Image-Based Malware Classification using Ensemble of CNN Architectures (IMCEC)

Article *in* Computers & Security · May 2020

DOI: 10.1016/j.cose.2020.101748

Danish Vasan
Tsinghua University
9 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Sobia Wassan
Nanjing University
3 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Babak Safaei
Eastern Mediterranean University
70 PUBLICATIONS **918** CITATIONS
SEE PROFILE

Contents lists available at ScienceDirect

Computers & Security

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

Image-Based malware classification using ensemble of CNN architectures (IMCEC)

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

[a] School of Software Engineering, Tsinghua University, Beijing 100084, China
[b] Department of Computer Science, Isra University Hyderabad 71000, Sindh Pakistan
[c] School of Business (Business Administration), Nanjing University, Jiangsu 210000, China
[d] University of Sindh Jamshoro, Sindh, Pakistan
[e] College of Engineering, IT and Environment, Charles Darwin University, Australia
[f] Department of Mechanical Engineering, Eastern Mediterranean University, G. Magosa, TRNC Mersin 10, Turkey

ARTICLE INFO

Article history:
Received 9 June 2019
Revised 29 December 2019
Accepted 4 February 2020
Available online 29 February 2020

Keywords:
Malware
Cybersecurity
Deep learning
Transfer learning
Fine-tuning
SVMs
Softmax
Ensemble of CNNs

ABSTRACT

Both researchers and malware authors have demonstrated that malware scanners are unfortunately limited and are easily evaded by simple obfuscation techniques. This paper proposes a novel ensemble convolutional neural networks (CNNs) based architecture for effective detection of both packed and unpacked malware. We have named this method Image-based Malware Classification using Ensemble of CNNs (IMCEC). Our main assumption is that based on their deeper architectures different CNNs provide different semantic representations of the image; therefore, a set of CNN architectures makes it possible to extract features with higher qualities than traditional methods. Experimental results show that IMCEC is particularly suitable for malware detection. It can achieve a high detection accuracy with low false alarm rates using malware raw-input. Result demonstrates more than 99% accuracy for unpacked malware and over 98% accuracy for packed malware. IMCEC is flexible, practical and efficient as it takes only 1.18 s on average to identify a new malware sample.

1. Introduction

Malware is an intentionally malicious software developed to cause harm to computer systems (Rieck et al., 2011; Alazab, 2015). Recently, significant proliferation of malware used for illegal and malicious goals has been recorded (E. Rezende et al., 2018; Farivar et al., 2019). For example, Kaspersky Lab detected 69,277,289 types of malware (scripts, executable files, etc.) in 2016 (Lab, 2016). McAfee Labs have reported an increase of 22% in malware attacks in the four years prior to 2017 up to 670 million (Beek, 2017). The rising trend in malware attacks demands a highly effective approach to detect malware. Most commercial antivirus applications typically use signature-based techniques, which require local signature databases for storing patterns that experts have detected in malicious software (Jung et al., 2020). This tactic suffers from significant limitations since malware authors reuse the code to generate new malwares and apply code obfuscation techniques such as packing alter signatures. Because of this, high numbers of malware can remain undetected by detection methods based on signature. Other limitations of these techniques include the need for reverse engineering and considerable domain expertise. In recent years, static and dynamic analysis has been tried for malware detection: static analysis focuses on statistical features (e.g., N-grams, API Calls, Opcode's Sequences, etc.) (YusirwanS et al., 2015; Shijo and Salim, 2015; Yuan et al., 2016); dynamic analysis uses virtual environment (e.g., Sandbox) to analyze the behavior of malicious applications (Lindorfer et al., 2015; Damodaran et al., 2017).

Various deep and machine learning techniques are currently used to detect and classify malwares (Ni et al., 2018; Mohamed Shakeel et al., 2018; Azmoodeh et al., 2018; Kumar et al., 2018; Huda et al., 2016). Most of these techniques rely on feature engineering work or domain knowledge to build a feature database. Since new malware is constantly created and updated with some specific changes, it becomes increasingly challenging to manually update feature databases against newly generated malware samples. To reduce feature engineering cost and domain expert knowledge, researchers have used visualization approaches to solve malware family classification problems. For example, Conti

* Corresponding author.
E-mail addresses: danish.wasan@gmail.com (D. Vasan), alazab.m@ieee.org (M. Alazab), sobiaali614@gmail.com (S. Wassan), babak.safaei@emu.edu.tr (B. Safaei), qingzh@tsinghua.edu.cn (Q. Zheng).

https://doi.org/10.1016/j.cose.2020.101748
0167-4048/© 2020 Elsevier Ltd. All rights reserved.

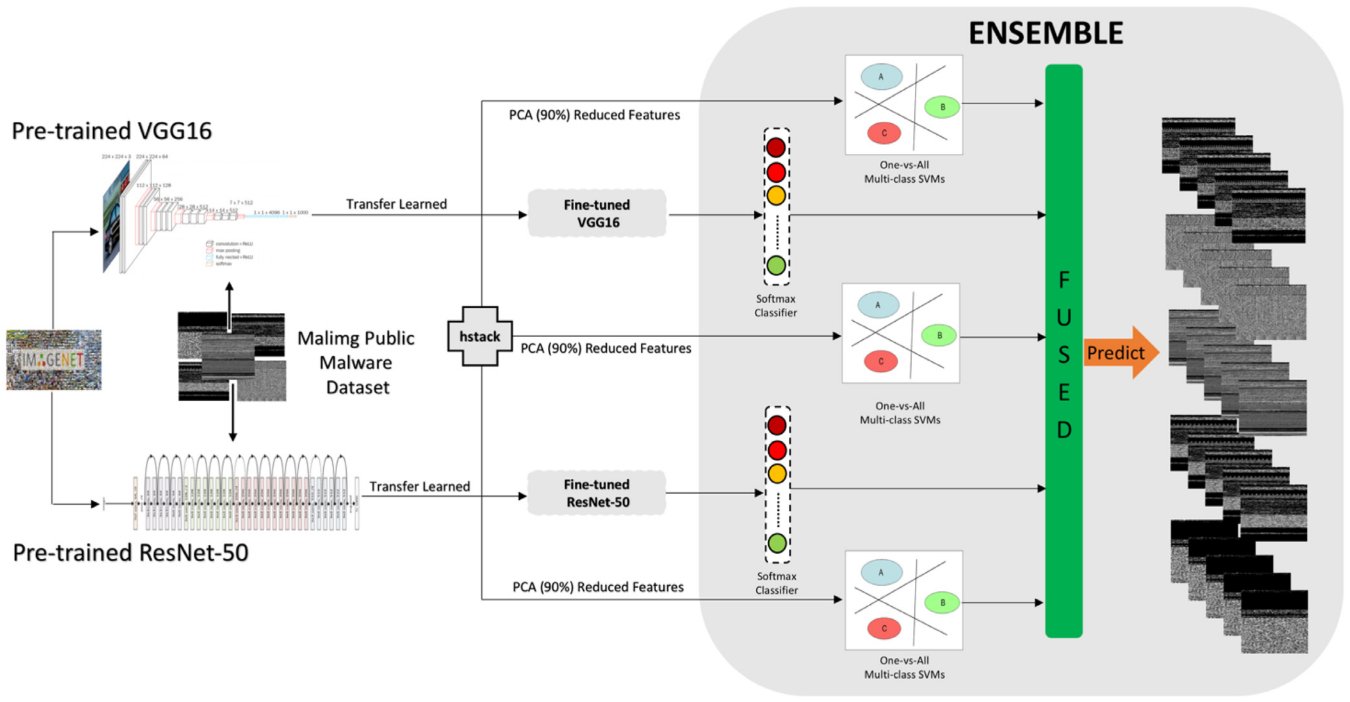

ENSEMBLE
Pre-trained VGG16
Transfer Learned
Fine-tuned VGG16
PCA (90%) Reduced Features
One-vs-All Multi-class SVMs
Softmax Classifier
Malimg Public Malware Dataset
hstack
PCA (90%) Reduced Features
One-vs-All Multi-class SVMs
F U S E D
Predict
Transfer Learned
Fine-tuned ResNet-50
Pre-trained ResNet-50
Softmax Classifier
PCA (90%) Reduced Features
One-vs-All Multi-class SVMs

Fig. 1. Overview of IMCEC.

et al. (Conti et al., 2008) have suggested a method for the visualization of malware binary into a grayscale image and argued that visual analyses of malware binary help differentiate various regions of data in the image. Kancherla et al. (Kancherla and Mukkamala, 2013) extracted low-level features like intensity-based and texture features from malware binary grayscale images and used support vector machine (SVM) as classifier. They obtained 95% accuracy on a dataset comprising 25,000 malware and 12,000 benign samples.

The structure of deep learning (DL) methods for malware analysis and classification, however, is very shallow compared to benchmark CNNs for ImageNet containing a great number of layers. Furthermore, training deep CNNs with a small labeled dataset is not easy and ImageNet CNN models are trained with ten million annotated images in ImageNet dataset. Some have tried to solve this problem via transfer learning techniques by using pre-trained ImageNet CNN models as feature extractors in small datasets in different domains (E. Rezende et al., 2018; Cetinic et al., 2018; Reyes et al., 2015; Kaya et al., 2019; Ng et al., 2015; Chang et al., 2017; Özbulak et al., 2016).

Based on this, an algorithm for malware classification called image-based malware classification using ensemble of CNNs (*IMCEC*) has been developed. It uses static features and combines binary visualization and an ensemble of CNNs, which have been previously trained using an extensive set of natural images ($\geq$ 10 million) to classify malware samples into their related families (Fig. 1). Ensemble learning is a machine learning technique in which higher predictive performance is attained by combining results obtained from several classification models, thus resulting in a stronger classifier.

Unlike conventional machine learning techniques, which use training data to learn one hypothesis (Fraz et al., 2012; Chen et al., 2017; Zhou, 2009; Yan et al., 2018), the method proposed here addresses these challenges employing CNNs on multiclass classification problems using pre-trained CNNs and fine-tuning them for malware images. The presence of several CNNs in our IMCEC makes it possible to extract rich features from visualized malware binaries at a distinctive level. We were able to detect subtle differences among malware families and successfully classify them as distinct. The proposed technique made it possible to obtain more specific generic features learned from natural images for different malware images.

This paper describes a novel IMCEC method, which consists of transferring knowledge from different well-known CNN architectures already trained with ImageNet data and adapting and fine-tuning them to malware images. The features obtained were utilized for training various multiclass classifiers were trained using the transferred features and posterior probabilities were fused to increase the accuracy of the classification of families of unknown malware samples. Finally, based on the Malimg malware benchmark dataset (Table 1), tests for malware family classification were conducted on the IMCEC. The IMCEC classification accuracy reached 99.50%. In addition, IMCEC was able to easily block different obfuscation attacks with an acceptable accuracy of 98.11% for packed samples and 97.59% for salted samples.

Section II reviews the current research on malware detection. Section III describes the materials used in the experiment and the methodology used to develop our IMCEC is outlined in Section IV. Section VI presents and discusses our experimental results, before drawing some conclusions and future work in Section VII.

2. Current methods of malware detection

Here, we review the current methods of malware detection, including static, dynamic, and visualization analyses based on deep learning and machine learning methods.

2.1. Static analysis

In (Schultz et al., 2002), the authors performed a static analysis and developed the first data mining technique for detecting malicious codes using three different static features of malware binaries: string sequences, byte sequences and portable executable (PE) heads. They later employed a rule-based method called Ripper (Cohen, 2014) for DLL and naive Bayesian method as learning algorithm to obtain useful features of byte sequence. They

Table 1
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

* PWS stands for password stealing.

achieved a level of classification accuracy up to 97.11% after feeding malicious codes as input data. The author (Shabtai et al., 2012), tested Opcodes with different n-grams sizes and classifiers. They claimed that the performance of 2-gram Opcode method was better than that of byte n-grams methods. The techniques proposed in (Schultz et al., 2002; Shabtai et al., 2012) were required domain expert knowledge and feature engineering tools, however they failed to achieve the satisfactory level of accuracy.

In (Saxe and Berlin, 2015), developed a technique to distinguish benign ware and malware based on neural network technology. They extracted and used four types of features to calculate DLL import, entropy histogram from binary data and metadata of execution files. These features were transformed into 256 dimensions of vector. They found that TRP result was 95.2% while FPR was 0.1%, however, their approach was required a feature engineering tools. The developed technique does not give enough information about the verity of features of the malware and benign samples, as well as, their achieved accuracy was not enough.

Using n-grams technique (Kolter and Maloof, 2006), extracted bytes from windows binaries and trained several classifiers based on one-vs-rest multi-class classification approach. They then combined the predictions of individual classifiers and obtained a true- and false-positive rates of 0.98 and 0.05, respectively, however, their technique did not consider the overhead time.

2.2. Dynamic analysis

Dynamic analyses generally include behavior-based analysis techniques and API call monitoring (Han et al., 2014). In (Bayer et al., 2009), developed an automatic malware binary cluster method based on malware behavior. The author (Zolkipli and Jantan, 2011), recorded malware samples using two types of security tools such as Amun and Honey- then used two virtual platforms to execute the collected malwares and analyzed their behaviors. In (Imran et al., 2016), used hidden Markov model for malware classification relying on system calls as observed symbols. The author in (Lim and Moon, 2015), analyzed network flow activity to develop a malware identification system. They applied sequence alignment and flow feature clustering for the comparison of character sequences to extract their similarities.

The main limitations of dynamic analysis, especially the sandbox-based solutions, are that some malware can detect and change behavior when running in virtual environments. Therefore, dynamic analysis might not always uncover malicious behavior.

Dynamic analysis is useful for malware binary transformation detection, while the use of the virtual environment is very time intensive. Hence, dynamic analysis does not fulfill the need of most of the practice application.

2.3. Visualization analysis

Some visualization also offers tools for the detection and classification of malwares. The author in (Yoo, 2005), was able to detect and visualize a virus by employing self-organizing map. Using image processing technique, Nataraj et al. (2011) visualized malware binaries into grayscale. Through a machine learning approach such as GIST they extracted features from malware grayscale images and used k-nearest neighbor (KNN) as a classifier. Accuracy reached 97.18% on a dataset containing 9458 malware samples related to 25 different malware families. In subsequent work (L. Nataraj et al., 2011), they compared their approach based on image processing technique with dynamic analysis and achieved almost the same accuracy using dynamic malware analysis. They claimed that their modified technique could address both unpacked and packed malware samples. It assumed that a set of texture features could be enough to present the entire content of malware images.

In (Choi et al., 2017), the authors visualized malware and benign binaries into grayscale images. They trained a model with DL techniques and achieved 95.66% test accuracy for a dataset with 10,000 benign and 2000 malware samples. The proposed system does not give enough information on the structure and characteristics of the malware, and additionally it did not consider the overhead time.

In (Su et al., 2018), focused on detecting malware in IoT environments. They created one-channel gray-scale images from executable binaries then employed light-weight DL techniques to classify them into their related families. They achieved classification accuracy of 94.0% for DDoS malware and goodware, and 81.8% for two leading malware families and goodware. However, their proposed network structure was very shallow, and the samples were limited to 2 malware families.

In (Yajamanam et al., 2018), found that deep learning techniques performed equally well as GIST descriptors for image-based malware classification. However, one of the potential advantages of deep learning is that there is no need for extracting GIST features during training. The proposed technique failed to achieve a certain level of accuracy and did not consider the overhead time.

Recently, transfer learning has been applied to adapt CNNs for classifying malware (E. Rezende et al., 2018; Bhodia et al., 2018; Shaha and Pawar, 2018; Hutt, 2017; Vinayakumar et al., 2019). Transfer learning consists of first training a CNN with a large dataset of well-labeled natural images to learn generic image features, which could be applied to all images. It is then employed to extract these generic features from smaller datasets (Ng et al., 2015) with successful results in several classification studies (Cetinic et al., 2018; Kaya et al., 2019; Özbulak et al., 2016; Li and Hoiem, 2018; Z. Zhou et al., 2017; Shelhamer et al., 2017; Wen, 2019). However, features learned through transfer tend to be those of natural image datasets rather than the subtle characteristics of malware images.

Fine-tuning is a modified transfer learning technique consisting of updating pretrained CNN weights by backpropagation. Fine-tuning is helpful for adapting pretrained CNNs to different datasets. When applied to medical imaging data, fine-tuning was as effective as training a CNN from scratch but it was more robust in relation to the size of the data (Kumar et al., 2017; Z. Zhou et al., 2017; Tajbakhsh et al., Jun. 2016). Fine-tuning has been applied to different classification tasks including image retrieval (Radenovic et al., 2019), gender and age classification, plants classification (Kaya et al., 2019) text classification (Howard and Ruder, 2019), object detection (Sun et al., 2018), re-identification (Wang et al., 2018) and fine art classification (Cetinic et al., 2018).

Different from previous research work, this work has advanced further from a previous work that proposed an ensemble-based deep learning framework for malware detection. We proposed a novel and unified method for multi-class malware classification. Our method utilized ensemble of fine-tuned CNNs previously trained with ImageNet dataset (≥10 million) and used various multiclass classifiers to classify the families of unknown malware samples. Our experimental results proved that IMCEC was highly efficient in the classification of malware families, even in small sample sizes, and was capable of blocking different obfuscation attacks. Our experimental considered both overall accuracy and run-time overhead.

3. Methodology

3.1. Malware datasets

For this project, we used a malware dataset called Malimg (L. Nataraj et al., 2011), which contains 9339 malware images belonging to 25 different malware families. Malimg consists of images and the samples do not require pre-processing before applying image-based analyses; however, binaries corresponding to Malimg images are not readily available. Family breakdown for Malimg dataset is shown in Table 1.

A packed malware dataset consisting of 96 executable binaries, typically found on Virus-share website, was also used. We used a well-known UPX packer to pack these executable malware binaries (Kim et al., 2010). File size was shrunk by 50% after being packed.

As CNNs require images as input, we were able to directly use the images from the Malimg dataset. The only preprocessing involved separating images into training (70%) and validation (30%) sets. For packed malware, we had recent executable malware binaries, which were converted into images by adapting the script used by the authors of (L. Nataraj et al., 2011). It took 0.005121 s per sample in average to convert them from executable malware binaries into malware images.

3.2. Overview of the methods used

Our IMCEC method is depicted in Fig. 1. We used CNN architectures previously trained (initialized) using natural images. Each CNN was then applied in one of the following two ways: 1) as classifiers for the generation of softmax probabilities based on fine-tuned, or 2) as image feature extractors with independent feature vectors concatenated and applied for training multiclass SVMs. Posterior probabilities from softmax classifiers and SVM ensembles were then combined to obtain the families of unknown malware samples.

We employed the following CNN models, each having various capabilities:

(1) Fig. 2 shows VGG16 network architecture (Simonyan and Zisserman, 2014). This well-known CNN has a standard neural network architecture of connected layers and contains 16 layers needing to be trained, 5 convolutional layer blocks and 3 fully-connected layers. The filter size used in the convolutional layers was 3 × 3 kernel with 1 padding and 1 stride to guarantee the same spatial dimension for each activation map as the previous layer. To accelerate training, rectified linear unit (ReLU) nonlinearity was applied to each convolutional layer, and a max-pooling step was used at each block end for decreasing spatial dimensions. A 2 × 2 kernel filter with no padding and 2 strides was used in max-pooling layers to guarantee that the spatial dimensions of activation map were halved compared to previous layer. Then, two fully-connected layers containing 4096 ReLU activated units were employed before the last 1000 fully-connected softmax layer.

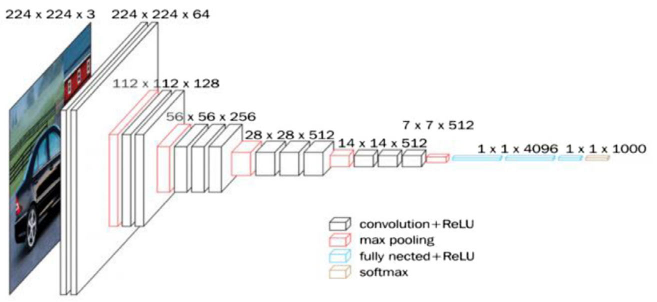


Fig. 2. . VGG16 Network Architecture.

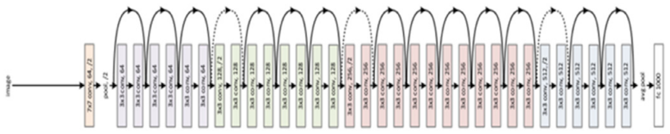


Fig. 3. ResNet-50 Network Architecture.

(2) Residual networks (ResNet-50) (Krizhevsky et al., 2012) 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. 3.

We selected these models because they are well understood and have high performance in different scenarios of malware image classification (E. Rezende et al., 2018; Yue, 2017).

In our ensemble, we have applied two different types of classifiers:

(1) Softmax is a generalized logistic function emphasizing the essential values of vectors while blocking those with values lower than maximum. Softmax function can be applied as a nonlinear version of multinomial logistic regression to a D-dimensional feature vector giving a D probability values vector in which the d^{th} element is the probability by which the vector may represent a d^{th} class member (Bishop, 2006). Several CNN architectures employ Softmax function as classification layer (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016)

(2) Multiclass SVMs (Williams, 2003): Support vector machines (SVM) called support vector networks (Wang and Lin, 2014) are generally employed in machine learning and are of great importance in deep learning processes. SVM is a machine learning classifier capable of classifying the data that have been previously obtained, which relies on assigning specific scores to each data (Karpathy and Li, 2015). Data classification is among the most critical aspects of deep learning. Generally, the resulting data are N-dimensional vectors and for this reason, it is called vector machine. We trained one-vs-all SVMs with extracted knowledge from trained CNNs.

We used TensorFlow (Dean and Monga, 2015), Scikit-learn (Varoquaux et al., 2017) and Keras (Chollet, 2015) library to test our method in python3. The hardware used in this work was 12-GB GPU (GeForce GTX 1080 Ti) for training and Intel Core i7-4790 processor with 10 GB main memory for testing.

3.3. Transfer learning

Transfer learning consists of transferring the parameters of a neural network trained with a single dataset and task to another problem with a different dataset and task (Özbulak et al., 2016; Long et al., 2018). Many deep neural networks trained on natural images share an interesting phenomenon in common: on the first layers, they learn general features, that is, features that do not appear to be specific to a particular dataset or task, but are applicable to many datasets and tasks. When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable the training of a large target network without overfitting.

Next, we used VGG16 and ResNet-50 as base models, which were pre-trained for detecting objects on ImageNet dataset. We employed the convolutional layers of VGG16 and ResNet-50 to obtain malware image bottleneck features, which then served as input in training SVM classifiers.

3.4. CNN fine-tuning

CNN architectures previously trained on ImageNet (Simonyan and Zisserman, 2014), a dataset consisting of 1000 classes, were customized to our problem by using a fully-connected layer containing 25 classes (25 malware families) instead of a final fully connected layer (intended for 1000 classes). Initial weights of pretrained CNNs of natural images were used then fine-tuned techniques was applied to optimize via back-propagation.

Assuming that X is a training dataset with n malware images, fine-tuning is an iterative procedure optimizing w filter weights and reducing error rate, according to the following:

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

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

Mini-batch stochastic gradient descent was used for obtaining optimal w, assuming that $B \subset X$ is a subset of b images where B is a mini-batch of X with size b, 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

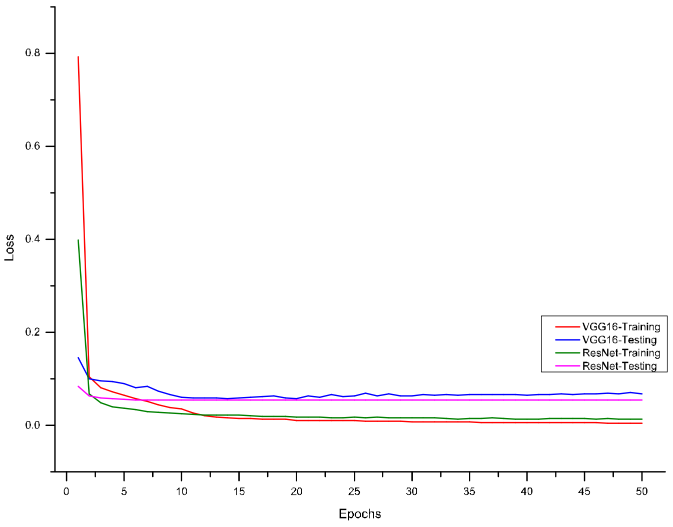


Fig. 4. Training and testing loss for both fine-tuned CNNs.

weights w_{t+1} were obtained based on loss L gradient when they were employed for mini-batch B with present weights w_t

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

where $\Delta wt = wt - wt{-}1$ 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 the learning process while instantaneously leveling the update of weights. Weight decay λ reduces weights for obtaining the lowest optimum weights.

All models needed memory for storing filter weights w and therefore, batch size b depended on the memory capacity of the training hardware. Batch size was set at $b = 32$ and learning rate was $\eta = 5 \times 10^{-6}$, which is the uniform learning rate that makes it possible for fine-tuning procedure to efficiently learn filter weights in different CNN models. We obtained this value experimentally through monitoring validation errors during fine-tuning via different learning rates. The learning rate was 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 when the learning process might not be globally optimal and serious variations could be more acceptable. We adopted $\alpha = 0.9$ 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.

Figs. 4 and 5 show the training, testing accuracy and loss when fine-tuning the proposed CNN models across 50 epochs. One can see that after 20 epochs, both accuracy and loss were stabilized in both networks. The similarity of testing and training curves suggests that the overfitting of training data was prevented in the pro-

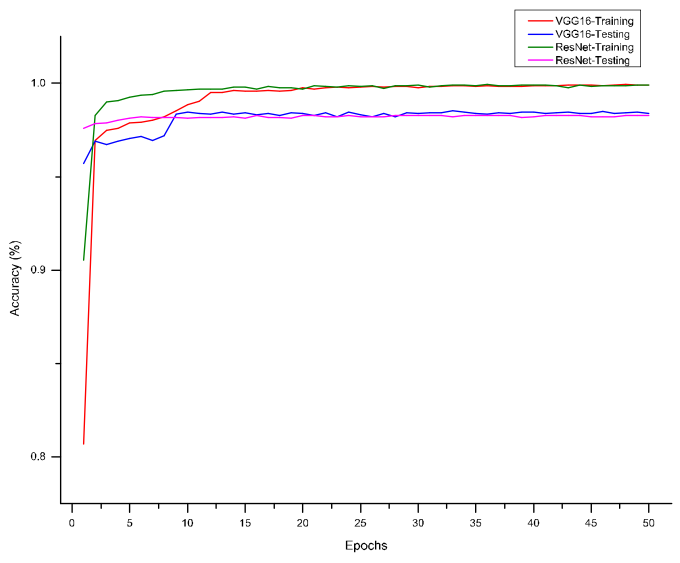


Fig. 5. Training and testing accuracy for both fine-tuned CNNs.

Table 2
Fine-tuning statistics.

Architecture	Parameters
VGG16	134,362,969
ResNet-50	23,585,817

posed fine-tuned models. Training statistics of all models are summarized in Table 2, the number of weights requiring fine-tuning over 50 epochs.

3.5. IMCEC design

Our proposed IMCEC design contains the following classifiers.

(1) Fine-tuned VGG16 employing a Softmax classifier.
(2) Fine-tuned ResNet50 employing a Softmax classifier.
(3) A one-vs-all multiclass SVM, which was trained based on the features obtained from pre-trained VGG16 network. We extracted 4096 features from the fully connected (FC2) layer of the pre-trained network. We then applied principal component analysis (PCA) (Maćkiewicz and Ratajczak, 1993) technique to decrease the dimensionality of the obtained features for efficient classifier training. Our feature vectors were the components explaining 90% of data variation (dimensionality from 4096 to 410).
(4) A one-vs-all multiclass SVM which was trained based on the features obtained from the pre-trained ResNet50 network. We extracted 2048 features from the last flatten (avg_pool) layer of the pre-trained network. We then applied PCA technique to decrease the dimensionality of the obtained features for efficient classifier training. We used PCA so that the proposed feature vectors comprised principal components explaining 90% of data variation (dimensionality from 2048 to 205).
(5) A one-vs-all multiclass SVM which was trained based on the features obtained from both pre-trained VGG16 and ResNet-50 networks. We extracted features of all CNNs and concatenated them into a 6144-dimensional vector. We then applied PCA to reduce the feature vectors for efficient classifier training. Our feature vectors were the principal components explaining 90% of data variation (dimensionality from 6144 to 615).

The multiclass SVMs employed in the proposed IMCEC were trained based on PCA-reduced features extracted from pre-trained CNNs based on training dataset. During classification, we obtained posterior probabilities $P_{i,\,k}(f)$where the i^{th} test image presented a certain family f based on the k^{th} classifier. We obtained image family f^{*} by integrating posterior probabilities as"

$$f^{*} = argmax\frac{\sum_{k}^{C} P_{i,\ k}\ (f)}{C} \tag{3}$$

Where $C = 5$ is IMCEC classifier number.

4. Experimental results and discussion

4.1. Experimental setup

Our experiment compared the performance of our IMCEC method with several well-established CNN architectures:

(1) Transfer learned from pretrained CNNs with multiclass SVMs as a classifier.
(2) Fine-tuned pretrained CNNs with Softmax as a classifier.
(3) Fine-tuned pretrained CNNs with multiclass SVMs as a classifier.

For all these baselines, we have used VGG16 and ResNet-50 architectures. We used accuracy, f1-score, recall, precision, false positive and receiver operating characteristics curve (ROC) for performance evaluation. These evaluation metrics have been extensively used in research community to provide detailed assessments of methods (Ni et al., 2018; Saxe and Berlin, 2015; Namanya et al., 2019).

- True Positive (TP): The true category is positive, predicted category is positive.
- True Negative (TN): The true category is negative, predicted category is negative.
- False Positive (FP): The true category is negative, predicted category is positive.
- False Negative (FN): The true category is positive, predicted category is negative.

The proposed system was evaluated using the following criteria, based on the above factors:

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 the proportion of positive predictions that are actually correct; it is calculated by dividing the number of true positives by all positive predictions:

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

Recall is the model's ability to correctly detect all the potential malware; it is calculated by dividing the 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).

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

Table 3
Classification Accuracy (%).

	Architecture	
	VGG16	ResNet-50
TL+SVM	98.62	98.70
FT+Softmax	98.40	98.26
FT+SVM	98.76	97.23
Our IMCEC	99.50	

FT = Fine-Tuned; TL = Transfer-Learned.

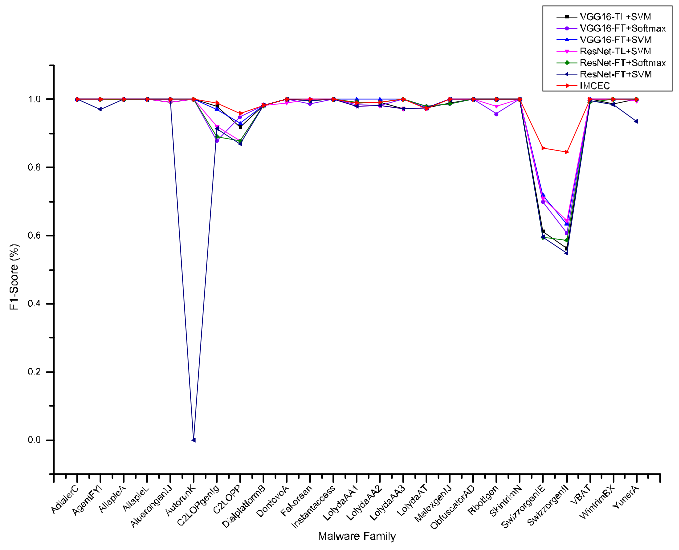


Fig. 6. F1 Score of different malware classification methods including VGG16-TL-SVM, VGG16-FT-Softmax, VGG16-FT-SVM, ResNet50-TL-SVM, ResNet50-FT-Softmax, ResNet50-FT-SVM, IMCEC in 25 malware families.

Receiver operating characteristic (ROC) curves show the performance of a classification model at all classification thresholds. These curves plot two parameters: true positive rate (TPR) similar to recall and false positive rate (FPR), which can be expressed as Eq. (8).

$$TPR = \frac{TP}{TP + FN} \text{ and } FPR = \frac{FP}{FP + TN} \quad (8)$$

We analyzed the capability of IMCEC with others baseline methods in classifying individual families. We obtained accuracy, precision, recall, F1-metrics, false positive and ROC curve for every malware family in the dataset.

We also compared the performance of the proposed IMCEC method with the existing machine and deep learning approaches mostly using "Malimg" benchmark dataset. It must be kept in mind that a variety of existing approaches used different data balancing techniques to boost the performance of the networks (Cui et al., 2019; Cui et al., 2018), while we used the imbalanced of malware family's malware dataset.

4.2. Results

Under the experimental conditions described above, our IMCEC recorded 99.50% accuracy in classification. Table 3 summarizes the classification accuracies of the proposed IMCEC method and baseline CNNs. It clearly shows that the developed method outperformed all other methods in terms of classification accuracy. The accuracy of the proposed method was 0.74 percentage points higher than that of the baseline method (fine-tuned VGG16 with SVM classifier).

Figs. 6–10 show the accuracy, precision, recall rate, F1-score and false positive rate of seven methods. The ROC curve and prob-

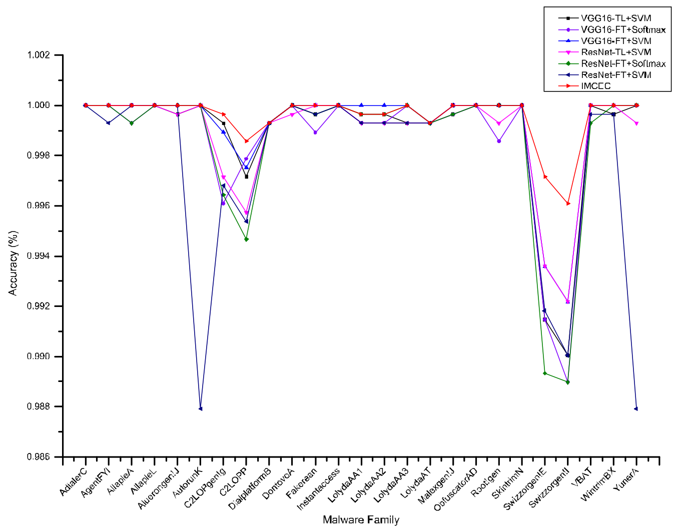


Fig. 7. Accuracy of different malware classification methods including VGG16-TL-SVM, VGG16-FT-Softmax, VGG16-FT-SVM, ResNet50-TL-SVM, ResNet50-FT-Softmax, ResNet50-FT-SVM, IMCEC in 25 malware families.

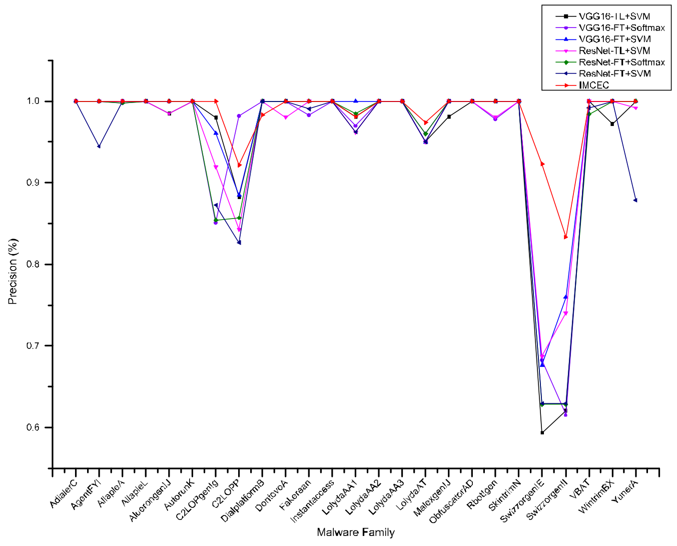


Fig. 8. Precision of different malware classification methods including VGG16-TL-SVM, VGG16-FT-Softmax, VGG16-FT-SVM, ResNet50-TL-SVM, ResNet50-FT-Softmax, ResNet50-FT-SVM, IMCEC in 25 malware families.

ability matrix of each method are presented in Fig. 12. Experimental results showed that our proposed IMCEC had higher accuracy (99.50%) than VGG16-TL+SVM (98.65%), VGG16-FT+Softmax (98.35%), VGG16-FT+SVM (98.85%), ResNet50-TL+SVM (98.60%), ResNet50-FT+Softmax (98.20%) and ResNet50-FT+SVM (97.19%). Besides, IMCEC gave fewer false positive than all other methods. Since Swizzor.gen! I and Swizzor.gen! E families are quite similar, the majority of methods are not able to distinguish them accurately; IMCEC algorithm, however, retained its high performance. As shown in Fig. 12, the same conclusion can be drawn regarding the ROC curve: IMCEC curve is closer to the upper left corner of the coordinates. Even with Swizzor.gen!E and Swizzor.gen!I families, the curve peak from IMCEC remained very close to 1, suggesting IMCEC performs well even in uneven data distributions.

Table 4 compares our IMCEC with existing malware classification methods based on machine learning and deep learning techniques. Some of the methods (indicated with an asterisk (*)) used Malimg dataset; others (indicated with double asterisk (**)) used

Table 4
A comparative summary.

Method	Precision (%)	Recall (%)	Accuracy (%)
GIST + SVM* (E. Rezende et al., 2018)	–	–	89.43
Byte 1-gram* (E. Rezende et al., 2018)	–	–	83.68
VGG16 + Softmax* (E. Rezende et al., 2018)	–	–	90.77
VGG16 + SVM* (E. Rezende et al., 2018)	–	–	92.29
ResNet+ Softmax* (E. Rezende et al., 2018)	–	–	98.62
Google + Softmax (Khan et al., 2018)	–	–	98.0
BAT** (Cui et al., 2018)	94.60	94.50	94.50
GLCM + SVM** (Cui et al., 2018)	93.40	93.0	93.2
GIST + KNN** (Cui et al., 2018)	92.70	92.30	92.50
Deep Learning* (Bhodia et al., 2018)	–	–	94.80
Light-weight DL (Su et al., 2018)	–	–	94.00
DL vs Gist Descriptor* (Yajamanam et al., 2018)	–	–	95.66
NSGA-II** (Cui et al., 2019)	–	88.40	97.60
Our IMCEC	99.50	99.46	99.50

* denotes methods that used Malimg dataset.
** denotes methods that used Malimg dataset with different data augmentation techniques.

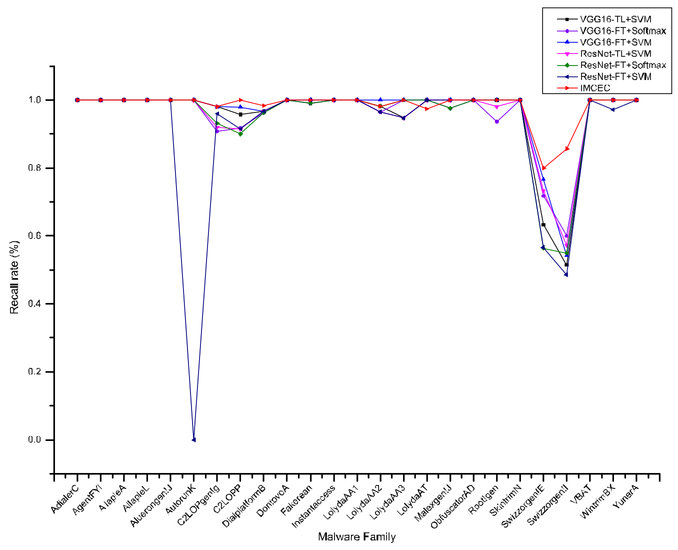


Fig. 9. Recall rate of different malware classification methods including VGG16-TL-SVM, VGG16-FT-Softmax, VGG16-FT-SVM, ResNet50-TL-SVM, ResNet50-FT-Softmax, ResNet50-FT-SVM, IMCEC in 25 malware families.

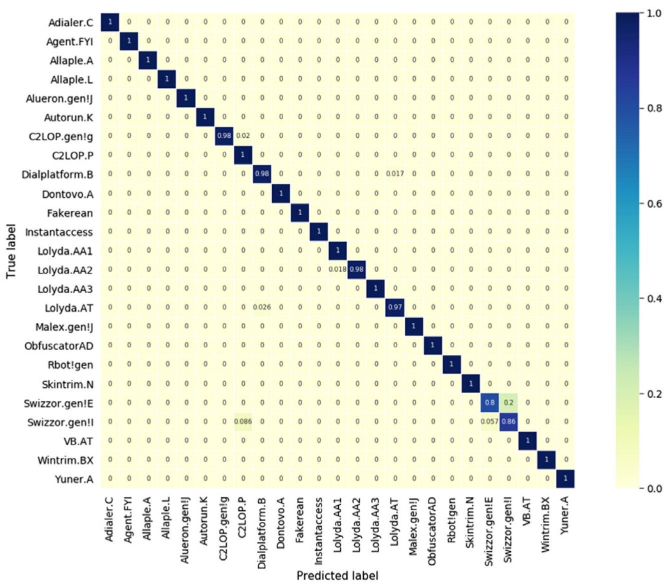


Fig. 11. A confusion matrix for IMCEC. The matrix entries have been scaled to a percentage to account for the uneven distribution of the malware families.

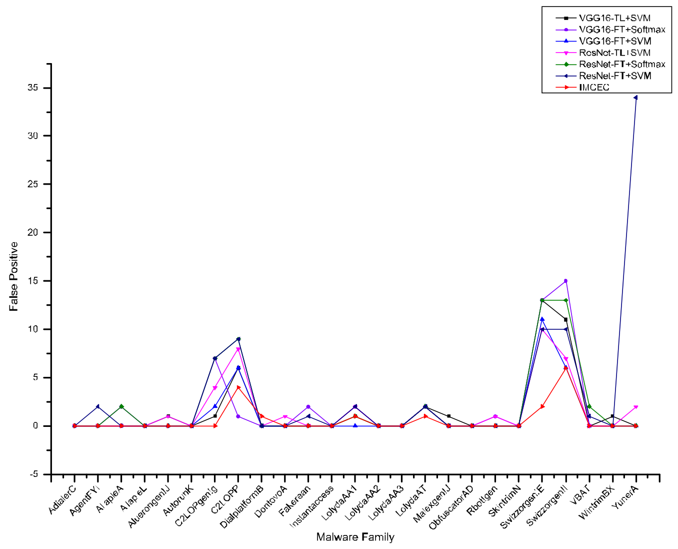


Fig. 10. False positive rate of different malware classification methods including VGG16-TL-SVM, VGG16-FT-Softmax, VGG16-FT-SVM, ResNet50-TL-SVM, ResNet50-FT-Softmax, ResNet50-FT-SVM, IMCEC in 25 malware families.

Table 5
Average image generation and prediction time (s) per sample.
FT = Fine-Tuned; TL = Transfer-Learned.

	Image Generation	Architecture	
		VGG16 Prediction	ResNet-50 Prediction
TL+SVM	0.005121	0.2017	0.1828
FT+Softmax		0.2038	0.1874
FT+SVM		0.2020	0.1830
Our IMCEC		1.18	

different techniques to expand and balance the imbalance of families in Malimg dataset during network training. Our method did not use any data balancing technique and achieved a higher classification accuracy rate than all other methods that used expanded and unexpanded datasets.

Table 5 presents the average image generation time by adopting the script used in (L. Nataraj et al., 2011) and predicting a malware image. Prediction time varied depending on the network. The classification of a new malware sample consists of two stages: generating a malware image and predicting it by CNN trained network. Experimental findings revealed that the average time required for the identification of a malware sample by IMCEC was 1.18 s.

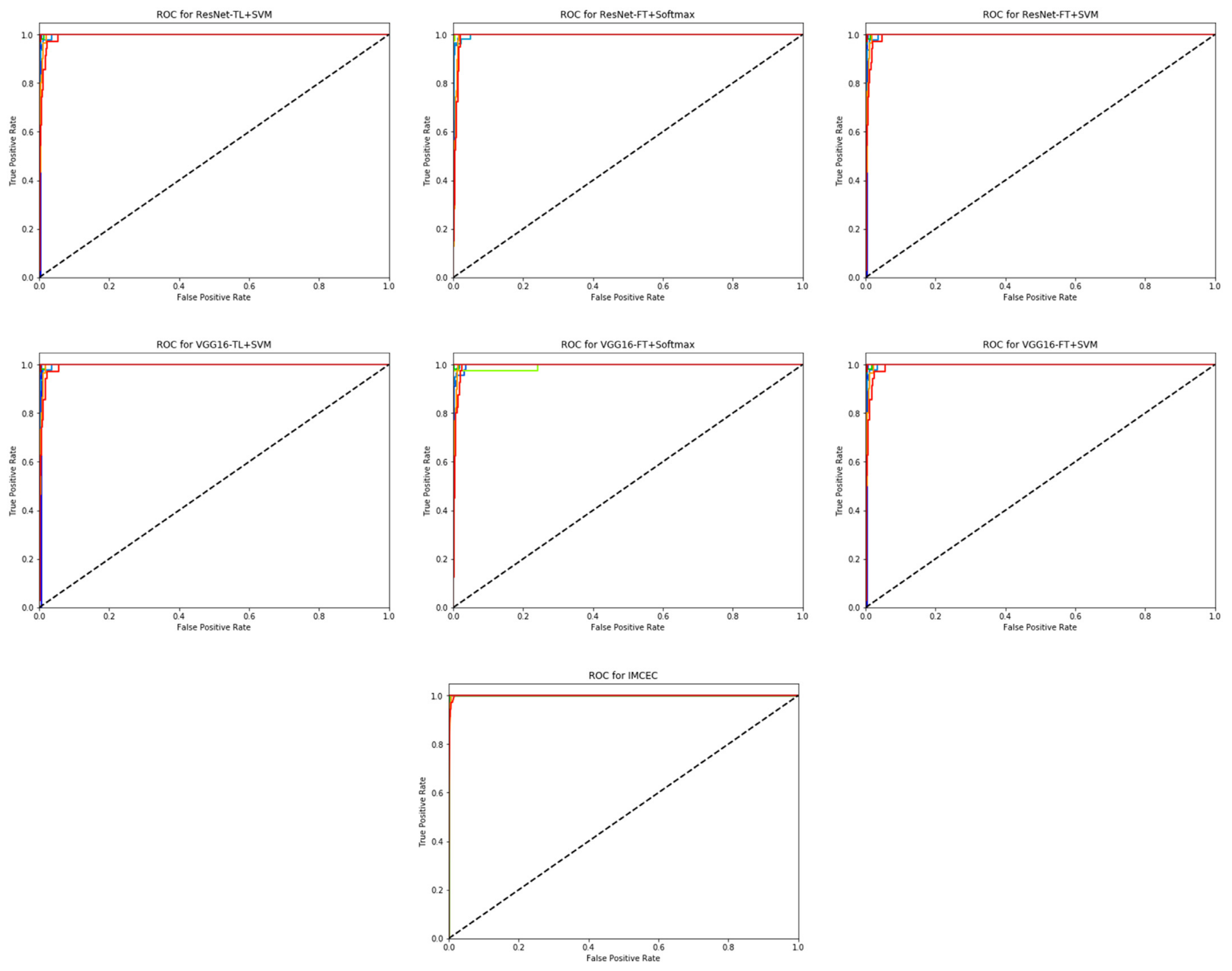


Fig. 12. ROC curves of different malware families with different algorithms including VGG16-TL+SVM, VGG16-FT+Softmax, VGG16-FT+SVM, ResNet50-TL+SVM, ResNet50-Ft+Softmax, ResNet-FT+SVM, (VGG16+ResNet50)-TL+SVM, IMCEC in 25 malware families.

Fig. 11 shows the detection accuracy of IMCEC in the classification of individual families using heatmap of confusion matrix. Due to the imbalance of malware families in test dataset, a confusion matrix was generated by scaling classification counts to a percentage.

4.3. Robustness experiments

We also tested different obfuscation attacks, that is, attacks with the aim of degrading the effectiveness of IMCEC-based score. First, we conducted experiments on packed malware samples using our packed malware dataset consisting of 96 packed malware samples, as described in Section III-A. We obtained an overall classification accuracy of 98.11%, which represented an insignificant decrease compared with the original (unpacked malware) accuracy of 99.24% (Table 6).

We then salted all the malware families in the Malimg dataset with benign samples like the ones typically found on recent Windows PCs. The same method has been previously applied to block several machine learning- and statistical-based scores (Yajamanam et al., 2018; Desai and Stamp, 2011; Singh et al., 2016; Lin and Stamp, 2011). In this test, the overall classification accuracy decreased slightly to 97.59% compared with the original accuracy of 99.50% (Table 7).

Overall, this experiment indicated that IMCEC-based scoring was reasonably robust against these specific obfuscation attacks.

4.4. Discussion

Our results showed that the IMCEC method we have developed remains superior to all other current baselines. This is due to two features of our method: 1) the application of a variety of deep network structures with various abilities in simplifying and modifying distinctive data, and 2) our ensemble of fine-tuning scheme for learning the precise features our dataset.

Despite the disparity between malware images and natural images, the structures obtained through transfer learning resulted in generic features, which could then be applied to all images. This resulted in good classification accuracy because the acquired knowledge was adaptable to malware image recognition. ResNet-50, which was already more accurate than VGG16 in ImageNet classification challenge (Bianco et al., 2018), had relatively higher accuracy when transferring the knowledge to malware images. This positive outcome supports the use of deeper networks, like ResNet-50. In contrast, networks with lower deepness, like VGG16,

Table 6
Classification accuracy of packed and unpacked malware samples (%).

	Architecture			
	VGG16 (Unpacked)	ResNet-50 (Unpacked)	VGG16 (Packed)	ResNet-50 (Packed)
TL+SVM	98.10	97.20	97.59	97.29
FT+Softmax	98.36	95.23	96.55	88
FT+SVM	98.11	98.11	97.29	97.29
Our IMCEC	99.24		98.11	

FT = Fine-Tuned; TL = Transfer-Learned.

Table 7
Classification accuracy for salted and unsalted Malimg malware dataset (%).

	Architecture			
	VGG16 (Unsalted)	ResNet-50 (Unsalted)	VGG16 (Salted)	ResNet-50 (Salted)
TL+SVM	98.62	98.70	95.76	95.79
FT+Softmax	98.40	98.26	95.39	95.94
FT+SVM	98.90	97.23	95.96	95.10
Our IMCEC	99.50		97.59	

FT = Fine-Tuned; TL = Transfer-Learned.

learn features with less semantic significance when transferred to other domains. Because transfer learned ResNet-50 features are semantically more enhanced for natural images compared with VGG16 network, they are more adaptable and generalizable when transferred to malware image classification problem. The transfer learning conclusions exhibited the strengths and characteristics of a variety of network structures.

After fine-tuning the CNN with Softmax classification, the standard for most natural image classifications gave significant results. The accuracy of ResNet-50 dropped by 0.44 percentage points compared to that of transfer learned ResNet-50 and the classification accuracy of VGG16 dropped by 0.22 percentage points compared to transfer learned VGG16. This happened because Softmax classifier did not perform a one-vs-all comparison in the classification of images and malware families with no sharp differences (i.e., Swizzor.gen! E and Swizzor.gen! I) could be misclassified. One-vs-all multiclass SVMs can distinguish between malware family with subtle differences.

The confusion matrix (Fig. 11) reveals that IMCEC is capable of classifying many malware families correctly. It also indicates that misclassification generally occurred within families (e.g., Swizzor.gen! I and Swizzor.gen! E families).

Our method is able to distinguish subtle variations among malware families because it combines the generalizability of both VGG16 and ResNet-50 networks. This is why the classification accuracy achieved by IMCEC remained higher than the other baseline methods (Table 3). IMCEC combines the advantages of individual techniques; therefore, it was able to classify images, which were misclassified by more specific methods.

Our IMCEC method also performed better than other existing methods because it uses handcrafted features (Pascanu et al., 2015) as well as those of VGG16 and ResNet-50 (E. Rezende et al., 2018; He et al., 2016; E. Rezende et al., 2018; Khan et al., 2018) (Table 4). In particular, the CNNs in our IMCEC were fine-tuned only for 50 epochs and produced higher accuracy than methods that used separate CNNs refined for (100–2000) epochs. Our ensemble of CNN architectures was capable of creating an accurate classifier, which was not fine-tuned for a large number of epochs.

Additionally, we performed extensive experiments to test our IMCEC against different obfuscation attacks. We found that IMCEC was able to handle all obfuscation attacks and still perform better than other methods (Tables 6 and 7).

CNN fine-tuning strengthens the capacity of IMCEC to learn the essential features that are particularly relevant to the dataset being classified. This capability to learn the most significant dynamic features from the image suggests that the IMCEC method can adapt itself to different malware datasets.

Our study could be extended by the addition or integration of deeper CNN architectures with new capabilities. For example, Inception-ResNet-V2 (Szegedy et al., 2017) merged with GoogLeNet and ResNet has achieved high accuracy on ImageNet classification. We believe that this deeper network may improve the performance of our ensemble. This is left for future work because more in-depth network training requires more computational cost and significantly longer time, which is currently out of our reach.

5. Conclusion and future work

Modern anti-malware solutions rely on ML techniques to protect digital assets from malware. While ML-based methods, have demonstrated effectiveness at detecting new malware, but they also come with substantial development costs. Creating a large set of useful features for the ML techniques takes significant amounts of time and expertise from both of malware analysts and data scientists. Deep learning architectures, in particular convolutional neural networks (CNNs), have demonstrated great performance in detecting malware simply by looking at the raw bytes such as Windows Portable Executable (PE) files.

We developed and tested a novel image-based Malware classification using an ensemble of CNN architectures (IMCEC) . Our IMCEC classified the majority of malware samples correctly even under obfuscation attacks and overall performed better than existing methods employing similar benchmarks. Our experiments have demonstrated great levels of accuracy that are competitive with traditional ML-based solutions, while avoiding the manual feature engineering phase. Our IMCEC is flexible, practical and efficient as it takes only 1.18 s on average to identify new malware samples.

In the future, we plan to evaluate IMCEC against larger datasets and implementing a prototype of the proposed approach in a real-world environment for evaluation and refinement, future research also would focus on time reduction.

Declaration of Competing Interest

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

Reference

Alazab, M., 2015. Profiling and classifying the behavior of malicious codes. J. Syst. Softw. doi:10.1016/j.jss.2014.10.031.

Azmoodeh, A., Dehghantanha, A., Conti, M., Choo, K.K.R., 2018. Detecting crypto-ransomware in IOT networks based on energy consumption footprint. J. Ambient Intell. Humaniz. Comput. doi:10.1007/s12652-017-0558-5.

Bayer, U., Milani-Comparetti, P., Hlauscheck, C., Kruegel, C., Kirda, E., 2009. Scalable, behavior-based malware clustering. 16th Symp. Netw. Distrib. Syst. Secur..

N. Bhodia, P. Prajapati, F. Di Troia, and M. Stamp, "Transfer learning for image-based malware classification," 2018.

Beek, C., et al., 2017. McAfee labs threats report: April 2017. McAfee Labs Rep. 1–49.

Bianco, S., Cadene, R., Celona, L., Napoletano, P., 2018. Benchmark analysis of representative deep neural network architectures. IEEE Access doi:10.1109/ACCESS.2018.2877890.

Bishop, C.M., 2006. Pattern recognition and machine learning (Information science and statistics), 1st edn. 2006. corr. 2nd printing edn. Mach. Learn..

Cetinic, E., Lipic, T., Grgic, S., 2018. Fine-tuning convolutional neural networks for fine art classification. Expert Syst. Appl. 114, 107–118. doi:10.1016/j.eswa.2018.07.026.

Chang, J., Yu, J., Han, T., Chang, H.J., Park, E., 2017. 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 doi:10.1109/HealthCom.2017.8210843.

Chen, J., Wang, Y., Wu, Y., Cai, C., 2017. An ensemble of convolutional neural networks for image classification based on LSTM. Proc. - 2017 Int. Conf. Green Informatics, ICGI 2017 21 (1), 217–222. doi:10.1109/ICGI.2017.36.

Choi, S., Jang, S., Kim, Y., Kim, J., 2017. Malware detection using malware image and deep learning. In: International Conference on Information and Communication Technology Convergence: ICT Convergence Technologies Leading the Fourth Industrial Revolution, ICTC 2017, pp. 1193–1195. doi:10.1109/ICTC.2017.8190895 2017-December.

Chollet, F.F., 2015. Keras: deep learning library for theano and tensorflow. GitHub Repos..

Cohen, W.W., 2014. Fast effective rule induction. In: Machine Learning Proceedings 1995.

Conti, G., Dean, E., Sinda, M., Sangster, B., 2008. Visual reverse engineering of binary and data files. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) doi:10.1007/978-3-540-85933-8_1.

Cui, Z., Du, L., Wang, P., Cai, X., Zhang, W., Jul. 2019. Malicious code detection based on CNNs and multi-objective algorithm. J. Parallel Distrib. Comput. 129, 50–58. doi:10.1016/j.jpdc.2019.03.010.

Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G., Chen, J., Jul. 2018. Detection of malicious code variants based on deep learning. IEEE Trans. Ind. Informatics 14 (7), 3187–3196. doi:10.1109/TII.2018.2822680.

Damodaran, A., Di Troia, F., Visaggio, C.A., Austin, T.H., Stamp, M., 2017. A comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hacking Tech. doi:10.1007/s11416-015-0261-z.

Dean, J., Monga, R., 2015. TensorFlow - Google's latest machine learning system, open sourced for everyone. Google Res. Blog.

Desai, P., Stamp, M., 2011. A highly metamorphic virus generator. Int. J. Multimed. Intell. Secur. doi:10.1504/ijmis.2010.039240.

Farivar, F., Haghighi, M.S., Member, S., 2019. Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber physical systems and industrial IOT. IEEE Trans. Ind. Informatics PP (c), 1. doi:10.1109/TII.2019.2956474.

Fraz, M.M., et al., 2012. An ensemble classification-based approach applied to retinal blood vessel segmentation. IEEE Trans. Biomed. Eng. doi:10.1109/TBME.2012.2205687.

Han, K., Kang, B., Im, E.G., 2014. Malware analysis using visualized image matrices. Sci. World J. doi:10.1155/2014/132713.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition doi:10.1109/CVPR.2016.90.

J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2019, doi: 10.18653/v1/p18-1031.

Huda, S., Abawajy, J., Alazab, M., Abdollalihian, M., Islam, R., Yearwood, J., 2016. Hybrids of support vector machine wrapper and filter based framework for malware detection. Futur. Gener. Comput. Syst. doi:10.1016/j.future.2014.06.001.

Hutt, S., 2017. Deep Learning for Cyber Security.

Imran, M., Afzal, M.T., Qadir, M.A., 2016. Using hidden markov model for dynamic malware analysis: first impressions. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015 doi:10.1109/FSKD.2015.7382048.

Jung, B., Il Bae, S., Choi, C., Im, E.G., 2020. Packer identification method based on byte sequences. Concurr. Comput. Pract. Exp. doi:10.1002/cpe.5082.

Kancherla, K., Mukkamala, S., 2013. 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 doi:10.1109/CICYBS.2013.6597204.

A. Karpathy and F.-F. Li, "Convolutional neural networks for visual recognition," *Available in http://cs231n.github.io/convolutional-networks*, 2015.

Kaya, A., Keceli, A.S., Catal, C., Yalic, H.Y., Temucin, H., Tekinerdogan, B., 2019. Analysis of transfer learning for deep neural network based plant classification models. Comput. Electron. Agric. 158 (January), 20–29. doi:10.1016/j.compag.2019.01.041.

Khan, R.U., Zhang, X., Kumar, R., 2018. Analysis of resnet and googlenet models for malware detection. J. Comput. Virol. Hacking Tech..

Kim, M.J., et al., 2010. Design and performance evaluation of binary code packing for protecting embedded software against reverse engineering. ISORC 2010 - 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing doi:10.1109/ISORC.2010.23.

Kolter, J., Maloof, M., 2006. Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res..

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. ImageNet Classification with Deep Convolutional Neural Networks.

Kumar, A., Kim, J., Lyndon, D., Fulham, M., Feng, D., 2017. An ensemble of fine-tuned convolutional neural networks for medical image classification. IEEE J. Biomed. Heal. Inform. doi:10.1109/JBHI.2016.2635663.

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, doi: 10.1145/3194452.3194459.

Lab, Kaspersky, 2016. What is a keylogger? | Definition | Kaspersky lab US. Kaspersky Lab.

Li, Z., Hoiem, D., 2018. Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. doi:10.1109/TPAMI.2017.2773081.

Lim, H., Moon, S.J., 2015. Stable nonpolar solvent droplet generation using a poly(dimethylsiloxane) microfluidic channel coated with poly-p-xylylene for a nanoparticle growth. Biomed. Microdevices doi:10.1007/s10544-015-9974-5.

Lin, D., Stamp, M., 2011. Hunting for undetectable metamorphic viruses. J. Comput. Virol. doi:10.1007/s11416-010-0148-y.

Lindorfer, M., Neugschwandtner, M., Platzer, C., 2015. MARVIN: efficient and comprehensive mobile app classification through static and dynamic analysis. In: Proceedings - International Computer Software and Applications Conference doi:10.1109/COMPSAC.2015.103.

Long, M., Cao, Y., Cao, Z., Wang, J., Jordan, M.I., 2018. Transferable representation learning with deep adaptation networks. IEEE Trans. Pattern Anal. Mach. Intell..

Maćkiewicz, A., Ratajczak, W., 1993. Principal components analysis (PCA). Comput. Geosci. doi:10.1016/0098-3004(93)90090-R.

Mohamed Shakeel, P., Baskar, S., Sarma Dhulipala, V.R., Mishra, S., Jaber, M.M., 2018. Maintaining security and privacy in health care system using learning based deep-q-networks. J. Med. Syst. doi:10.1007/s10916-018-1045-z.

Namanya, A.P., Awan, I.U., Disso, J.P., Younas, M., 2019. Similarity hash based scoring of portable executable files for efficient malware detection in iot. Futur. Gener. Comput. Syst. doi:10.1016/j.future.2019.04.044.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011a. Malware images: visualization and automatic classification. Proc. 8th Int. Symp. Vis. Cyber Secur. 4. doi:10.1145/2016904.2016908.

Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J., 2011b. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: Proceedings of the ACM Conference on Computer and Communications Security doi:10.1145/2046684.2046689.

Ng, H.-.W., Nguyen, V.D., Vonikakis, V., Winkler, S., 2015. Deep learning for emotion recognition on small datasets using transfer learning. In: Proceedings of the 2015 ACM on International Conference on Multimodal Interaction - ICMI '15 doi:10.1145/2818346.2830593.

Ni, S., Qian, Q., Zhang, R., 2018. Malware identification using visualization images and deep learning. Comput. Secur. 77 (August), 871–885. doi:10.1016/j.cose.2018.04.005.

Özbulak, G., Aytar, Y., Ekenel, H.K., 2016. How transferable are CNN-based features for age and gender classification? In: Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) doi:10.1109/BIOSIG.2016.7736925.

Pascanu, R., Stokes, J.W., Sanossian, H., Marinescu, M., Thomas, A., 2015. Malware classification with recurrent networks. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings doi:10.1109/ICASSP.2015.7178304.

Radenovic, F., Tolias, G., Chum, O., 2019. Fine-Tuning CNN image retrieval with no human annotation. IEEE Trans. Pattern Anal. Mach. Intell. doi:10.1109/TPAMI.2018.2846566.

Reyes, A.K., Caicedo, J.C., Camargo, J.E., 2015. Fine-tuning deep convolutional networks for plant recognition. In: CEUR Workshop Proceedings.

Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., De Geus, P., 2018b. 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 doi:10.1109/ICMLA.2017.00-19.

Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F., de Geus, P., 2018a. Malicious software classification using VGG16 deep neural network's bottleneck features. Advances in Intelligent Systems and Computing doi:10.1007/978-3-319-77028-4_9.

Rieck, K., Trinius, P., Willems, C., Holz, T., 2011. Automatic analysis of malware behavior using machine learning. J. Comput. Secur. doi:10.3233/JCS-2010-0410.

Saxe, J., Berlin, K., 2015. Deep neural network based malware detection using two dimensional binary program features. In: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), pp. 11–20. doi:10.1109/MALWARE.2015.7413680.

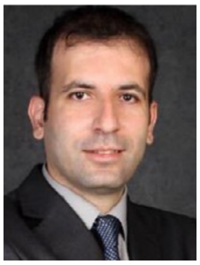
Schultz, M.G., Eskin, E., Zadok, F., Stolfo, S.J., 2002. Data mining methods for detection of new malicious executables. In: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, pp. 38–49. doi:10.1109/SECPRI.2001.924286.

Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., Elovici, Y., 2012. Detecting unknown malicious code by applying classification techniques on OpCode patterns. Secur. Inform. doi:10.1186/2190-8532-1-1.
Shaha, M., Pawar, M., 2018. Transfer learning for image classification. In: Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, pp. 656–660. doi:10.1109/ICECA.2018.8474802.
Shelhamer, E., Long, J., Darrell, T., 2017. Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. doi:10.1109/TPAMI.2016.2572683.
Shijo, P.V., Salim, A., 2015. Integrated static and dynamic analysis for malware detection. Procedia. Comput. Sci. doi:10.1016/j.procs.2015.02.149.
K. Simonyan and A. Zisserman, "VGG-16," *arXiv Prepr.*, 2014, doi: 10.1016/j.infsof.2008.09.005.
Singh, T., Di Troia, F., Corrado, V.A., Austin, T.H., Stamp, M., 2016. Support vector machines and malware detection. J. Comput. Virol. Hacking Tech. doi:10.1007/s11416-015-0252-0.
Su, J., Danilo Vasconcellos, V., Prasad, S., Daniele, S., Feng, Y., Sakurai, K., 2018. Lightweight classification of IOT malware based on image recognition. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2, pp. 664–669. doi:10.1109/COMPSAC.2018.10315.
Sun, S., Yin, Y., Wang, X., Xu, D., Wu, W., Gu, Q., 2018. Fast object detection based on binary deep convolution neural networks. CAAI Trans. Intell. Technol. doi:10.1049/trit.2018.1026.
Szegedy, C., et al., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition doi:10.1109/CVPR.2015.7298594.
Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A., 2017. the impact of residual connections on learning. AAAI Conference on Artificial Intelligence.
Tajbakhsh, N., et al., Jun. 2016. Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans. Med. Imaging 35 (5), 1299–1312. doi:10.1109/TMI.2016.2535302.
Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., Mueller, A., 2017. Scikit-learn. GetMobile Mob. Comput. Commun. doi:10.1145/2786984.2786995.
Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access doi:10.1109/ACCESS.2019.2895334.
Wang, K., Wang, H., Liu, M., Xing, X., Han, T., 2018. Survey on person re-identification based on deep learning. CAAI Trans. Intell. Technol. doi:10.1049/trit.2018.1001.
Wang, P.W., Lin, C.J., 2014. Support vector machines. Data Classification: Algorithms and Applications.
Wen, L., 2019. A transfer convolutional neural network for fault diagnosis based on resnet-50. Neural Comput. Appl. 0123456789. doi:10.1007/s00521-019-04097-w.
Williams, C.K.I., 2003. Learning with kernels: support vector machines, regularization, optimization, and beyond. J. Am. Stat. Assoc. doi:10.1198/jasa.2003.s269.
Yajamanam, S., Selvin, V.R.S., Di Troia, F., Stamp, M., 2018. Deep learning versus gist descriptors for image-based malware classification. Icissp 553–561. doi:10.5220/0006685805530561.
Yan, J., Qi, Y., Rao, Q., 2018. Detecting malware with an ensemble method based on deep neural network. Secur. Commun. Networks 2018, 1–16. doi:10.1155/2018/7247095.
I. Yoo, "Visualizing windows executable viruses using self-organizing maps," 2005, doi: 10.1145/1029208.1029222.
Yuan, Z., Lu, Y., Xue, Y., 2016. Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci. Technol. doi:10.1109/TST.2016.7399288.
S. Yue, "Imbalanced malware images classification: a CNN based approach," 2017.
YusirwanS, S., Prayudi, Y., Riadi, I., 2015. Implementation of malware analysis using static and dynamic analysis method. Int. J. Comput. Appl. doi:10.5120/20557-2943.
Zhou, Z., et al., 2017a. Fine-tuning convolutional neural networks for biomedical image analysis. Cvf, 128785 doi:10.1109/CVPR.2017.506.
Zhou, Z., Shin, J., Zhang, L., Gurudu, S., Gotway, M., Liang, J., 2017b. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017 2017 (January), 4761–4772. doi:10.1109/CVPR.2017.506.
Zhou, Z.-.H., 2009. Ensemble Learning Main Body Text. Springer, Berlin, pp. 270–273.
Zolkipli, M.F., Jantan, A., 2011. An approach for malware behavior identification and classification. ICCRD2011 - 2011 3rd International Conference on Computer Research and Development doi:10.1109/ICCRD.2011.5764001.

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. 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.

Mamoun Alazab: He is an Associate Professor in the College of Engineering, IT and Environment at Charles Darwin University, Australia. He received his PhD degree in Computer Science from the Federation University of Australia, School of Science, Information Technology and Engineering. He is a cyber security researcher and practitioner with industry and academic experience. Dr. Alazab's 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. Alazab's look into the intersection use of Artificial Intelligence and Machine Learning as essential tools for cybersecurity, for example, detecting attacks, analyzing malicious code or uncovering vulnerabilities in software and hardware. 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 Vehicular 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. Her research interest includes, Sentiment Analysis using Deep Learning, Recommendation System, Use of Artificial Intelligence in E-Commerce.


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.