Backdoors. These classes further divide into families based of the type of variants. Malware writers deploy many obfuscation methods such as dead-code insertion, subroutine reordering, and code transposition, to create variants of an existing malware family in order to evade detection [4]. The most challenging part of internet security is discovering malware variants. The similarities between many malware variants like Nuwar, Kelihos and Storm suggest they were developed by the same malware coders [3].

* Corresponding author.
E-mail address: qingzh@tsinghua.edu.cn (Q. Zheng).

https://doi.org/10.1016/j.comnet.2020.107138
Received 19 April 2019; Received in revised form 22 January 2020; Accepted 3 February 2020
Available online 16 February 2020

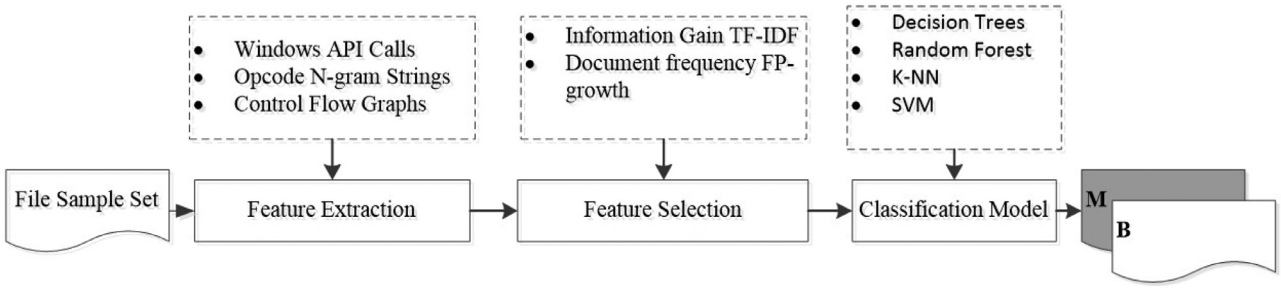


Fig. 1. Malware static analysis based on data mining. The fingerprints use in the static analysis comprises opcode frequency distribution, byte sequence n-grams, and the control flow graph.

According to Symantec, 317 million new malware variants appeared in 2014, and this number is increasing. Traditional antimalware solutions are mostly incapable of dealing with the diversity and volume of malware variants which is evident today [3].

Malware analysis is a fast-growing field demanding a great deal of attention because of technological development progresses in social networks, cloud computing, mobile environment, smart grid, Internet of Things (IoT) and Industrial Internet of Things (IIoT), etc. Most malware detection systems rely on feature vectors, which represent the essential features of malware. These feature vectors separate into two categories: static analysis and dynamic analysis. Static analysis works by disassembling the malicious code, without executing it. As shown in Fig. 1, the fingerprints use in the static analysis comprises opcode frequency distribution, byte sequence n-grams, and the control flow graph. However, this approach is vulnerable to code obfuscation. The other option, is dynamic analysis works by executing the malicious code in a virtual environment, simulator, and sandbox. Monitoring tools like process monitor or capture BAT install before running the malicious code [5]. As indicated by Alazab [6,7] 'static analysis is faster and more effective as compared to dynamic analysis due to its advantages from the information captured relating to structural properties such as sequence of byte "signatures" and anomalies in file content. Dynamic analysis can be effective with runtime information, such as running process or by using control flow graph that could be less prone to obfuscated malware'.

Conventional malware detection methods mainly based on malicious codes feature analysis such as static features and dynamic features. These features also apply powerful machine learning based malware detection methods to uncover malicious code variants. However, even these methods fail to detect malware variants. Rather than focus on the non-visible features for malware detection, Natraj et al. [8] proposed a new malware detection approach using visible features. They converted the structure of packed binary executable into two-dimensional grayscale images. Later these visible features used for malware detection. The results indicated that binary texture analysis was more accurate and less time-consuming.

The main challenge however that most of the visualization approaches [9,10] compute texture similarity of gray scale image. These approaches resolve code obfuscation issues, but they need high computational cost for complex texture feature extraction of malware images i.e., GIST, SURF, DSIFT, LBP, and GLCM. Apart this, these feature extraction techniques are less efficient when applied to large datasets. The malware is continuously creating, updating and changing. Consequently, how to reduce the cost of feature extraction, extract appropriate information from raw binary data and improve the accuracy of malware detection are our key inspirations.

The main objective of the paper is to cope with polymorphic transformations and metamorphic obfuscations of malware for achieving effective and efficient solutions to address the malware detection and identifying variants. Furthermore, the number of malicious code variants among various malware families varies significantly. The greatest challenge is to introduce a comprehensive malware detection model that can deal with massive amounts of malicious code variants. Some of the existing methods mentioned that data augmentation on data pre-processing level was the solution of data imbalance, but they failed to represent entire variations of malware. Therefore, to introduce a novel automatic features extraction and efficient malware family classification approach, resolve the data imbalance issue using augmentation technique and low running time overhead are our main aims.

Overall, the main contributions of the current research work are:

- Providing a critical review of related work on image-based malware classification.
- Developing a novel hybrid deep learning model (IMCFN) which combines the visualization and fine-tuned CNN architecture for malware detection and classification that are computationally cost-effective and scalable, with relatively low run-time overhead.
- Our model does not demand any feature engineering or domain expert knowledge, like: reverse engineering, binary disassembly, assembly language. The performance of our model exceeded feature extraction-based approaches.
- Implementing wise fine-tuning scheme to malware fingerprint images via back-propagation method. Data augmentation technique is utilized to optimize the performance of IMCFN algorithm and to handle the imbalance dataset. Only fine-tuned layers including FC2, FC1 and Block5 used for less computational cost and faster in classification which fulfill the demand of most of the practical applications. We demonstrate its effectiveness and its low overhead.
- Implementing image normalization, to identify possible obfuscated or packed malware of the known malicious code within the normalized program.
- Conducting an in-depth analysis by performing of various classical machine learning and state-of-the-art deep learning architectures on large datasets with a purpose to evaluate our proposed architectures in terms of their efficacy in dealing with large datasets of new malware variants.
- Evaluating our model against obfuscated malware, using 25 malware families of the Malimg dataset, to predict the resilience of the obfuscated malware attack.

The remainder of this article is organized as the following: Section II presents related works of malware identification. Section III presents a convolutional neural network that is trained on more than a million images from the ImageNet database, called VGG16 architecture, ResNet50 architecture, and Google's Inception V3. The proposed hybrid model IMCFN adopted for this study is presented in Section IV. Section V presents discussion on obfuscation attacks. The results of the performance evaluation of our proposed model indicating high classification accuracy achieved using machine learning and deep learning architectures are presented in Section VI. Finally, we provide our conclusions, highlighting the limitations of the study and future research work in Section VII.

2. Related work

For efficient malware variant detection in Windows operating system, several researchers worked on malware visualization to attain maximum classification accuracy and reduce time overhead. This section presents the related works regarding visualization, including malware identification based on statistical similarity measure, malware identification based on machine learning and malware identification based on deep learning.

2.1. Malware identification based on statistical similarity measure

Currently, many techniques can visualize and operate on binary data such as static analysis and dynamic analysis. In the dynamic visual analysis of malware, various methods such as tree map, thread graph, and malware behavior image have been proposed. For example, Trinius et al. [11] presented dynamic malware detection model to analyze various samples by collecting information about API calls and operations of the performed actions in a sandbox. Later, information visualized by using malware tree map and thread graph method. Additionally, they also described dynamic visual analysis by using thread graph technique. Malware thread graph displays the sequential order of the sections and the transition between the regions that are represented by an API call. Further, their malicious behavior detected through statistical clustering method. Due to section-based representation of malware behavior, their model remained unsuccessful to differentiate similar malware families.

Based on their work, Syed et al. [12] proposed a malware variant detection method based on the behavior of malware sample; the critical point was collecting information on malware behavior sequence. In their method, the behavior of malware sample gathered by executing them in a virtual environment. After that, the gathered behavior i.e., API call sequence and operating system resources is transformed to a color image by using color map. The similarity between these images is calculated by running statistical method. Their experiment showed that their method achieves detection rates ranging from 95.92%–98% while taking the 1102 samples from 12 different malware families. Dynamic analysis is useful for malware binary transformation detection, while the use of the virtual environment is very time intensive.

Keeping an eye on above limitations in the dynamic visual analysis, a few researchers tried to detect malware by combining static and dynamic visual analysis. For instance, KyoungSoo Han et al. [13] proposed a hybrid malware variant detection that statically converted the Opcode sequences into image matrices. To reduce computation time, the model extracted Opcode sequences of functions and API calls only. Also, their method dynamically extracted execution traces to avoid binary transformation strategies. After that, these image matrices used to classify malware families both statically and dynamically with an accuracy of 98.96% and 97.32%, respectively. Their method is suitable for small-scale malware detection without adversely affecting the binary transformation, while they use excessive resources for static and dynamic visual analysis to target the behavior of malware. It might lead to more computational overhead for large-scale malware detection. In recent research, Yao et al. [14] highlighted that large-scale malware detection through dynamic analysis is a significant challenge.

Another practical approach that does not require execution of malware sample for feature extraction is a static visual analysis of malware. Currently, many methods of static visual analysis such as image similarity matrix and graph similarity have been proposed to detect malware variants. For example, Jae et al. [15] first introduced malware variant classification through Image similarity matrix technique. Their method used malware binary information such as Opcode instructions to generate RGB colored pixels on image matrices. By using selective area matching, they calculated the similarity between image matrices. Their experimental results showed that their method achieved 98% similarity rate while taking 50 samples from 10 different malware families. Despite this, the similarity computation for large-scale malware analysis by using selective area matching is quite slow.

EulGyuim et al. [16] put forward a graph-based similarity technique for malware variant classification. Firstly, their method converted section information of malware binary into gray scale images, and then entropy graphs generated from these gray scale images. They used Strelkov histogram similarity measuring method for graph similarity. Their method achieved 97.9% similarity rate while taking 1000 samples from 50 different malware families. Their method is helpful to process a large number of unpacked malware binary samples. However, in limitation section of their work, they clearly mentioned that the similarity calculation for packed malware binary samples by using entropy graphs is a very difficult task.

2.2. Malware identification based on machine learning

Machine learning based methods use datasets of malware and benign files and extract features of these files to train different machine learning classifiers. For example, Kesav Kancherla et al. [17,18] proved that machine learning classification was more capable for binary transformation detection. They first converted each malware executable into the image and then extracted their texture features. They calculated the similarity between these features using Support Vector Machine (SVM). Ban Xiaofang et al. [9] proposed a malware classification method that extracts local features through SURF descriptor (Speeded up robust feature) for each malware image and then does fast fingerprint matching with LSH. Their experimental results showed 85% detection accuracy while taking 8410 samples from 25 malware families. Their method more focuses on capturing local visual characteristics of significant interest points with the distributions of intensity gradients.

To reduce false positive ratio in large-scale malware detection, some researchers try to pay more attention to global feature extraction of malware images. Nataraj et al. [19] proposed that an image texture analysis is more suitable to classify the families of malware variants as compared to other existing malware analysis techniques. Firstly, they transformed malware binary information into images and then extracted GIST features for classification. They used the K-nearest neighbor classification technique. Their technique attained 97% classification accuracy while taking 9339 samples from 25 different malware families. It assumed that a set of texture features could be enough to present entire content of malware images.

Aziz et al. [10] extracted GIST based texture features to identify new malware and their variants, and then fed forward to ANN for classification. Their technique obtained 98% classification accuracy while taking only 1710 samples from 8 malware families. While for large- scale analysis, their method achieved 96.3% classification accuracy while taking only 3131 samples from 24 malware families. Barath et al. [20] introduced a malware detection technique that first extracted global features using PCA, and then applied nearest neighbor classification. Their method indicated 96% classification accuracy while taking only 10,000 samples from 8 malware families. Hashem et al. [18] first extracted texture features using LBP and then applied to machine learning classifiers for malware variant detection. Their model achieved maximum results for malware variant identification.

Instead of particular feature extraction, Hamad et al. [21] designed a malware classification system (MICS) that first extracted hybrid features of malware and then used SVM for classification. Their method achieved 97.4% classification accuracy while taking only 9339 samples from 25 malware families. While for small-scale analysis, their method achieved 99.6% classification accuracy by taking only 5116 samples from 10 malware families. These conventional approaches needed high computational time for texture feature analysis. To resolve this issue, we used deep learning to identify images efficiently.

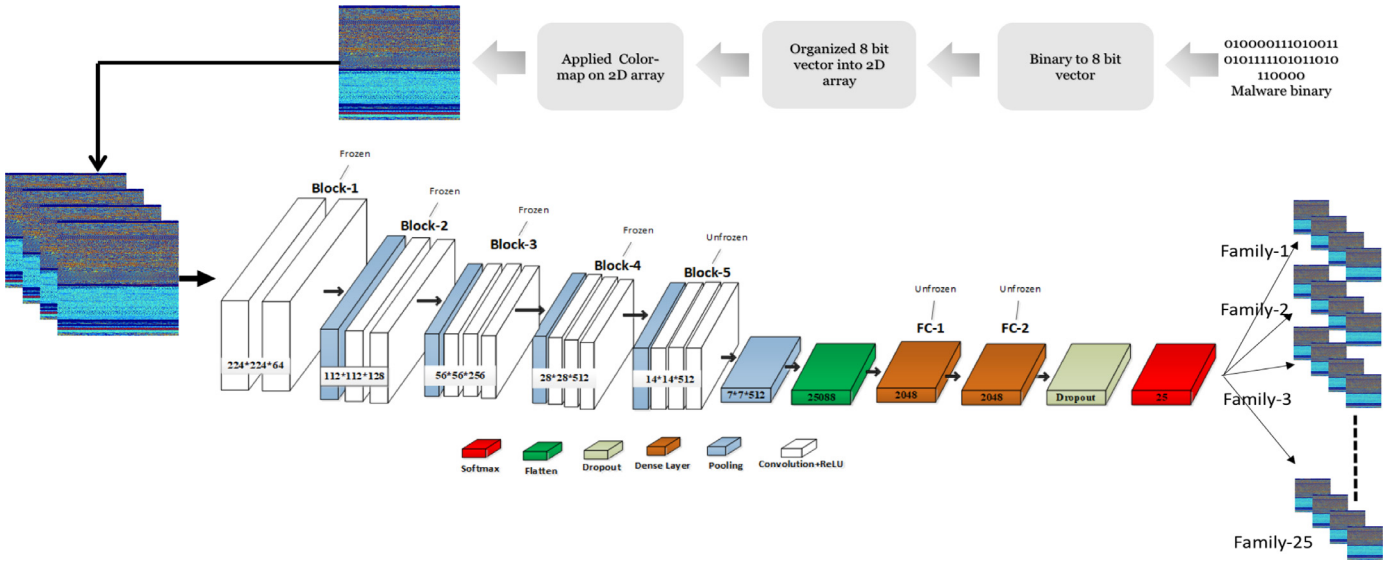


Fig. 2. Basic procedure of IMCFN including two parts: malware image generation and CNN fine-tuning.

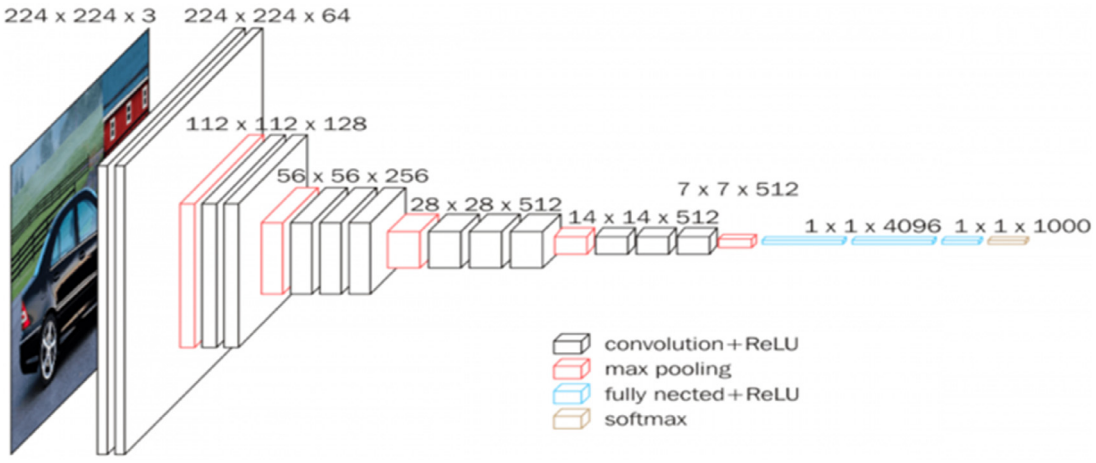


Fig. 3. VGG16 Architecture.

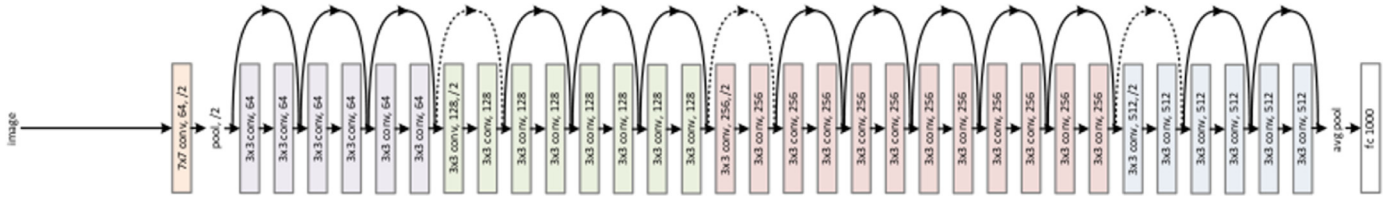


Fig. 4. ResNet50 Architecture.

2.3. Malware identification based on deep learning

Deep learning uses data sets of malware and benign files and learns the characteristics of these files. Deep learning has applied in many fields such as speech recognition and image recognition as a powerful artificial intelligence tool [22-24] For example, Sang et al. [25] designed a malware categorization technique called MCSC which combined malware visualization with deep learning techniques. They first extracted the orders of Opcode from malware executable and then encoded them with SimHash. They took SimHash values as pixels and then converted them to grayscale images. Finally, they applied CNN to train images and identified malware families. Their method achieved high classification accuracy for small-scale application environments, while it could not be used for faster malware detection in large-scale application environments. Quan et al. [26] proposed a deep learning model without using reverse engineering. Their model achieved 98.2% classification accuracy while taking only 10,860 samples from 9 malware families.

Jinpei et al. [27] proposed a deep neural network which took CNN and LSTM networks to automatically learning features from the malicious files. It greatly reduced the cost of artificial features engineering. Their model obtained 99.36% classification accuracy while taking only 10,860 samples from 9 malware families. Mahmoud et al. [28] proposed a CNN-based architecture to classify malware samples. They performed experiments on most challenging malware dataset named Malimg. Their architecture achieved 98.52% accuracy while taking 9339 samples from 25 malware families. They randomly selected only 10% samples in a

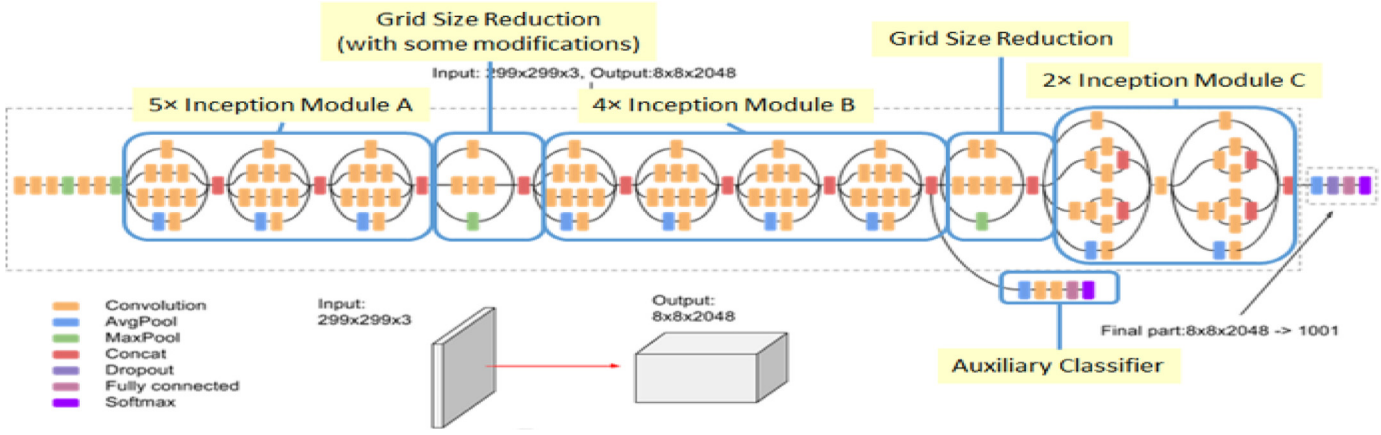


Fig. 5. InceptionV3 Architecture.

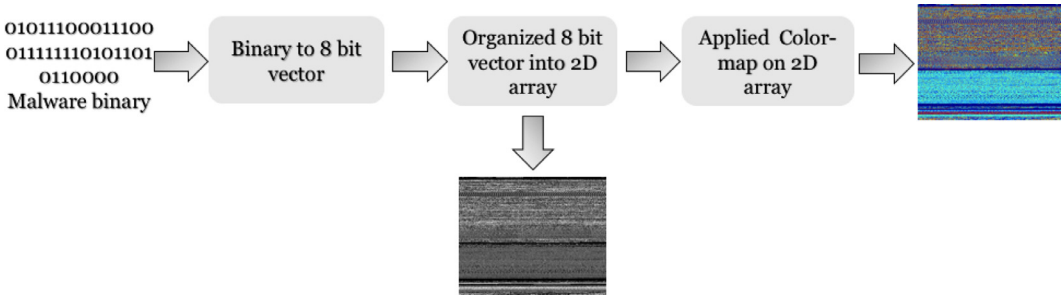


Fig. 6. The detail process of raw malware binary visualization into image.

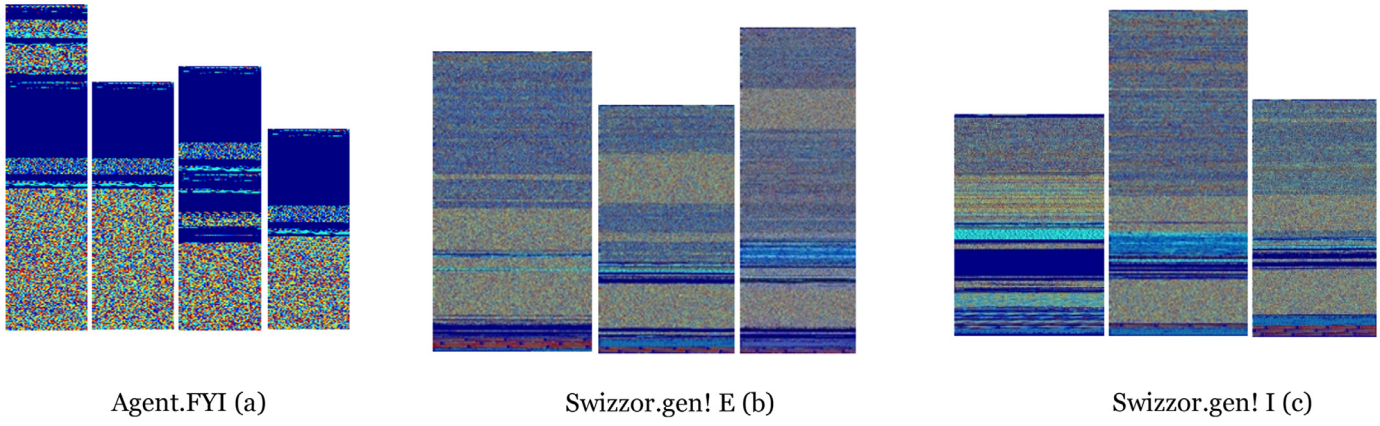


Fig. 7. Illustration of malware images.

family for testing. While our technique selected 30% samples in a family for testing and achieved more than 98% accuracy from 25 malware families.

Rajesh et al. [29] presented a CNN-based deep learning architecture for malware detection. They performed experiments on 25 malware families of the Malimg dataset. Their model obtained 98% accuracy for 9339 samples. They randomly selected only 10% samples in a family for testing. The models described in [28-30] were unable to deal imbalance datasets.

Vinayakumar et al. in [31] and [32] used various deep learning methods including CNN architecture for intrusion detection of both network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS), while the reported accuracy of more than 98%. The proposed system does not give enough information on the structure and characteristics of the malware, as well as, did not consider the overhead time.

Table 1
Image Width for Various File Sizes.

File Size	Image Width	File Size	Image Width
< 10KB	32	100 KB ~200KB	384
10 KB ~30KB	64	200 KB ~500KB	512
30 KB ~60KB	128	500 KB ~1000KB	768
60 KB ~100KB	256	> 1000KB	1024

Venkatraman et al. [33] presented the use of image-based techniques for detecting suspicious behavior of systems, and proposed the application of hybrid image-based approaches with CNN-based deep learning architectures for an effective malware classification. They proposed two CNN-based models: Unidirectional GRU (UniGRU) and Bidirectional GRU (BiGRU) models, also measured and compared the performance to other existing CNN architectures like: Unidirectional LSTM (UniL-

Table 2
Fine-tuning statistics.

Architecture	Total Parameters	Trainable Parameters	Non-Training Parameters
IMCFN	134,362,969	126,727,705	7635,264
VGG16	138,357,544	138,357,544	0
ResNet50	23,638,937	23,585,817	53,120
InceptionV3	21,802,784	21,768,352	34,432

Table 3
Augmentation Settings.

Method	Settings	Method	Settings
rescale	1/255	*shear range*	0.0
width shift	0.0	*zoom range*	0.0
height shift	0.0	*horizontal flip*	False
rotation range	0.0	*fill mode*	None

Table 4
Malimg Malware Dataset.

No.	Family	Family Name	No. of Samples
1	Worm	Allaple.L	1591
2	Worm	Allaple.A	2949
3	Worm	Yuner.A	800
4	PWS	Lolyda.AA 1	213
5	PWS	Lolyda.AA 2	184
6	PWS	Lolyda.AA 3	123
7	Trojan	C2Lop.P	146
8	Trojan	C2Lop.gen!G	200
9	Dialer	Instantaccess	431
10	Trojan Downloader	Swizzor.gen!I	132
11	Trojan Downloader	Swizzor.gen!E	128
12	Worm	VB.AT	408
13	Rogue	Fakerean	381
14	Trojan	Alueron.gen!J	198
15	Trojan	Malex.gen!J	136
16	PWS	Lolyda.AT	159
17	Dialer	Adialer.C	125
18	Trojan Downloader	Wintrim.BX	97
19	Dialer	Dialplatform.B	177
20	Trojan Downloader	Dontovo.A	162
21	Trojan Downloader	Obfuscator.AD	142
22	Backdoor	Agent.FYI	116
23	Worm:AutoIT	Autorun.K	106
24	Backdoor	Rbot!gen	158
25	Trojan	Skintrim.N	80

STM), and Bidirectional LSTM (BiLSTM). They performed experiments on two publicly available datasets: i) Microsoft Malware Classification Challenge (BIG, 2015) dataset and ii) Malimg dataset. Their model obtained ~96% accuracy on average, however, the model did not consider the overhead time.

To reduce misclassification risk in malware detection, some researchers consider the use of data balancing techniques. For example, Songqing et al. [34] prepared a weighted softmax loss function to balance the imbalance malware dataset. Zhihua et al. [35] proposed a CNN based malware variant detection model. Also, they used a BAT algorithm for data equilibrium approach to resolve the imbalance data issue. Their technique obtained 94.5% accuracy while taking 9339 samples from 25 malware families. To handle the multiclass malware families imbalanced issue, cost sensitive approach were used [31-33].

Different from previous research work, this work has advanced further from a previous work that proposed a hybrid deep learning-based framework for malware detection. We proposed a novel and unified method for multi-class malware classification. Our algorithm converts the raw malware binary into both grayscale and color images and utilize the fine-tuned CNN architecture previously trained with ImageNet dataset (≥10 million). Furthermore, during the fine-tuning data aug-

Table 5
IoT- Android Mobile Dataset.

Family	Samples
Malware	14,733
Benign	2486

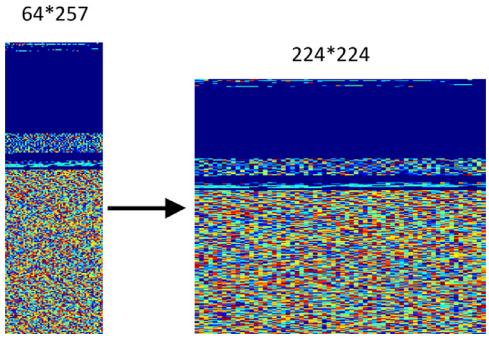


Fig. 8. Reshape the malware image to a fixed size square image.

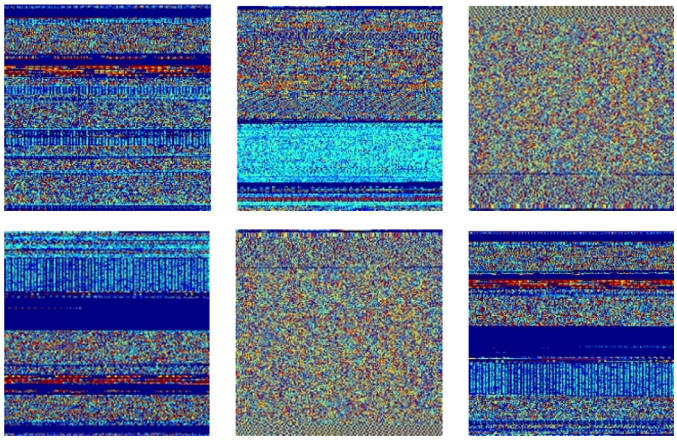


Fig. 9. Sample Images generated by data augmentation for 25 malware families.

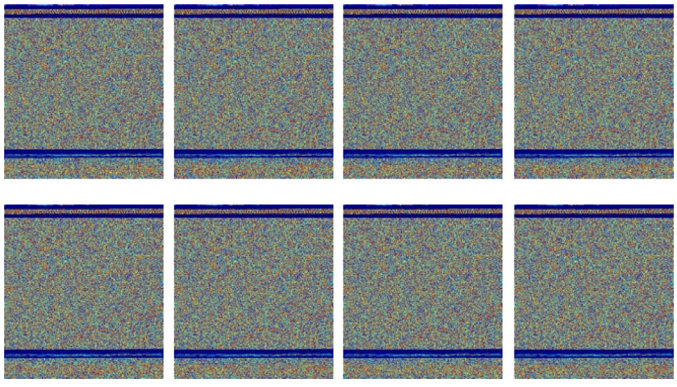


Fig. 10. shows Obfuscated malware samples from ObfuscatorAD family of Malimg dataset.

Table 6
A Comparative Summary of IMCFN algorithm and No-Aug algorithm.

Dataset	IMCFN				No-Aug			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Malimg (Color)	98.82	98.85	98.81	98.75	98.09	98.27	98.03	97.81
Malimg (Grayscale)	98.27	98.25	98.19	98.20	97.85	98.02	97.83	97.90

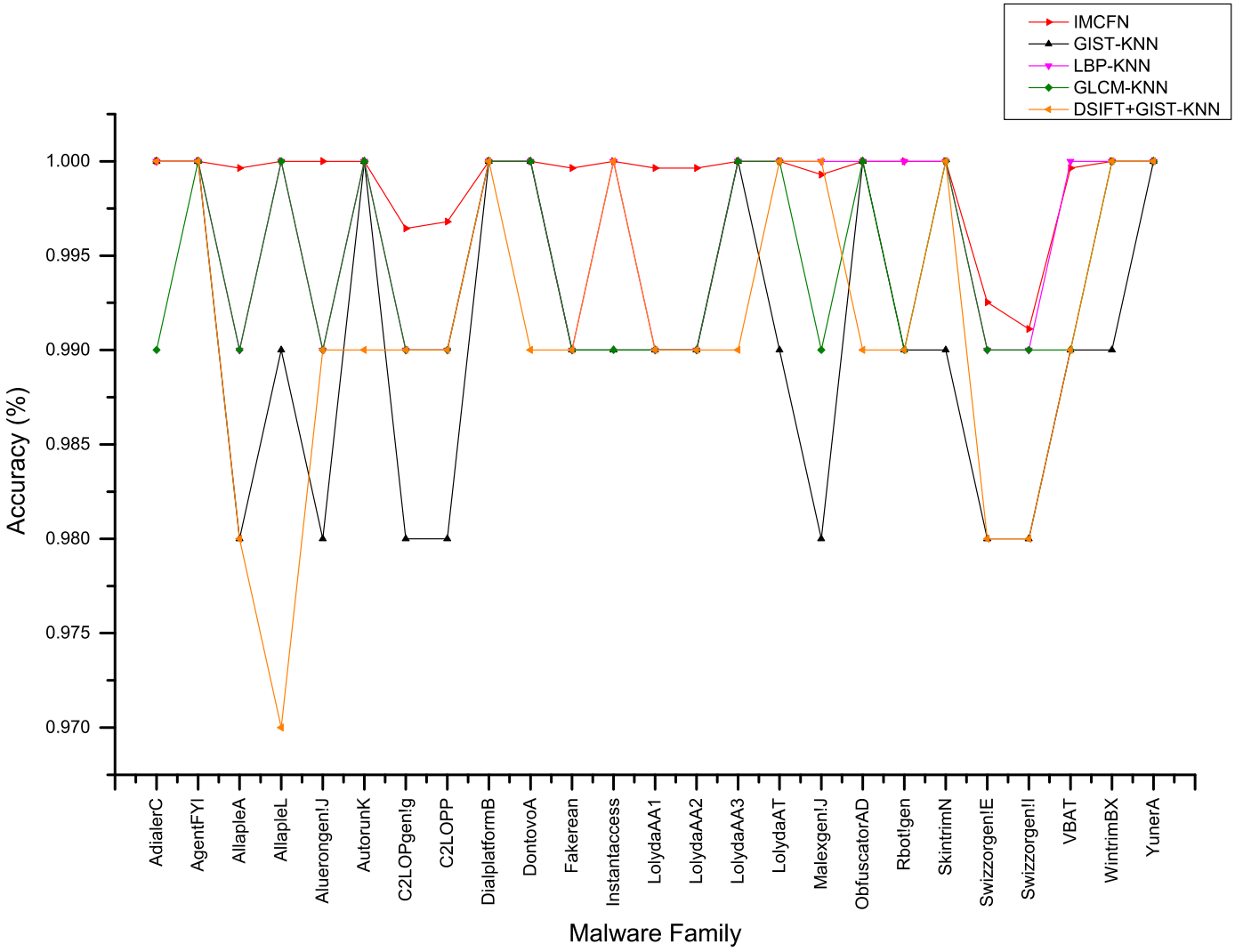


Fig. 11. Accuracy of different malware classification algorithms including IMCFN, GIST-KNN, LBP-KNN, GLCM-KNN, DSIFT-KNN, DSIFT+GIST-KNN in 25 malware families.

mentation method is used to improve the performance of IMCFN algorithm. Our method is a self-learning system capable of identifying malware family variants, and detecting new malware. More importantly, during the fine-tuning, data augmentation method is adopted in our system to improve the performance and the accuracy to distinguish malicious malware binaries and variants. Our model composed of four interconnected different subsystems: i) Malware Binary Visualization, ii) CNN fine-tuning, iii) image normalization, iv) data augmentation, and v) employee CNN-based deep learning architecture. Our experimental considered both overall accuracy and run-time overhead.

3. VGG16, resnet50 and inception-v3

Recently, security researchers started transforming image classification problem to malware classification problem [9,31,36]. CNN based on the VGG16 and Inception-v3 architectures started to be used in the malware classification and intrusion detection research.

VGG16 is a CNN-based deep learning architecture that is trained on (≥14 million) images from the ImageNet database [37]. The ImageNet project runs an annual software contest and designed for use in visual object recognition software research. Our proposed model, adopt a pre-trained VGG-16 model [23] from VGG family (Visual Geometry Group at University of Oxford), and retrain it to achieve a promising result on the malware images classification.

VGG16 network architecture [23] is shown in Fig. 3. The input to layer is of fixed size 224×224 RGB image. Then the image is passed through a stack of convolutional layers, where the filters size used in convolutional layers was 3 × 3, it also utilizes 1 × 1 filter. The convolution stride is fixed to 1 pixel with 1 padding to guarantee the same spatial dimension for each activation map as the previous layer. VGG16

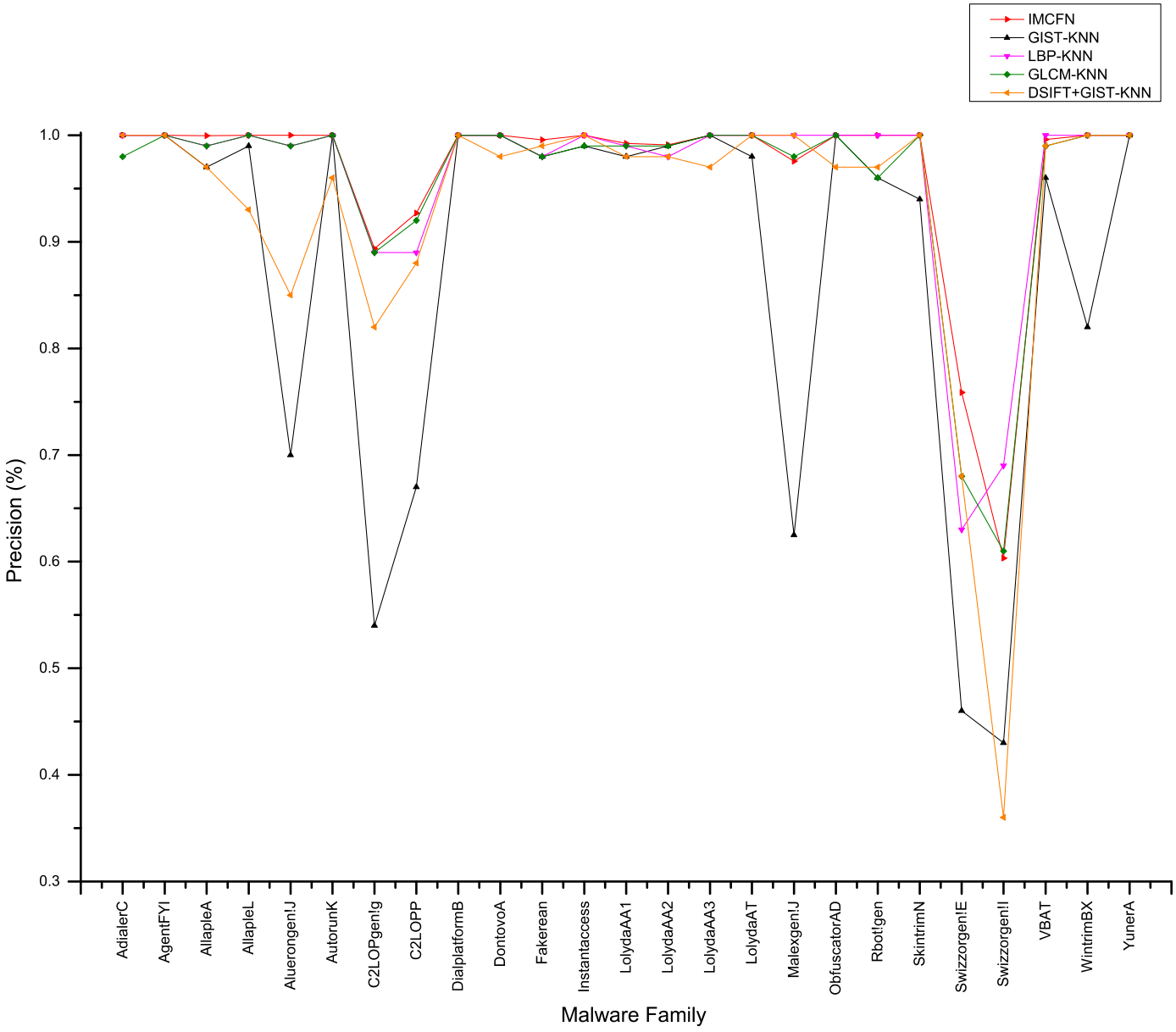


Fig. 12. Precision of different malware classification algorithms including IMCFN, GIST-KNN, LBP-KNN, GLCM-KNN, DSIFT-KNN, DSIFT+GIST-KNN in 25 malware families.

has a standard neural network architecture of connected layers and contains:

- 16 layers in total,
- 5 convolutional layers,
- 5 max pooling layers,
- 3 fully-connected layers
- output layer (softmax)

To accelerate training, all hidden layers are equipped with the rectified linear unit (ReLU), and with the max-pooling layers 2×2 kernel filter with no padding and 2 strides was used. For the output layer, a 1000-way softmax classifier. In this research, top layers frozen and its last three layers retrained to classify malware.

Residual networks (ResNet) [38] are deep convolutional networks and the fundamental idea behind these networks is to skip convolutional layer blocks with shortcut connections. The basic blocks, called bottleneck blocks, obey two design rules: (i) for similar sizes of output feature maps, the same filter numbers were applied to layers, and (ii) if feature map sizes were halved, filter number was doubled. Down-sampling was directly conducted by convolutional layers with 2 strides and batch normalization was carried out immediately after each convolution and before the activation of ReLU. When output and input had similar dimensions, identity shortcut was applied. By the increase of dimensions, projection shortcut was applied for dimension matching via 1×1 convolutions. In both cases, when shortcuts passed through feature maps with two different sizes, they were performed with 2 strides. Networks ended with 1000 fully-connected layers with softmax activation. The total weighted layer number was 50, containing 23,534,592 trainable parameters. The original ResNet- 50 architecture is shown in Fig. 4.

Google's Inception-v3 is another CNN-based deep learning architecture that is trained on (≥1 million) images from the ImageNet Large Visual Recognition Challenge (LVRC) where it was a first runner up. Google's Inception v3 model outperformed other architectures like: VGG-Net, GoogLeNet, PreLU and BN-Inception in error rate [39]. Inception-v3 is based on the original paper [40] by Szegedy, et al. Similar to VGG16 the main idea is to improve CNNs performances without

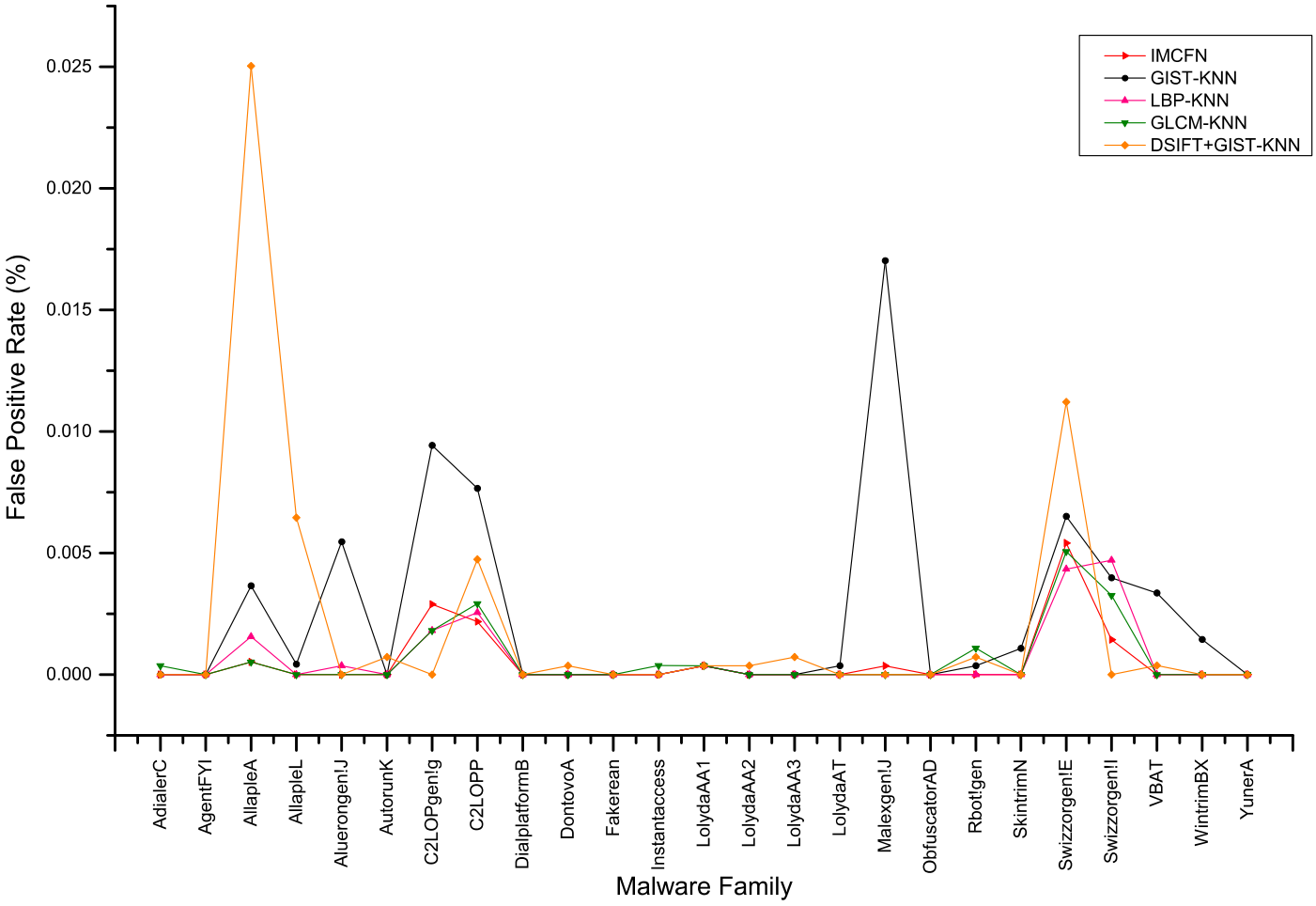


Fig. 13. False Positive Rate of different malware classification algorithms including IMCFN, GIST-KNN, LBP-KNN, GLCM-KNN, DSIFT-KNN, DSIFT+GIST-KNN in 25 malware families.

the need to getting deeper with the number of convolutional layers, where a layer will have different kernel sizes and will learn different sets of features in a single iteration. Also, it has proven to be computationally efficient and effective in image classification.

Inception-v3 network architecture [41] is shown in Fig. 5. The model contains both symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, Concat, dropouts, and fully connected layers. Batch-normalization is used extensively throughout the model and applied to activation inputs, and loss is computed via Softmax.

4. IMCFN algorithm

IMCFN algorithm is mainly divided into two parts: malware image generation and CNN fine-tuning via backpropagation technique. During the fine-tuning, we utilized data augmentation technique to improve the performance of IMCFN algorithm. The basic procedure of our IMCFN is depicted is Fig.2.

4.1. Image representation of malware

To extract image-based features, a malware binary need to be transformed to an image. In this paper, we offer a simple yet effective method which visualization raw malware binary into color image. Our method does not require any feature engineering and domain expert knowledge. The detail process is described in Fig. 6. Our method read a given malicious binary as a vector of 8-bit unsigned integers. We then organized a vector into a 2D array. Finally, we applied a color-map on the 2D array and visualized malware binary. Fig. 7, shows visualized raw malware binaries into color images based on proposed method. The height of the malware image is allowed to vary depending on the file size, and the width of the malware image is fixed. Based on empirical observations. Table 1 presents the image widths for different file sizes.

As shows in Fig 7, its observed that malware fingerprint images belonging to the same family appeared similar in layout and style, which similar result to [35]. Correspondingly, it also noticed that malware images belonging to dissimilar families displayed differently in form. For example, in Fig. 7(a) showed four variants of the Agent.FYI malware family. The image ratio of each sample was different, but they still had visual similarities. Also, when families were dissimilar, images displayed their transformations. Compared with the Swizzor.gen! E family in Fig. 7(b), the Swizzor.gen! I family in Fig. 7(c) had different stripes and visualization.

Inspired by the visual resemblance of malware images, we turned malware classification problem into image classification.

4.2. Fine-Tuning

CNN was customized to our problem by adding a dropout layer and a fully-connected layer containing 25 classes (25 malware families) instead of final fully connected layer (intended for 1000 classes). We only fined the later layers including FC1, FC2 and block5 of the customized CNN architecture. Initial weights of pretrained CNN of natural images

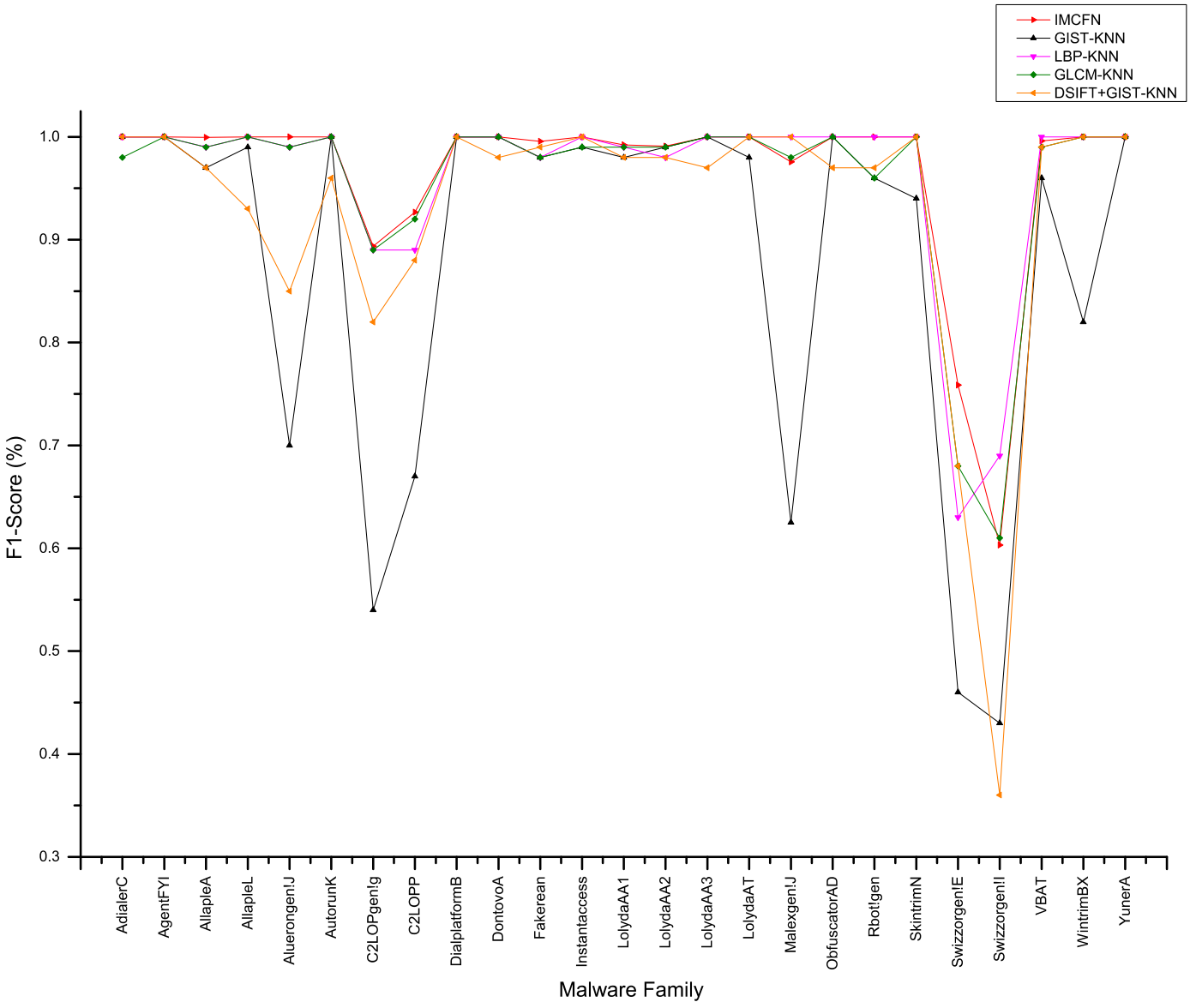


Fig. 14. F1-Score of different malware classification algorithms including IMCFN, GIST-KNN, LBP-KNN, GLCM-KNN, DSIFT-KNN, DSIFT+GIST-KNN in 25 malware families.

were used and after that, fine-tuned technique was employed to optimize via back-propagation.

Assume that X is training dataset in n malware images. Fine-tuned is an iterative procedure optimizing w filter weights, reducing error rate.

$$L(w, X) = \frac{1}{n}\sum_{i=1}^{n} l\left(f\left(x_i, w\right), \hat{c}_i\right) \tag{1}$$

Where $f(x_i, w)$ is CNN function predicting the class c_i of x_i by assuming w, $\hat{c}_i$ is the true class of the i^{th} image, x_i is the i^{th} image of X, and $l(c_i, \hat{c}_i)$ is a penalty function for the prediction of c_i instead of $\hat{c}_i$. $l$was logistic loss function.

In our deep learning model (IMCFN), mini-batch stochastic gradient descent was employed for obtaining optimal w. Assume that $B \subset X$ is a subset of b images where B is a mini-batch of X with size b. Therefore, the difference between iteration and epoch of the term is presented. Repetition is a training pass over all B elements. An epoch, on the other hand, is a training pass (weight update) employing all training samples in X. At each epoch, a random set of separate mini-batches was produced to cover all X elements. For each epoch, mini-batches were iterated and, in each iteration, the weights of CNN were updated. Updated weights w_{t+1} were obtained based on loss L gradient when they were employed for mini-batch B with present weights w_t

$$w_{t+1} = w_t + \eta\left[\alpha\Delta w_t - \frac{\partial L\left(w_t, B\right)}{\partial w_t} - \lambda w_t\right] \tag{2}$$

where $\Delta w_t = w_t - w_{t-1}$ represent the weight updated in the previous iteration and η is learning rate which controls weight update sizes. Momentum coefficient α reduces weight fluctuation variations over successive iterations through the addition of a proportion of the earlier update to the present one. This accelerates learning process while instantaneously leveling the update of weights. Weight decay λ reduces weights for obtaining lowest optimum weights.

All models needed memory for storing filter weights w and therefore, batch size b depended on the memory capacity of training hardware. Batch size was set at $b = 32$ and learning rate was $\eta = 5\ x\ 10^{-6}$, which is the uniform learning rate that made it possible for fine-tuning procedure to efficiently learn filter weights in different CNN models. We experimentally obtained this value through monitoring validation errors during fine-tuning via different learning rates. Learning rate was

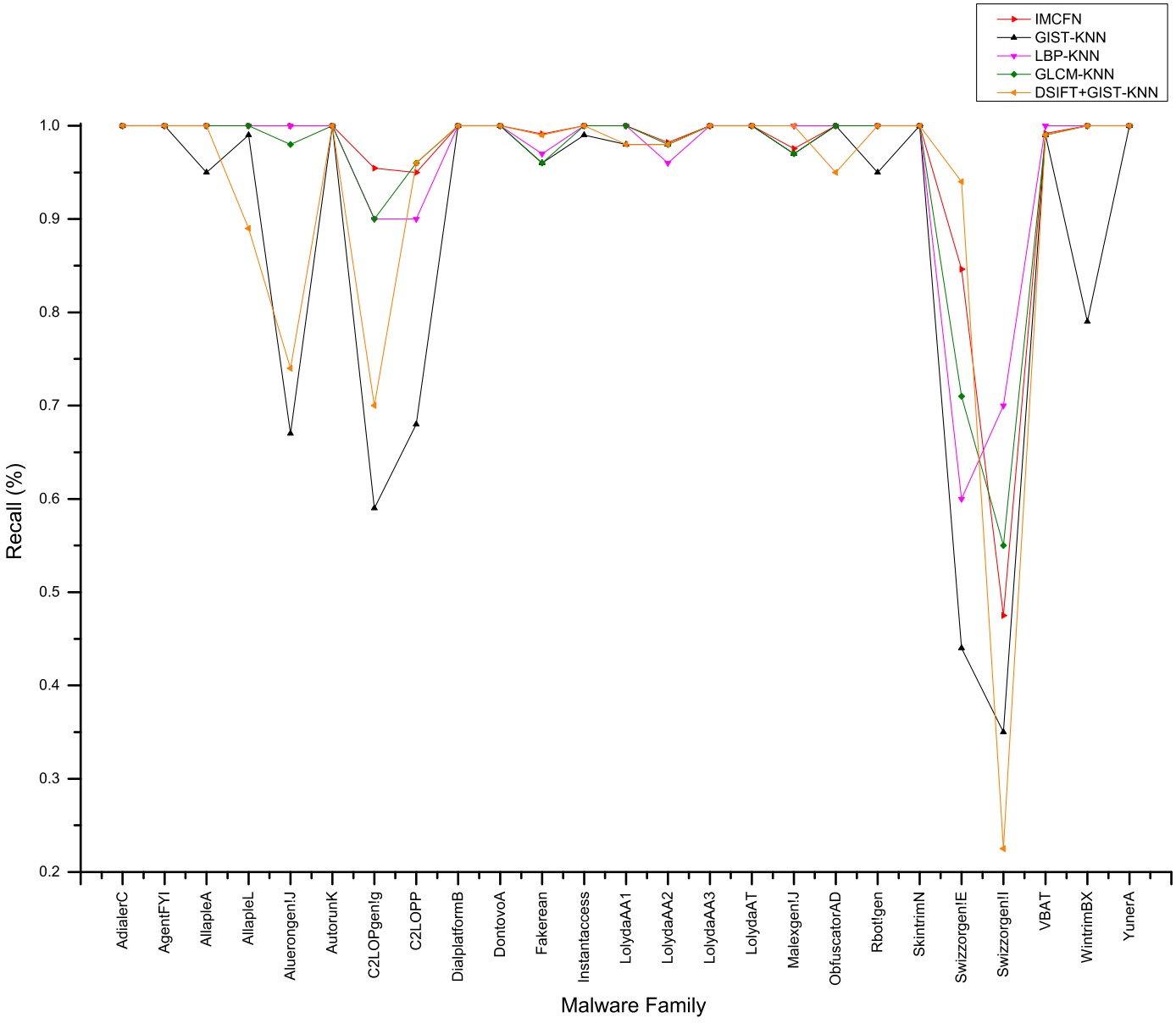


Fig. 15. Recall of different malware classification algorithms including IMCFN, GIST-KNN, LBP-KNN, GLCM-KNN, DSIFT-KNN, DSIFT+GIST-KNN in 25 malware families.

constantly modified until the optimum value was achieved. Generally, higher learning rates resulted in overfitting; however, lower rates limited error variations across epochs (i.e., slow learning).

Weight fluctuations were controlled momentum term $\eta\alpha\Delta w_t$ in our work via adding a proportion of changes in previous iteration to current one [Eq. (2)]. Therefore, higher α values decreased fluctuations by forcing weight changes along similar direction to the previous iteration creating faster and smoother convergence to optimal weight. Lower α values were often employed in previous iterations while learning process might not be globally optimal and serious variations could be more acceptable. We adopted in all epochs since we employed CNNs previously trained on a large image dataset whose pretrained weights were suitable for a variety of image data.

Weight decay term $\eta\lambda w_t$ regularized gradient descent via the prevention of the growing of weights to very high values. Prevention of overfitting was also very important. We applied $\lambda = 1$ as default value. Training statistics of IMCFN algorithm is summarized in Table 2 the number of weights requiring fine-tuning over 10 epochs.

4.3. Malware image normalization

The normalization is a preprocessing step for data which used to normalize the input data according to the network settings. The malware images are created from malware binaries and are not fixed in size. To resolve this issue, we first resized the malware images into a fixed rectangular size 224×224. In this way, the malware images were normalized and ready to enter IMCFN algorithm. The main benefits of normalization process were that it decreased the size of input images and much favorable for network training. Besides this, some important features were lost during the dimensionality reduction process.

During the normalization process, the majority of malware images in our dataset well maintained their texture features for example, the variants of Agent.FYI family in Fig. 8, the original image dimension was 64×257. While after normalization into 224×224 size, the texture features remained sharp such as the top of the portion was blue, and the bottom part was a mixture of various colors.

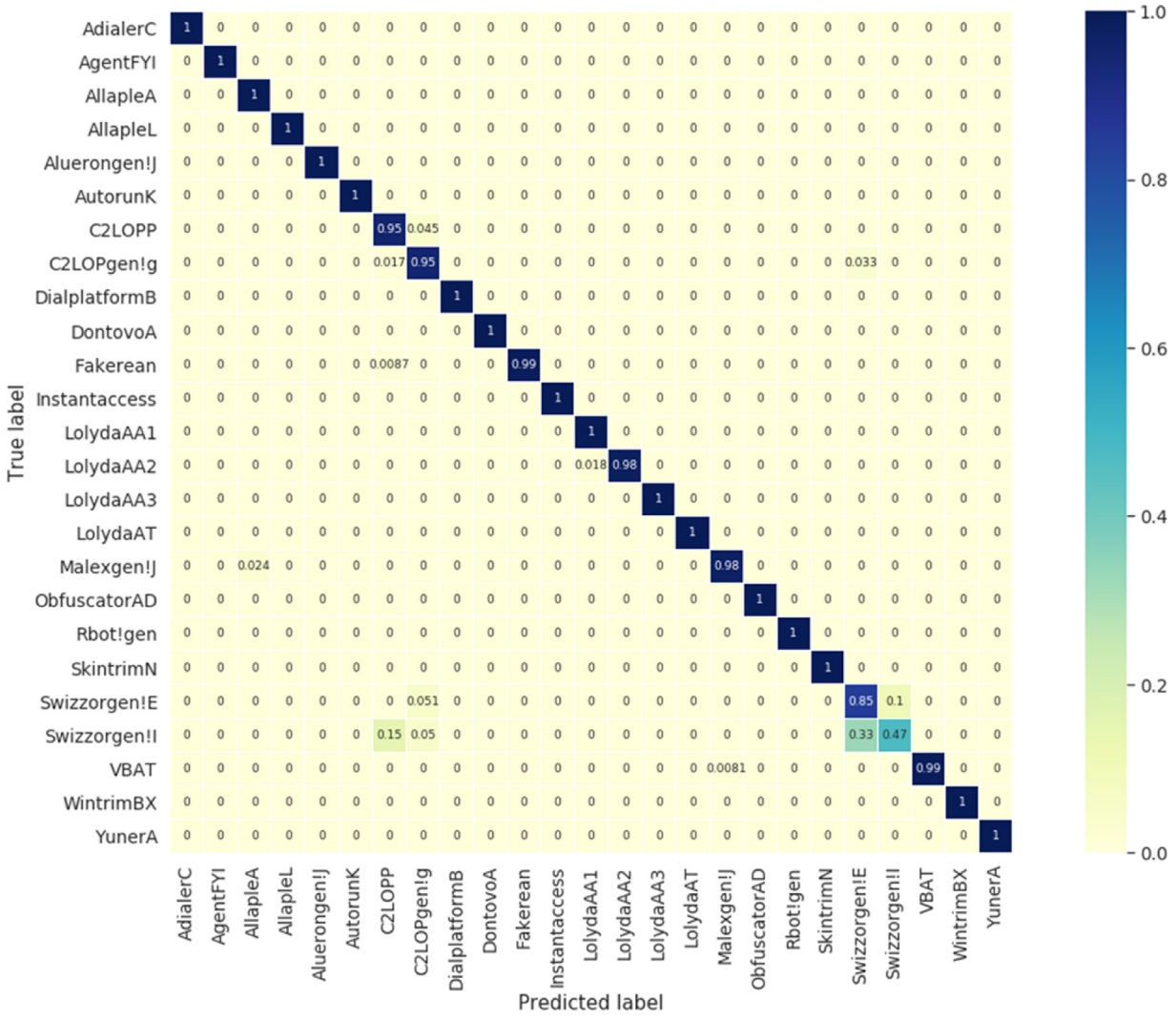


Fig. 16. A confusion matrix for IMCFN algorithm.

4.4. Data augmentation

In deep learning, to avoid overfitting, we usually need to enter sufficient data to train the model. If the data sample is relatively small, we can use data augmentation to increase the sample, thereby restraining the influence of imbalanced data. The appropriate data augmentation method can avoid overfitting problems effectively, and improve the robustness of the model.

Generally, transformation of the original image data (changing the location of image pixels and ensuring that the features remain unchanged) is used to generate new data. There are many kinds of data augmentation techniques for images, for example, rotation/reflection, flip, zoom, shift, scale, contrast, noise, and color transformation. Let $A = [a1, a2, ..., a8]$ be a collection of these techniques. $M_i = a_m, \ldots., a_n$ is the sequence of Table 3 presents the augmentation settings used in our experiments.

Typically, each augmentation technique has a weight λ_i, so the sequence of operations of weighted data augmentation techniques can be given by

$$M_i = \lambda_m a_m, \ldots.., \lambda_n a_n \tag{3}$$

where $\lambda_m$is the weight of the augmentation technology, and $a_m \subseteq$ A. Usually it is necessary to perform multiple data augmentations for a color image matrix m, $M_k(m)$ represents k augmentations. Fig. 9 shows some images generated by data augmentation technique for training. Clearly, most of the newly generated images retained the original texture features. This scheme generated 65,622 augmented training images from the 6527 training images.

4.5. Evaluating obfuscation resistance

Malware authors adopt circumvention techniques such as polymorphic and metamorphic obfuscations so that they cannot be detected by antimalware solutions. As indicated by Alazab et al. [42] the use of obfuscation methods continues to pose a major threat, the main two obfuscations types classified as:

A. ***Metamorphic obfuscation*** changes the code itself without the need of using encryption and while keeping the same functionality. Overall, there are four techniques commonly used for metamorphic obfuscation. These are:
   (i) **Dead-code Insertion**: also known as trash insertion, it does nothing to the code logic but change the byte string of the code can be difficult to detect such as changes in the code using complicated code of sequence of operations if the function or code has no effect, used to change the virus signature to some extent. Examples; [NOPs: No Operation Performed], `[MOV eax, eax]`, `[SHL eax, 0]`, [ADD eax, 0] and [INC eax] followed by `[DEC eax]`, not only, also passing values through memory rather than registers

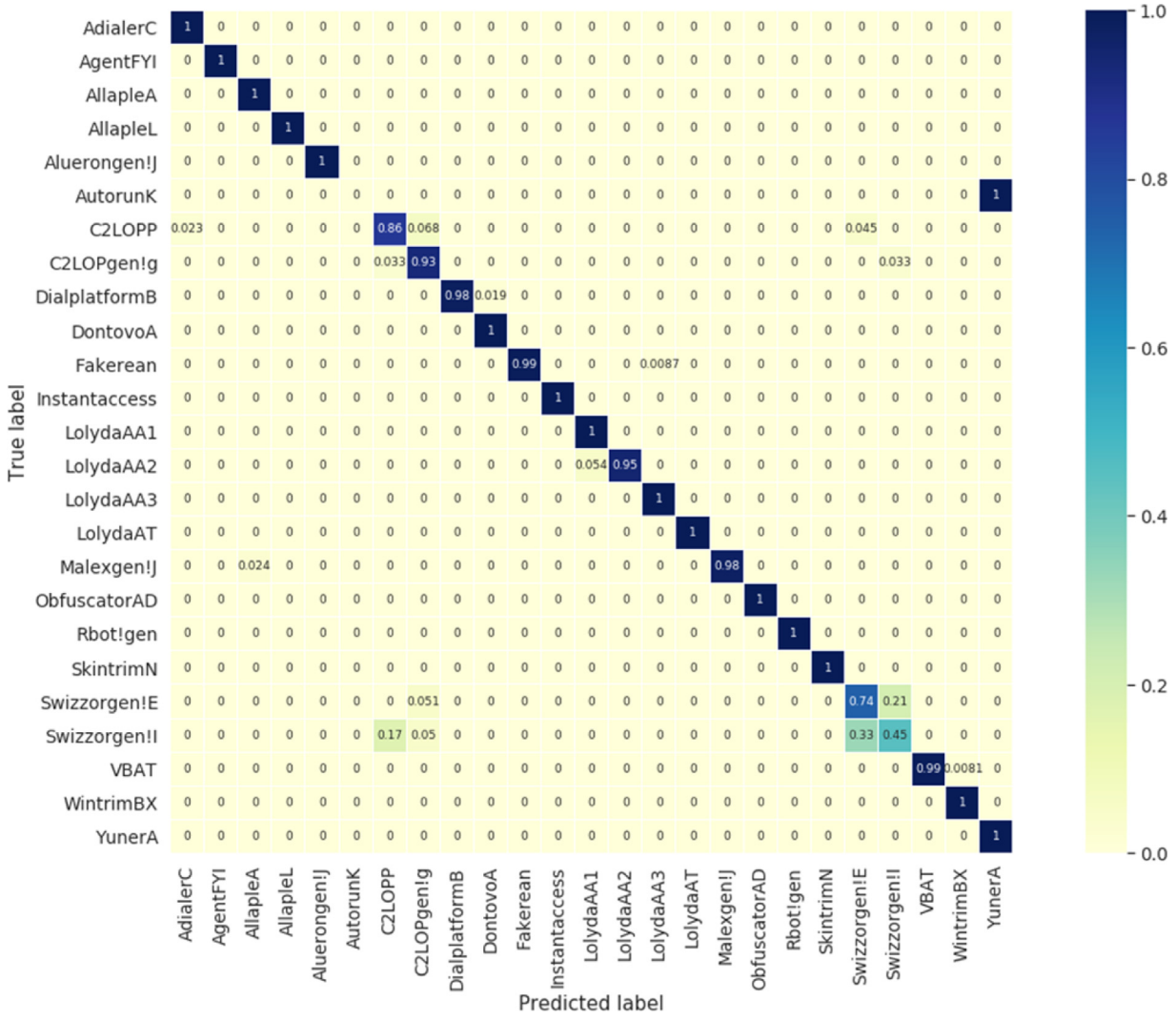


Fig. 17. A confusion matrix for VGG16.

(ii) **Code Transposition,** shuffles the instruction sequence so that the order in the binary content is completely different, such as using `JMP`, `CALL` instructions so that the order of instructions is different from the original one. Code Transposition such as Subroutine Permutation, Subroutine Inlining and Subroutine Outlining, results in changing of the instructions such that the order of instructions is different than the original code.

(iii) **Register Reassignment,** replaces code between registers by exchanging register names with no other effect on program behavior. For example, if register ebx is dead throughout a given live range of the register eax, it can replace eax in that live range. such as replacing `[PUSH ebx]` with `[PUSH eax]` to exchange register names,

(iv) **Instruction Substitution,** uses dictionary of equivalent instruction sequences to replace one instruction sequence with another, for example:

[MOV EAX, ECX] equivalent to [XOR EBX, EAX], or
[XOR EAX, EAX] equivalent to [MOV EAX, 0]

B. ***Polymorphic obfuscation***, uses encryption and data appending/data pre-pending in order to change the body of the malware, and further, it changes decryption routines from infection to infection as long as the encryption keys change, making it very difficult to create signatures to block infections. *Packing,* is commonly used today for code obfuscation or compression. Packers are software programs that could be used to compress and encrypt the PE in secondary memory and to restore the original executable image when loaded into main memory (RAM). Polymorphic packers have the ability to make its "signature" mutate. This obfuscation method is the most popular obfuscation techniques used among malware writing and distributors.

Statistics reports and literatures suggest that more than 80% of malware binaries use obfuscation methods to avoid detection [56]. Also, suggests that texture-based malware classification can be resilient to obfuscation methods. As In CNN-based approach classifies pattern and edges displaced in space through the convolution and pooling operations. Once the malware binary is transformed to an image, a texture-based feature is computed on the image for malware classification. For this task, our analysis is focused only on the polymorphic obfuscation/packing, and considering the 25 most prevalent malware families, as reported in Table 4. The experimental result demonstrated that our proposed IMCFN is resistance to polymorphic obfuscation/packing, as well as, demonstrated that texture-based malware detection is effective. Fig 10 shows some packed malware samples images from malimg ObfuscatorAD family.

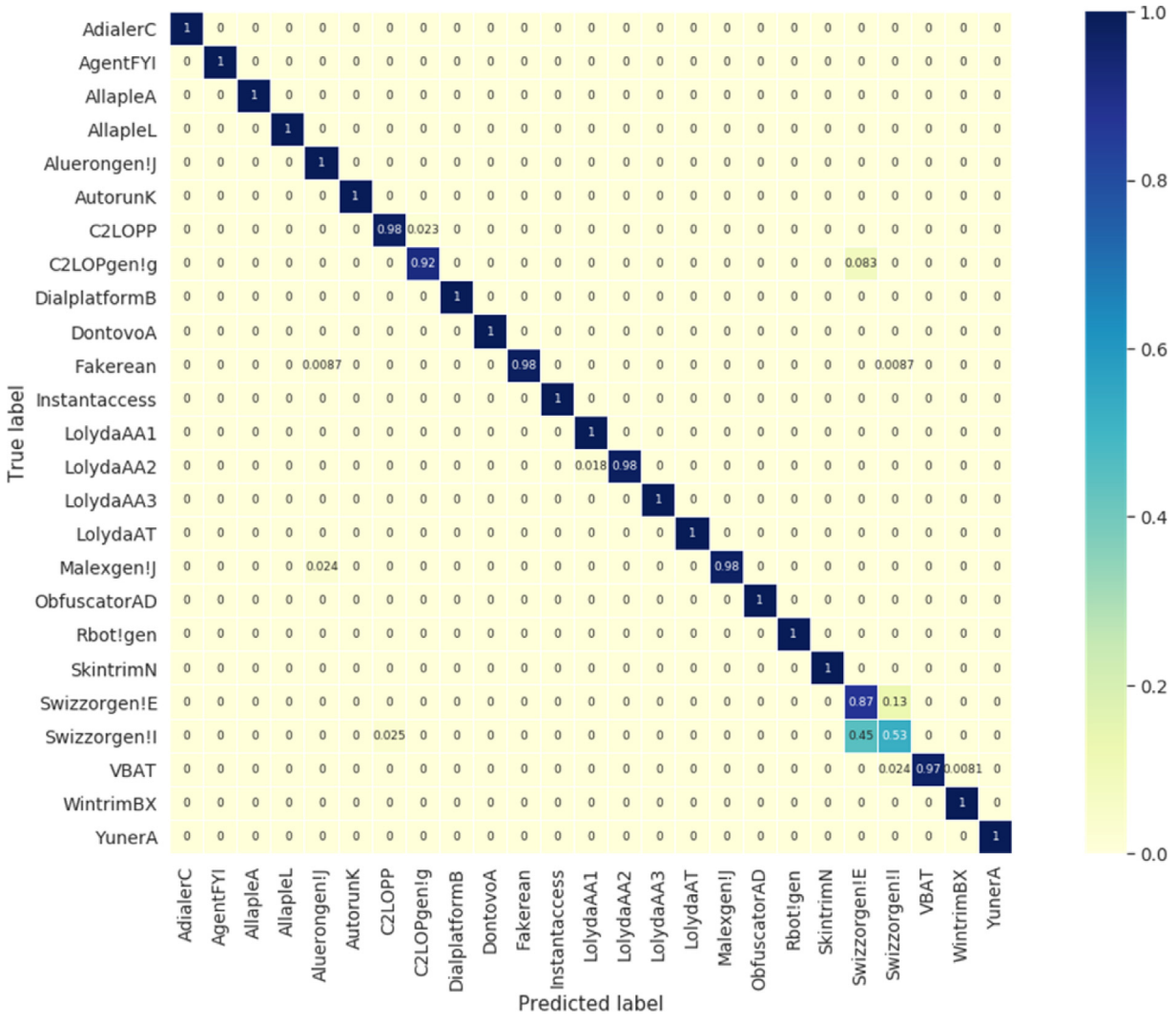


Fig. 18. A confusion matrix for InceptionV3.

5. Datasets and statistical measures

5.1. Datasets and experimental setup

We conduct the evaluation experiments under two types of datasets:

1) Malimg malware dataset [8] is a real-life malware contains 9435 malwares collected from 25 malware families, the details of each family including number of each samples in each family is shown in Table 4. We partition the data as follows: 70% for training and 30% for testing.
2) IoT- android mobile dataset: 14,733 malware and 2486 benign samples [43]. We partition the data as follows: 70% for training and 30% for testing, (See Table 5).

For handling the data imbalance problem, we used data augmentation during the fine-tuning network, next section provides further details on this.

The experimental investigation analysis was carried out by implementing the various programs in Python Programming Language. The experiment was run in NIVIDIA Ti-1080 12GB GPU for training and Intel Core i7-4790 processor with 8 GB main memory for classification. The implemented analysis in Python aided in the programs classification and detecting malware variants.

5.2. Statistical measures

In this paper, we have used state of the art malware classification algorithm including GIST+KNN, LBP+KNN, GLCM+KNN and DSIFT+GIST+KNN for all these baselines.

To evaluate the performance of classifiers, we have considered: accuracy, f1-score, recall or TPR (true positive rate), precision, and FPR (false positive rate). These evaluation metrics have been extensively used in research community to provide detailed assessments of methods [25,36,44].

True Positive (TP): means correct detection of a benign.

True Negative (TN): means correct identification of malware.

False Positive (FP): means false identification of malware as a benign application.

False Negative (FN): means false identification of a benign file as malware.

Accuracy is defined as the ratio of correctly predicted outcomes to the sum of all predictions, and is defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision determines if the positive predictions of the model are correct and is calculated by dividing the sum of true positives by all

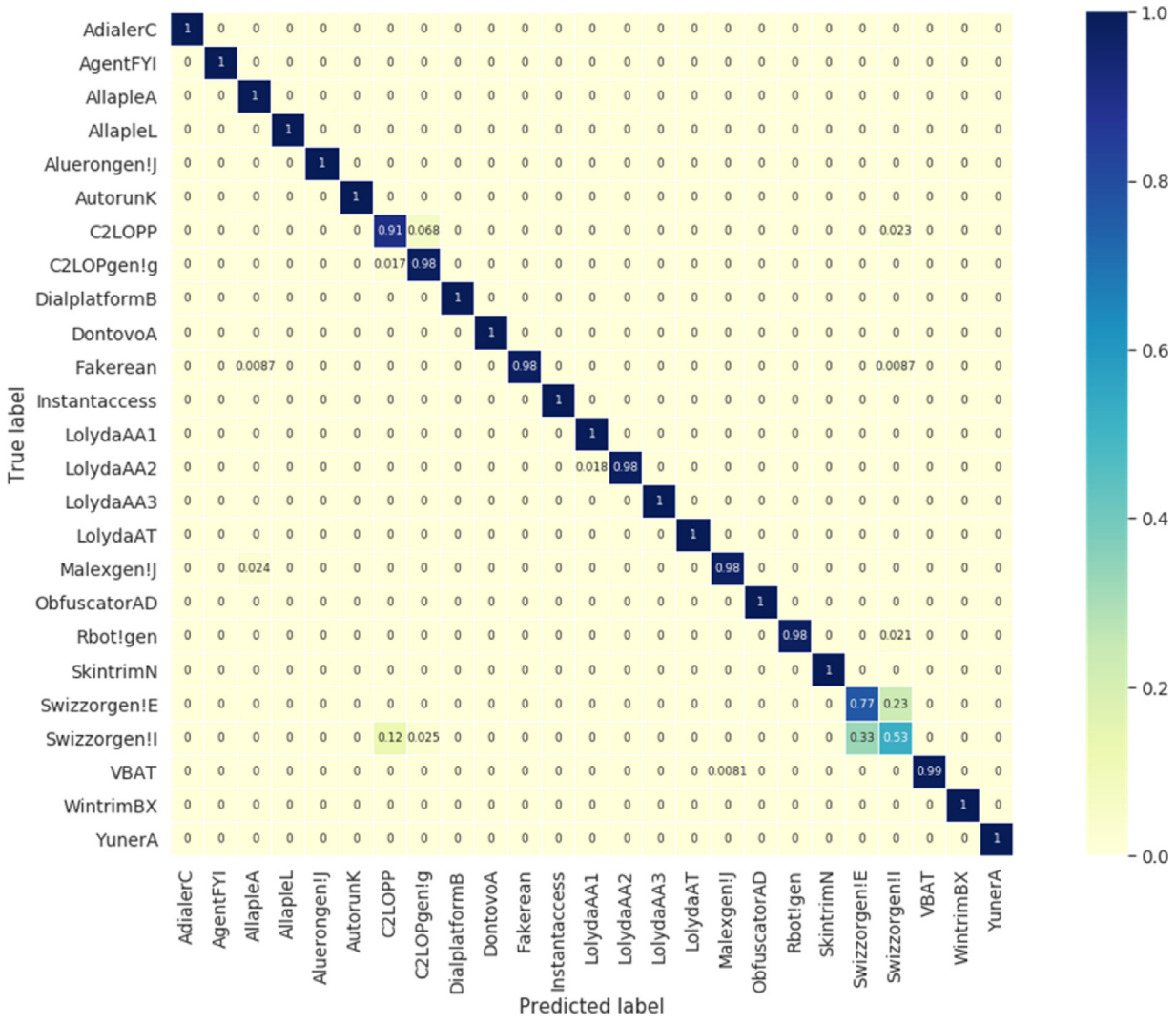


Fig. 19. A confusion matrix for ResNet50.

positive predictions as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

Recall is the positives identified by the model among all possible positives and is obtained by dividing true positives by the sum of actual positives as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

F1 score is the weighted average of recall and precision. Balanced F-score (F1 score) or F-measure is the harmonic mean of recall and precision, stated as Eq. (7).

$$\text{F1} = 2\,\text{X}\frac{Precision * Recall}{Precision + Recall} \tag{7}$$

6. Experiments and results

To validate the effectiveness and efficiency of the proposed model (IMCFN), we designed based on the following: (1) Impact of data augmentation technique on classification accuracy, (2) Comparison the performance with grayscale and color malware samples, (3) Comparison with other texture-based malware classification, (4) Comparison with VGG16 and InceprtionV3 (5) Comparison of IMCFN algorithm performance over previously studied malware family classification techniques, and (6) Validation with IoT-Android mobile dataset.

6.1. Impact of data augmentation technique on performance matrices

As the distribution of malware families and samples are highly imbalanced. Data augmentation is a possible approach to balance the data. To verify the impact of data augmentation technique on performance metrices, we compared the results when no augmentation technique was used (No-Aug) with IMCFN algorithm.

As shows in Table 6, experiment results show that the F1-score of 98.75% for IMCFN algorithm and 97.81% for No-Aug algorithm. The IMCFN algorithm was 1.01% more accurate than No-Aug algorithm. This suggests that both algorithms achieved a high accuracy but the performance of the IMCFN is outperformed the No-Aug in term of accuracy, precision, recall and f1-score for both color and grayscale Malimg datasets.

6.2. Comparison the performance between grayscale and color samples

Different from previous related research works, this work has advanced further from a previous work, most research methods convert section information of malware binary into gray scale images. In this research we argue that color images can be more effective in identifying variants and classification. The main assumption that color images contains color natural images, and has more sharp features than grayscale, this likely to increase the effectiveness and the accuracy of the detection

Table 7
A Comparative Summary of IMCFN Algorithm with other State-of the-art CNN Architectures.

Dataset	IMCFN (224×224)				VGG16 (224×224)				InceptionV3 (299×299)				ResNet50 (224×224)			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Malimg	98.82	98.85	98.81	98.75	97.12	96.26	97.29	96.72	98.65	98.80	98.64	98.66	98.61	98.58	98.61	98.58
Prediction Time (Sec)	0.81				0.97				0.69				0.63			

and classification. As our model converts the raw malware binary into both grayscale and color images. In this section, we compared the performance of our model on both color and grayscale images. We chose two Malimg datasets (color and grayscale) with same distribution of the samples in the malware families. We found that the overall classification accuracy for the color samples was better than grayscale samples. The performance metrices results are summarized in Table 6 for both color and grayscale samples, as shows, it is clear that our model performed better with color samples compared to grayscale samples. Also, this suggest that color can provide more effective and accuracy in malware classification and variants identifications.

6.3. Comparison with other texture-based malware classification

As indicated earlier in (Section V) that literatures suggest that more than 80% of malware binaries use obfuscation methods and that texture-based malware classification can be resilient to obfuscation methods and improve the accuracy. As malware variants from each family exhibit some significant amount of texture. Image texture is a block of pixels with repeated patterns. In Fig. 10, we show that samples from the same malware family (Obfuscator.AD) has a similar patterns and image texture among themselves. In this paper, we implement, and compare our model (IMCFN) with four well-known texture-based malware classifications, each algorithm repeats 5 times for random sampling.

1) GIST descriptors, in the first experiment we extracted texture features from malware color images using GIST descriptor. GIST has been selected as it is a useful feature descriptor that has been widely used in existing study [45].
2) Visual descriptors, the Local Binary Pattern (LBP) features, we extracted local binary patterns from malware color images using LBP descriptor. Like GIST, LBP has been selected as it is useful for texture analysis and texture classification and widely used in malware images [55].
3) Gray-level Co-occurrence Matrix (GLCM) features, for texture feature extraction from grayscale malware images. GLCM provided ideal texture features of malware images [46], and the computational complexity is relatively lower than other algorithms
4) Dense Scale Invariant Feature Transform (DSIFT) Combined with GIST feature descriptor, (GLCM+ GIST). We extracted hybrid local and global features from malware color images. We selected 512-dimensional GIST feature, 512-dimensional LBP feature, 512-dimensional DSIFT+GIST feature and 20-dimensional GLCM feature.

In order to compare our proposed model with original global and local descriptors, we report the results of IMCFN and the above four descriptors using KNN as a classifier. For this task, we used the 25-malware family in the Malimg malware dataset. Figs 11-15 shows the Accuracy, Precision, Recall rate, F1 score and FPR (false positive rate) of the five different algorithms. In term of accuracy our model has achieved 98.82%, which is better than the other four texture-based malware classifications, which means that our model has much more stable performance, as shows in Fig 11. As demonstrated in Fig 14, it also can be seen that IMCFN has the highest F1 score of 98.75%, compared to GIST with 87.94%, LBP with 96.12%, GLCM with 95.76%, and DSIFT+GIST with 93.12%. Since there are only 142 samples in the family Obfuscator.AD, most of the algorithms cannot recognize Obfuscator.AD very well. It is worth noting that IMCFN algorithm keeps an excellent performance. Also, Fig 11-15 show IMCFN algorithm found superior in all of the statistical measures for all malware families, even in family Swizzorgen!E and Swizzorgen!I where other algorithms struggle. Therefore, IMCFN shows a degree of advantages for those similar malware variants. Also, the result suggests the capability of IMCFN to discover correspondences between similar visual contents, and allows the malware analysis to classify malware and identify variants.

Table 8
A Comparative Summary of IMCFN Algorithm with Previously Studied Multi-class Malware Family Classification Techniques.

Methods	Year	Dataset	Binary Visualization Method	Technique	Precision (%)	Recall (%)	Accuracy (%)	F1-Score (%)
Nataraj et al. [19]	2011	Malimg	Grayscale	Machine Learning	–	–	97.18	–
Vgg-verydeep-19 [34]	2017	Malimg	Grayscale	Deep Learning	–	–	97.32	–
Gabor wavelet-KNN [10]	2018	Malimg	Grayscale	Machine Learning	–	–	89.11	–
Zhihua et al. [35]	2018	Malimg	Grayscale	Deep Learning	–	88.40	97.60	–
Yajamanam et al. [45]	2018	Malimg	Grayscale	Machine Learning	–	–	97	–
Bhodia et al. [51]	2018	Malimg	Grayscale	Deep Learning	–	–	94.80	–
GIST+SVM [35]	2018	Malimg	Grayscale	Machine Learning	92.5	91.4	92.2	–
GIST+KNN [35]	2018	Malimg	Grayscale	Machine Learning	92.10	91.70	91.90	–
GLCM+SVM [35]	2018	Malimg	Grayscale	Machine Learning	93.40	93	93.20	–
GLCM+KNN [35]	2018	Malimg	Grayscale	Machine Learning	92.70	92.30	92.50	–
IDA+DRBA [35]	2018	Malimg	Grayscale	Deep Learning	94.60	94.50	94.50	–
NSGA-II [52]	2019	Malimg	Grayscale	Deep Learning	97.6	88.4	97.6	–
IMCFN	**–**	Malimg	**Color**	**Deep Learning**	**98.85**	**98.81**	**98.82**	**98.75**

Table 9
Performance of IMCFN algorithm and No-Aug algorithm on IoT-Android Mobile Dataset.

Dataset	IMCFN				No-Aug			
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
IoT-Android Mobile Dataset	97.35	97.30	97.34	97.31	96.69	96.66	96.68	96.67

6.4. Comparison with VGG16, resnet50, and inceprtionv3

In this experiment, we compared the IMCFN performance with VGG16, ResNet50 and InceptionV3 already trained on ImageNet dataset. These state-of-the-art architectures have previously been used to solve the multi-class malware family classification problem [47-50]. For this experiment, we obtain an overall classification accuracy of 97.12% for VGG16 which represents a significant decline from the IMCFN accuracy of 98.82%, 98.61% for ResNet50 and 98.65% for InceptionV3 which represent a modest decline from the IMCFN accuracy of 98.82%, the others performance metrices are summarized in Table 7. Beside this, we calculate the classification time for all of these state-of-the-art architectures, we obtain 0.97 s on average for VGG16, 0.63 s on average for ResNet50, 0.69 s on average for InceptionV3 and 0.81 s on average for IMCFN algorithm. IMCFN was 1.70% more accurate than VGG16 which is significant increase, 0.21% and 0.17% more accurate than ResNet50 and InceptionV3 which is modest increase.

When we predict a new malware sample, the process mainly consists of the two parts (generating a malware fingerprint image and predicting it by CNN fine-tuned model). Experimental results show that identifying a malware sample takes o.81 s in average.

The confusion matrix values are composed of the true positive rate and the false negative rate of the malware family classification. Figs 16-19 are showing confusion matrixes of the IMCFN, VGG16, ResNet50 and InceptionV3 for 25 malware families. The value of the sub diagonal represents the true positive rate, and the other values indicate the false negative rate.

6.5. Comparison with existing malware classification

We compare the performance of IMCFN algorithm with existing malware classification study, which used image-based malware classification techniques based on machine and deep learning methods. Firstly, these techniques extracted features from the malware images, and then applied machine learning or deep learning classifiers (e.g., Softmax, KNN, SVM) for multi-class classification. For this comparison, we used IMCFN algorithm. Table 8 presents the recent reported results of previously discussed state-of-the-art study on multi-class malware family classification problem. From the reported results it is found that IMCFN algorithm is superior than existing classification techniques.

6.6. Validation with iot-android mobile dataset

Next, we validate the IMCFN algorithm with IoT-android mobile dataset of 14,733 malware and 2486 benign samples. First, we evaluate the IoT-android mobile dataset when No-Aug is applied. In this case, we might intuitively expect that the accuracy will be decrease, as they are more imbalance, and this imbalance of the samples in the family will likely result in more misclassifications. For this experiment we attained an overall accuracy of 96.69% for No-Aug algorithm.

For our next experiment, we evaluate the IoT-android mobile dataset with IMCFN algorithm. In this case, we might expect that the overall classification accuracy will actually improve. This follows, because IMCFN algorithm uses the data augmentation technique during the fine-tuning, and hence there will likely be fewer misclassification involved. For this experiment we achieved an accuracy of 97.35% for IMCFN compared to the No-Aug algorithm with an accuracy of 96.69%, results are summarized in Table 9.

The experimental results show the excellent recognition capability of IMCFN algorithm, even for unevenly distributed samples. Experimental results demonstrate that the dynamic features extraction capability of the IMCFN algorithm recommends that IMCFN algorithm could be voluntarily adjusted to different malware datasets.

7. Conclusion

There is a constant race between cyber-attackers and antimalware software. Therefore, it is necessary to maintain the persistent pressure on cyber-attackers. Malware is a common attack vector to the internet. In this paper, we propose a malware classification algorithm called IMCFN which combines malware visualization and fine-tuned CNN architecture already trained on ImageNet dataset. During the fine-tuning, data augmentation method is adopted to improve the performance of the IMCFN algorithm. The experimental results show the excellent classification capability of IMCFN. The classification accuracy is 98.82% for Malimg dataset, which is higher than other discussed methods.

Compared with well-known classification methods based on gray and colored images analysis, our model achieves lower computational costs and better results in terms of accuracy. Furthermore, to validate the efficiency and sustainability we evaluated IMCFN algorithm with an imbalance IoT-android mobile dataset of 14,733 malware and 2486 benign samples. Experimental outcomes proved that IMCFN algorithm is also remained effect full with 97.56% accuracy rate. For IMCFN, on average,

it just took 0.81 s to recognize a new sample. The transformation of malicious code into color images need to be explored in future research.

Declaration of Competing Interest

There is no conflict-of-interest in this study.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2020.107138.

Reference

[1] Symantec Corporation, Symantec internet security threat report, Netw. Secur. (2019).
[2] Kaspersky, https://www.smetechguru.co.za/number-of-users-attacked-by-banking-trojans-grew-by-16-in-2018-reaching-almost-900000.
[3] J. Su, V. Danilo Vasconcellos, S. Prasad, S. Daniele, Y. Feng, K. Sakurai, Lightweight classification of iot malware based on image recognition, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2, 2018, pp. 664–669.
[4] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey, Inf. Secur. Tech. Rep. (2009).
[5] M. Egele, T. Scholte, E. Kirda, C. Kruegel, A survey on automated dynamic malware–analysis techniques and tools, ACM Comput Surv (2012).
[6] M. Alazab, Profiling and classifying the behavior of malicious codes, J. Syst. Softw. (2015).
[7] S. Venkatraman, M. Alazab, Use of data visualisation for zero-day malware detection, Secur. Commun. Networks (2018).
[8] L. Nataraj, S. Karthikeyan, G. Jacob, B.S. Manjunath, Malware images: visualization and automatic classification, Proc. 8th Int. Symp. Vis. Cyber Secur. (2011) 4.
[9] X. Ban, C. Li, W. Hu, W. Qu, Malware variant detection using similarity search over content fingerprint, in: 26th Chinese Control and Decision Conference, CCDC 2014, 2014, pp. 5334–5339.
[10] A. Makandar, A. Patrot, Malware class recognition using image processing techniques, 2017 Int. Conf. Data Manag. Anal. Innov. ICDMAI 2017 (2017).
[11] P. Trinius, T. Holz, J. Göbel, F.C. Freiling, Visual analysis of malware behavior using treemaps and thread graphs, in: 6th International Workshop on Visualization for Cyber Security 2009, VizSec 2009 - Proceedings, 2009.
[12] S.Z.M. Shaid, M.A. Maarof, Malware behaviour visualization, J. Teknol. (2014).
[13] K. Han, B. Kang, E.G. Im, Malware analysis using visualized image matrices, Sci. World J. (2014).
[14] Y. Du, J. Wang, Q. Li, An android malware detection approach using community structures of weighted function call graphs, IEEE Access (2017).
[15] K. Han, J.H. Lim, E.G. Im, Malware analysis method using visualization of binary files, in: Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS 2013, 2013.
[16] K.S. Han, J.H. Lim, B. Kang, E.G. Im, Malware analysis using visualized images and entropy graphs, Int. J. Inf. Secur. 14 (1) (2014).
[17] K. Kancherla, J. Donahue, S. Mukkamala, Packer identification using byte plot and markov plot, J. Comput. Virol. Hacking Tech. (2016).
[18] K. Kancherla, S. Mukkamala, Image visualization based malware detection, in: Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013, 2013.
[19] L. Nataraj, V. Yegneswaran, P. Porras, J. Zhang, A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proceedings of the ACM Conference on Computer and Communications Security, 2011.
[20] B.N. Narayanan, O. Djaneye-Boundjou, T.M. Kebede, Performance analysis of machine learning and pattern recognition algorithms for malware classification, in: Proceedings of the IEEE National Aerospace Electronics Conference, NAECON, 2017.
[21] H. Naeem, B. Guo, M.R. Naeem, A light-weight malware static visual analysis for iot infrastructure, 2018 Int. Conf. Artif. Intell. Big Data, ICAIBD 2018 (2018) 240–244.
[22] A. Graves, N. Jaitly, Towards end-to-end speech recognition with recurrent neural networks, in: ICML'14 Proceedings of the 31st International Conference on Machine Learning, 2014.
[23] K. Simonyan, A. Zisserman, VGG-16, arXiv Prepr (2014).
[24] H. Yu, J. Wang, Y. Bai, W. Yang, G.S. Xia, Analysis of large-scale uav images using a multi-scale hierarchical representation, Geo-Spatial Inf. Sci. (2018).
[25] S. Ni, Q. Qian, R. Zhang, Malware identification using visualization images and deep learning, Comput. Secur. 77 (Aug. 2018) 871–885.
[26] Q. Le, O. Boydell, B. Mac Namee, M. Scanlon, Deep learning at the shallow end: malware classification for non-domain experts, Digital Investigation, 2018, pp. S118–S126. 26, Supplement, July.
[27] J. Yan, Y. Qi, Q. Rao, Detecting malware with an ensemble method based on deep neural network, Secur. Commun. Networks 2018 (2018) 1–16.
[28] M. Kalash, M. Rochan, N. Mohammed, N.D.B.B. Bruce, Y. Wang, F. Iqbal, Malware classification with deep convolutional neural networks, in: 2018 9th IFIP Int. Conf. New Technol. Mobil. Secur., 2018, pp. 1–5.
[29] R. Kumar, Z. Xiaosong, R.U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," pp. 81–85, 2018.
[30] R.U. Khan, X. Zhang, R. Kumar, Analysis of resnet and googlenet models for malware detection, Journal of Computer Virology and Hacking Techniques (2018).
[31] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access (2019).
[32] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning, IEEE Access (2019).
[33] S. Venkatraman, M. Alazab, R. Vinayakumar, A hybrid deep learning image-based analysis for effective malware detection, J. Inf. Secur. Appl. (2019).
[34] S. Yue, "Imbalanced malware images classification: a cnn based approach," 2017.
[35] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, J. Chen, Detection of malicious code variants based on deep learning, IEEE Trans. Ind. Informatics 14 (7) (2018) 3187–3196 Jul.
[36] A.P. Namanya, I.U. Awan, J.P. Disso, M. Younas, Similarity hash based scoring of portable executable files for efficient malware detection in iot, Futur. Gener. Comput. Syst. (2019).
[37] A. Krizhevsky, I. Sutskever, H. Geoffrey E., Imagenet, Adv. Neural Inf. Process. Syst. 25 (2012).
[38] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, ImageNet Classification with Deep Convolutional Neural Networks, 2012.
[39] J. Chang, J. Yu, T. Han, H.J. Chang, E. Park, A method for classifying medical images using transfer learning: a pilot study on histopathology of breast cancer, 2017 IEEE 19th International Conference on E-Health Networking, Applications and Services, Healthcom 2017, 2017.
[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016.
[41] C. Szegedy, et al., Going deeper with convolutions, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015.
[42] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, A. Alazab, Cybercrime: the case of obfuscated malware, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2012.
[43] Mila, DSEncrypt android sample, contagio mobile (2015).
[44] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015, pp. 11–20.
[45] S. Yajamanam, V.R.S. Selvin, F. Di Troia, M. Stamp, Deep learning versus gist descriptors for image-based malware classification, no. Icissp (2018) 553–561.
[46] X. Chen, H. Zhong, Z. Bao, A GLCM-feature-based approach for reversible image transformation, Comput. Mater. Contin. 59 (1) (2019) 239–255.
[47] M. Shaha, M. Pawar, Transfer learning for image classification, in: Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, 2018, pp. 656–660.
[48] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, P. de Geus, Malicious software classification using VGG16 deep neural network's bottleneck features, Advances in Intelligent Systems and Computing, 2018.
[49] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, P. De Geus, Malicious software classification using transfer learning of resnet-50 deep neural network, in: Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, 2018.
[50] S. Das, CNN architectures: leNet, alexnet, VGG, googlenet, resnet and more, Medium (2017).
[51] N. Bhodia, P. Prajapati, F. Di Troia, and M. Stamp, "Transfer learning for image-based malware classification," 2018.
[52] Z. Cui, L. Du, P. Wang, X. Cai, W. Zhang, Malicious code detection based on CNNs and multi-objective algorithm, J. Parallel Distrib. Comput. 129 (Jul. 2019) 50–58.

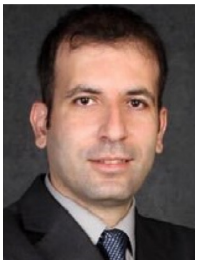
Danish Vasan He received the Master of Philosophy in Computer Science (MPhil-CS) degree from Isra University of Pakistan in 2014. Presently, he is pursuing Ph.D. in School of Software Engineering, Tsinghua University China. He is a cyber security researcher and practitioner with industry and academic experience. His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems including current and emerging issues in the cyber environment like cyber-physical systems and internet of things, by taking into consideration the unique challenges present in these environments, with a focus on cybercrime detection and prevention. He has various publications in well reputed journals.

Mamoun Alazab is an Associate Professor at the College of Engineering, IT and Environment at Charles Darwin University, Australia. He received his Ph.D. degree in Computer Science from the Federation University of Australia, School of Science, Information Technology and Engineering. He is a cybersecurity researcher and practitioner with industry and academic experience. Alazab's research is multidisciplinary that focuses on cybersecurity and digital forensics of computer systems with a focus on cybercrime detection and prevention. He has more than 150 research papers in many international journals and conferences, such as IEEE transactions on Industrial Informatics, IEEE Transactions on Industry Applications, IEEE Transactions on Big Data, IEEE Transactions on Vehicu-

lar Technology, Computers & Security, and Future Generation Computing Systems. He delivered many invited and keynote speeches, 24 events in 2019 alone. He convened and chaired more than 50 conferences and workshops. He works closely with government and industry on many projects, including Northern Territory (NT) Department of Information and Corporate Services, IBM, Trend Micro, the Australian Federal Police (AFP), the Australian Communications and Media Authority (ACMA), Westpac, United Nations Office on Drugs and Crime (UNODC), and the Attorney General's Department. He is a Senior Member of the IEEE. He is the Founding Chair of the IEEE Northern Territory (NT) Subsection.

Sobia Wassan She received her Master degree in Commerce (M.Com) from Sindh University Jamshoro, Pakistan in 2016. Presently, she is pursuing Ph.D. in School of Business Administration, Nanjing University China. Her research interest includes, Sentiment Analysis using Deep Learning, Recommendation System, Use of Artificial Intelligence in E-Commerce.

Hamad Naeem received the B.E. degree in Computer Systems Engineering from the Bahauddin Zakariya University, Pakistan, in 2012, M.E. degree in Software Engineering from Chongqing University Chongqing, China, in 2016, and the Ph.D. degree in Software Engineering from Sichuan University Sichuan, China, in 2019. He received outstanding Master Student Award from Chongqing University Chongqing, China, in 2016. He is currently serving as an Associate Professor with the Department of Computer Science, Neijiang Normal University, Neijiang, China. His-research work is published in various renowned journals of Elsevier, Springer, Wiley, MDPI, and IEEE. His-research interests include cyber security, malware analysis, code clone and program analysis.



Babak Safaei He is currently an Assistant Professor in Department of Mechanical Engineering at Eastern Mediterranean University. He received his Bachelor degree, Master degree in Mechanical Engineering from School of Mechanical Engineering at Azad University-Tabriz branch and Ph.D. degree in Mechanical Engineering from Department of Mechanical Engineering at Tsinghua University. His-research area is Deep Learning, Machine Learning, Computational Mechanics, Micro and Nano Mechanics and composite materials.

Qin Zheng He is a doctoral supervisor, professor, the Director of Software Engineering and Management Research Institute and Information Institute, Tsinghua University. He is the evaluation expert of National Science and Technology Award, Ministry of Education Technology Award, Major State Basic Research Development Program (973), National High Technology Research and Development Program of China (863), National Defense 10th 5-Year Plan and Ministry of Education College Undergraduate Teaching. He is also the member of Ministry of Education E-Commerce Specialty Teaching Guidance Committee. He is now the professor of Tsinghua University and Xi'an Jiaotong University, and the part-time professor for many other high-education organizations including Ministry of National Supervision. He is the guest researcher of Shanxi Academy and Henan Academy, editor of International Journal of Plant Engineering and Management(E) and Journal of E-Business. At present, he is undertaking 6 projects of research and development, including 3 national projects (973, 863, National Defense 11th 5-year Plan), 3 foundational projects. Besides, he has finished 12 projects, including 3 national and 9 provincial ones. For decades, he has won 7 awards of national or provincial level (1 first prize, 3 s prize and 3 third prize). He has applied 10 patents of invention and acquired 15 software copy rights. He has published 2 books through the oversea presses as well as 10 books through the domestic presses, among which 3 are for management, 4 for E-Commerce, 2 for information. He has published 22 SCI-indexed papers, more than 40 EI-indexed papers and 131 papers in Chinese Core Journals (among which includes 4 in Chinese Journal of Computers, 4 in Chinese Journal of Electronics, 1 in Software Journal, 5 in Journal of Computer Research and Development, 4 in Journal of System Simulation). He is often invited to give congress lectures in information and management areas.