JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 8 Number 1 | June 2023: 61-71
DOI: 10.15575/join.v8i1.1051

Malware Image Classification Using Deep Learning InceptionResNet-V2 and VGG-16 Method

Didih Rizki Chandranegara[1], Jafar Shodiq Djawas[2], Faiq Azmi Nurfaizi[3], Zamah Sari[4]
[1,2,3,4]Department of Informatics, University of Muhammadiyah Malang, Indonesia

Article Info

Article history:

Received March 29, 2023
Revised June 13, 2023
Accepted May 12, 2023

Keywords:

Classification
Convolutional neural network
InceptionResNet-V2
Malware
VGG-16

ABSTRACT

Malware is intentionally designed to damage computers, servers, clients or computer networks. Malware is a general term used to describe any program designed to harm a computer or server. The goal is to commit a crime, such as gaining unauthorized access to a particular system, so as to compromise user security. Most malware still uses the same code to produce another different form of malware variants. Therefore, the ability to classify similar malware variant characteristics into malware families is a good strategy to stop malware. The research is useful for classifying malware on malware samples presented as bytemap grayscale images. The malware classification research focused on 25 malware classes with a total of 9,029 images from the Malimg dataset. This research implements the VGG-16 and InceptionResNet-V2 architectures by running 2 different scenarios, scenario 1 uses the original dataset and the other scenario uses the undersampled dataset. After building the model, each scenario will get an evaluation form such as accuracy, precision, recall, and f1-score. The highest score was obtained in scenario 2 on the VGG-16 method with a score of 94.8% and the lowest in scenario 2 on the InceptionResNet-V2 method with a score of 85.1%.

Corresponding Author:

Faiq Azmi Nurfaizi
Informatics Department, Faculty of Engineering, University of Muhammadiyah Malang,
Jl. Raya Tlogomas No.246, Babatan, Tegalgondo, Kec. Lowokwaru, Malang, East Java, Indonesia
Email: faiqazmi123@webmail.umm.ac.id

1. INTRODUCTION

Malware is represented as a form of malicious software with the aim of damaging a system or program on a computer. Malware itself has many variations such as viruses, worms, or Trojans. Over the years, many systems or computer programs damaged by malware worldwide will be damaged by malware [1].

In today's modern world, many criminals in cyberspace develop malware to commit crimes such as data theft, bypassing unauthorized access controls such as personal computers or applications. In fact, according to records, at least almost every day there are thousands of crime cases using malware as a tool to gain access to the victim's network, steal victim's data, or steal by transferring the victim's money [2]. This makes malware a serious form of threat in the world of business, education, government, and individuals through the misuse of software containing malware.

The malware development industry can be said to be very lucrative for cybercriminals, when one looks at the profits it makes. With the advantages gained in the industry, both in terms of the number, type, and complexity of malware created by cybercriminals, it's getting higher [3]. With the high number and variance of malware produced, this has led to an increase in the number of malwares detected by 22.9% with a growth in the number of files contaminated with malware of 1 billion files in early 2021 [4].

As a form of prevention, initially malware-contaminated files can be detected by traditional pattern-based antivirus programs that perform evaluations based on hash value parameters, string content, or the behavior of each malware in the file. However, this malware continues to evolve and are designed to evade detection by these antivirus programs by producing variants of malware within the same malware family [5]. Basically, in developing malware variants, the average malware author still reuses code to produce different variants with similar characteristics that can be grouped into a single malware family. Thus, malware belonging to the same family can be ascertained to have a similar binary pattern and is constructive [6]. Therefore, detecting malware requires alternative solutions such as carrying out the process of identifying and classifying samples belonging to the same malware family, so that it can be easier to obtain general characteristics or patterns of the malware. With this, it is possible to implement removal procedures for malware and create new forms of mitigation strategies that work for all classes or variants of the malware.

There are several approaches to identifying malware, and the most popular form of approach is to implement static and dynamic analysis to identify a malware. Implementation of static analysis by capturing information from the malware binary without executing it first, while dynamic analysis by observing the behavior of the malware by executing it first [5]. Of the two approaches, the approach using dynamic analysis is considered better and more efficient in the long term than dynamic analysis. Even though it has maximum results in identifying a malware, dynamic analysis has limitations in the time to produce results. This is because dynamic analysis requires time for malware to fulfill its purpose first, and after that a dynamic analysis approach can be carried out by observing the behavior of the malware as a whole and this does not solve the problem of the rapid development of malware in the future [7].

To overcome the shortcomings in identifying malware using static analysis and static analysis approaches, we can use machine learning approaches and artificial neural networks (ANNs). Both have an important role in detecting important properties in malware, so that the process of extracting knowledge and patterns from the malware can help in detecting malware [8]. Specifically, each binary in the malware is processed to produce a gray image which later features of the image such as image texture, color intensity in the image, and wavelets can be extracted so that it can be used as a visualization feature for the characteristics of the malware, including the variants of the malware [4]. Thus, the results of the feature extraction process obtained can be used to train a machine learning classification model on malware images.

Inception-ResNet-v2 and VGG-16 are one of the most popular CNN (Convolutional Neural Network) deep learning architectures. Inception-ResNet-v2 is a combination architecture of the Inception structure and residual network (ResNet) connections that can perform feature extraction for classification purposes with 3 main structures, namely the convolutional layer, activation layer, and pooling layer [9]. Whereas VGG-16 is a CNN architecture consisting of 16 layers combined by iterative convolutional layers 3x3 and pooling layers 2x2, and feature extraction on VGG-16 has a good ability to obtain appropriate results in the case of image classification [10].

Previous research has been carried out in classifying malware images, one of which is research conducted by Dipendra Pant and Rabindra Bista. This study proposes image classification by implementing the VGG16, ResNet-18, and Inception-V3 methods to detect malware consisting of 25 types of malwares with a total of 9,939 images. The VGG-16 model that was built can achieve 88.40% accuracy, ResNet-18 at 90.21%, and Inception-V3 at 92.48% with a total of 20 epochs. The researcher hopes that there will be an appropriate parameter configuration for training the VGG-16 model in order to increase accuracy and lower the loss rate obtained in the VGG-16 model [11]. Then, further research using the malimg dataset consisting of 9,342 images and 25 types of malwares by Ahmed Bensaoud, Nawaf Abudawaood, and Alsoal Kalita concluded that to classify malware images using three ImageNet Large Scale Visual Recognition Challenge models and three other CNN models to get a score Perfect. Of the six models used in this study, the VGG-16 model obtained the lowest accuracy value of 15.92% with an average accuracy per malware variant of 14.31%, while the model with the highest accuracy was Inception-V3 with a value of 99.24% and an average the accuracy value per malware variant is 99.25% [12].

The same research was also conducted by Md. Haris Uddin Sharif, Nasmin Jiwani, Ketan Gupta, Mehmood Ali Mohammed, and also Dr. Meraj Fareen Ansari. They developed intelligent malware detection using 9,458 malware samples from the Malimg dataset which consists of 25 different malware classes. they implemented the CNN customization algorithm with 90.01% accuracy, Caps-Net with 90%

accuracy, VGG16 with 80.16% accuracy, ResNet-50 with 81.20% accuracy, InceptionV3 with 87.10% accuracy, and Ensemble Model with 93.30% accuracy in detecting and classifying malware. with the results of the accuracy obtained, they feel the need to develop in the form of a training scheme adjustment or model architecture that will be used to classify complex types of malwares accurately [13].

In contrast to the 3 previous studies, the research conducted by Manish Goyal and Raman Kumar used API Call visual conversions in observing different patterns of malware within the same class. So that the conversion process produces 15,217 image samples from 14 types of malware. They implemented four deep learning models in their malware classification research, namely AVMCT as the proposed model with 98.88% accuracy, VGG-16 with 94% accuracy, Resnet-50 with 98.13 accuracy, and Alexnet with 98.75% accuracy [14]. Based on four previous studies, the researcher decided to create a malware image classifier program using the InceptionResNetV2 and VGG-16 models on the Malimg malware dataset which consists of 9,029 images from 25 different malware classes.

2. METHOD

2.1. InceptionResNet-V2

InceptionResNet-V2 is a combination of the two newest neural networks methods, namely residual connections and the latest version of the Inception architecture [15]. The Inception model is famous for its multi-branch architectural form, they have a set of filters (1x1, 3x3, 5x5, etc.) which are combined with a thread on each branch.

This architecture has a split-transform-merge form of the initial module which is observed as a strong representation capability in the density of existing layers [15]. Residual models are well known for training very deep architectures. The InceptionResNet-V2 hybrid network uses a residual connection with a good level of efficiency [16]. The trunk configuration for the InceptionResNet-V2 network is not shown here but the basic concept of the network interior module form includes blocks such as Inception-ResNet-A, Inception-ResNet-B, Inception-ResNet-C [16], shown in Figure 4. The original tissue is stabilized by minimizing the residue before adding it to the previous layer.

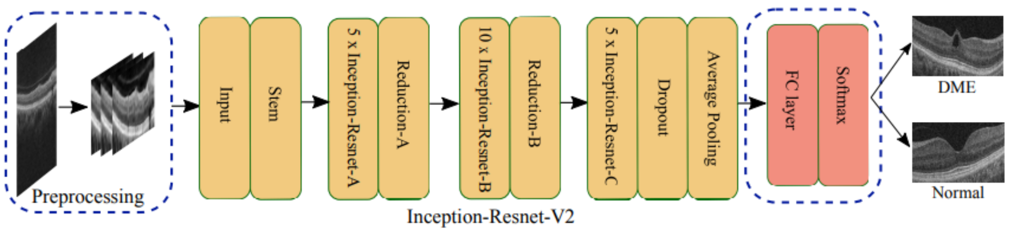


Figure 1. Deep Learning Architecture InceptionResNet-V2 Model

2.2. VGG-16

CNN's architecture has grown rapidly in recent years. One of the well-known is VGG-16 [17]. VGG-16 has a simple layer architecture but is able to provide high accuracy results on Imagenet datasets and is better than Alexnet and ZF-NET. VGG-16 is one of the most common deep learning architectures introduced by the University of Oxford, like the pre-trained model, VGG-16 requires heavy training if the weights are randomly initialized. So, in general the CNN model uses transfer learning techniques. transfer learning refers to the mechanism by which a model trained on one task is used in some way on a similar task in the second stage [18].

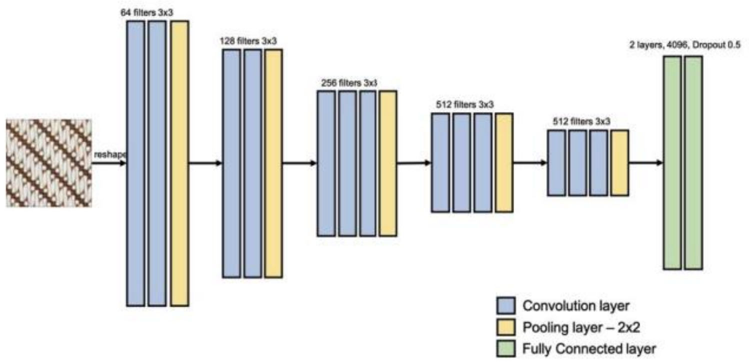


Figure 2. Deep Learning Architecture VGG-16 Model

CNN's architecture has grown rapidly in recent years. One of the well-known is VGG-16 [17]. VGG-16 has a simple layer architecture but is able to provide high accuracy results on Imagenet datasets and is better than Alexnet and ZF-NET. VGG-16 is one of the most common deep learning architectures introduced by the University of Oxford, like the pre-trained model, VGG-16 requires heavy training if the weights are randomly initialized. So, in general the CNN model uses transfer learning techniques. transfer learning refers to the mechanism by which a model trained on one task is used in some way on a similar task in the second stage [18].

CNN is an artificial neural network that uses certain layers in the hidden layer. One of the main layers is the convolutional layer. The hidden layers used are usually quite a lot or more than three layers [19]. Because of that, CNN is categorized as a deep learning method. The function of this layer is to extract features automatically from the input data and classify them at the end. CNN was introduced to recognize handwriting. The image will move into the existing layer and be treated as a high-order matrix or tensor. The training weight values are stored in a particular layer and are often referred to as parameters. Common layers in CNN are as follows:

2.2.1. Input Layer

Input Layer is the layer associated with image data input [20]. The input layer is usually a tensor with dimensions similar to the dimensions of the input image, namely the length, width and number of channel images or their transformations.

2.2.2. Convolutional Layer

The convolutional layer is formed from a set of convolutional kernels, in which each neuron acts as a kernel [21]. The convolution layer is the layer that performs the convolution process from the previous layer. This layer stores the parameters or weights of the training results. The output of this layer (in tensor form is often referred to as a feature map) usually has a smaller length and width than the input layer but has a greater depth. This layer stores training weight values as parameters which can be calculated by:

$$P = (N \, x \, M \, x \, L + B) \, x \, K \tag{1}$$

Where N and M are filter sizes, feature map L as input, B as bias whose value is one, and K as output.

2.2.3. Activation Layer

This layer functions as an activation function of the convolution layer, which is useful for solving non-trivial problems in an artificial neural network [22]. The activation function is usually a rectified linear unit (Relu). It is very lightweight because it only changes the negative input value to zero while the positive has a fixed value.

2.2.4. Pooling Layer

The purpose of pooling is to down-sample or sub-sampling feature maps, and besides that pooling can also study large-scale image features that are invariant [23]. This layer performs subsampling to the previous layer to retrieve optimal features from the input tensor. The kernel size determines the output size of the input tensor. For example, if the kernel size is two, then the output size is divided by two.

2.2.5. Fully-connected Layer

Fully-connected layer is the last layer that functions as a classifier, so this layer is an important element of a convolutional neural network [24]. This layer generally uses a neural network that can be trained and can store the weight values of the training results. The activation function in this layer is usually softmax to highlight the most dominant class. This is the layer that stores the weight values and the number of parameters calculated by a formula like a convolution layer with a kernel size of 1.

$$P = (N\ x\ M\ x\ L + B)\ x\ K \tag{1}$$

Where L as input, K as output, and B as bias whose value is one.

2.3. Research Stages

Based on the presentation in the introduction and the problem formulation on the topic and research raised, the researcher intends to conduct research with the aim of implementing the methods used into the problems on the main topic of this research, namely classifying malware images using the InceptionResNetV2 and VGG-16 methods. Another goal of research conducted using these two methods is to find out how the results or performance are obtained by implementing these two methods for this study. In addition, because the original dataset is still unbalanced, the researcher carries out a balancing process for the dataset used. So, there are two datasets which later there will be two scenarios that will be implemented in this study, namely implementing the method using an imbalanced dataset and implementing the method using a balanced dataset. The stages and steps in the research method this time can be seen in the flowchart and the following explanation:

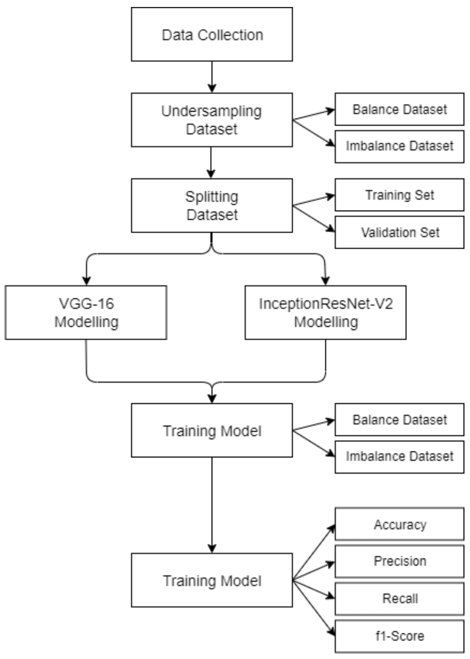


Figure 3. Research Flow

Based on the flow in Figure 3, the following is an explanation of each stage in this research method:

a. Dataset preparation, the dataset used is a malware image dataset sourced from Kaggle.
b. Because the original dataset was imbalanced, the researcher conducted 2 tests with a dataset that was still imbalanced and a dataset that had been undersampled or a dataset that was balanced.

c. Building a setup using a batch size for both datasets that will be used is 25, this will affect the results at the data training stage with that batch size, the image will be re-produced as much as the value set per iteration.
d. Splitting the two datasets with a comparison of 90% for training data and 10% for validation data..
e. Pre-processing the train data and test data by normalizing the previous data which has a scale of [0.128] to [0.1] and resizing the image to a size of 128x128 using class_mode 'Categorical' and with the color type RGB.
f. Create models from both datasets that have been prepared using the activation type 'softmax'.
g. After creating models from the two datasets used, these models will be implemented with the summary model library with two methods, namely InceptionResNetV2 and VGG-16.
h. Doing data training on a model that has been prepared with a total of 20 epochs, because this uses a Deep Learning model, the number of epochs used is 20, in our opinion, this is more than enough considering the large number of steps in each epoch and the batch size value used is quite large.
i. Create a function for predicting data that has passed the training stage for test data using a heatmap and displaying plots of Accuracy and Loss values for each method.

2.4. Research Variables

Based on the explanation in the previous sub-chapter and data collection, there are several determining variables which are the main keys in this research, namely the number of image categories in the dataset used, the number of batch sizes used in images, and the pre-processing stages carried out will affect how the results of the course of this research process and the final results of the accuracy performance produced by each method.

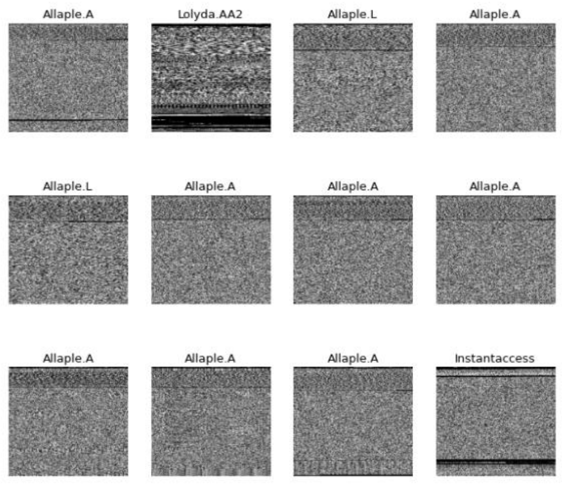


Figure 4. Malware bytemap grayscale

In this study, researchers created two scenarios using two datasets. The first dataset (balance) is a dataset with a total sample of 9,029 image samples spread unevenly across 25 classes and the second dataset (balance) is a dataset with a total sample of 2,000 image samples spread evenly across 25 classes. The form of this image is a form of conversion of malware that has been converted into a bytemap with grayscale colors. Accuracy at a high level of dataset similarity in each scenario makes a fairly high level of difficulty in classifying each existing malware category.

Table 1. Comparison of Dataset Undersampling Results

No	Data	Imbalance Dataset (Original)	Balance Dataset
1	Training Data	8404 Images	1375 Images
2	Validation Data	625 Images	625 Images

3. RESULT AND DISCUSSION

In this study, researchers will implement 2 scenarios. Scenario 1 will implement the VGG-16 and InceptionRestNet-V2 models using the original dataset which is still imbalanced and Scenario 2 will implement the VGG-16 and InceptionRestNet-V2 models using the original dataset that has been undersampled or balanced dataset.

3.1. Scenario 1

In Scenario 1, the researcher used the original dataset to obtain 8404 training data images and 625 image validation data. The results of accuracy in scenario 1 can be seen in table 2.

Table 2. 1st Scenario Accuracy Results

No	Evaluation	VGG-16	InceptionResNet-V2
1	Accuracy on Data Train	93.6%	94.8%
2	Accuracy on Data Validation	87.2%	87.5%

After getting the results, the next step is to display the results of the model training process which are represented through plotting in the form of lines. This is to display the performance and overfitting of the model in each iteration. Following are the results of plotting for each model.

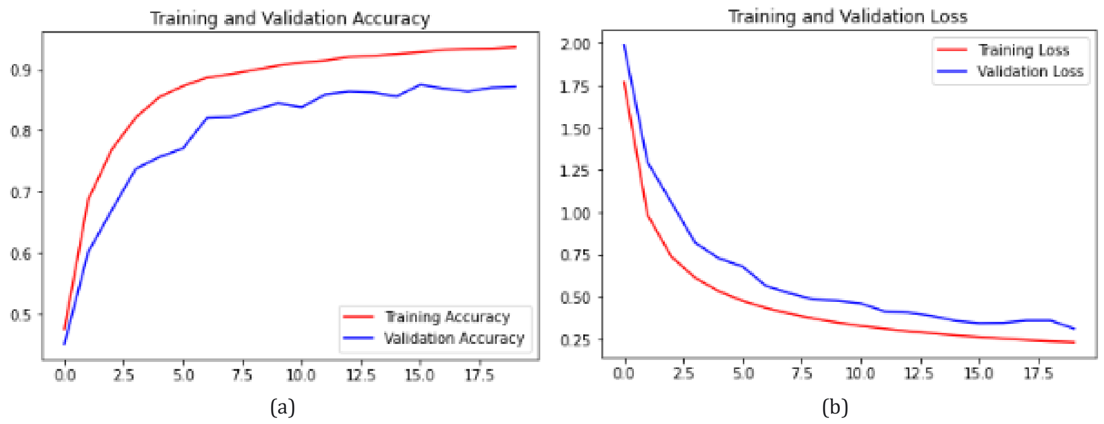


Figure 5. (a) Accuracy Graph and (b) Loss Graph of VGG-16 Training Model in Scenario 1

In Figure 5, the distance between the accuracy values of the training data and validation data in the VGG-16 model is not too large. Based on the graph, the more iterations are done, the value is closer to 1.

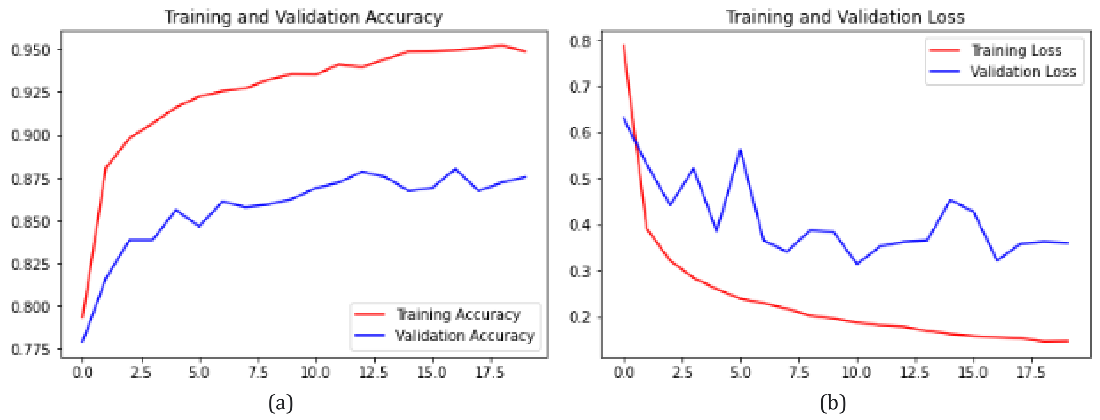


Figure 6. (a) Accuracy Graph and (b) Loss Graph of InceptionRestNet-V2 Training Model in Scenario 1

In Figure 6, the InceptionResNet-V2 model has a fairly large accuracy value between the training data and validation data. In training data, the more iterations that are implemented, the closer to one. Meanwhile, in data validation, the more iterations, the more stable at 80% to 87%. After getting the results of the training performance of each model which is interpreted through a line graph, then evaluate the model that has been made. The performance results for each model are then interpreted through reports in table 3.

Table 3. Model Evaluation Results in Scenario 1

No	Evaluation	VGG-16	InceptionResNet-V2
1	Accuracy	87.2%	87.5%
2	Precision	86%	86%
3	Recall	87.2%	87.5%
4	F1-Score	85.4%	85.6%

In table 3, the difference in the evaluation results for the two models is very small. However, the InceptionRestNet-V2 model has a higher score in all evaluation types when compared to the VGG-16 model.

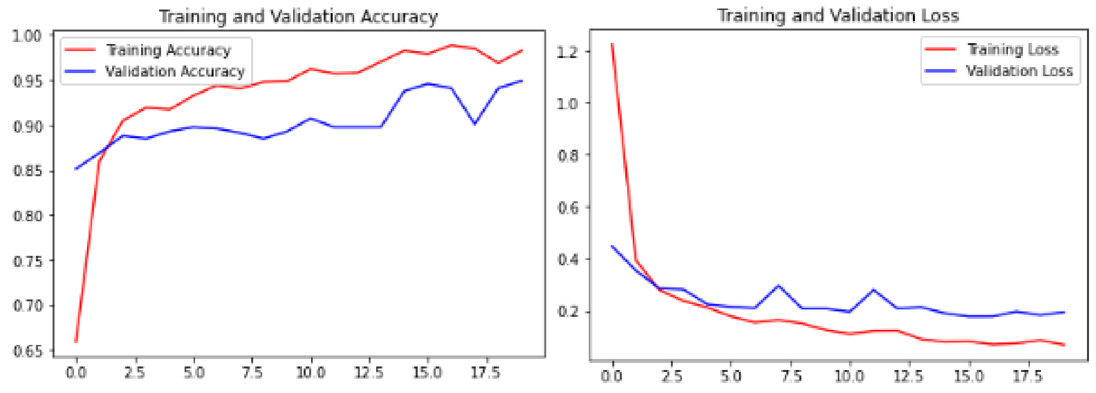
3.2. Scenario 2

In Scenario 2, the researcher uses the original dataset that has been undersampled so that the data is balanced. In this division, 1,375 images are training datasets and 625 are validation datasets. The results of the accuracy of the training model in scenario 2 can be seen in table 4 below.

Table 4. 2st Scenario Accuracy Results

No	Evaluation	VGG-16	InceptionResNet-V2
1	Accuracy on Data Train	98.2%	94.1%
2	Accuracy on Data Validation	94.8%	85.1%

After getting the accuracy results for each model, then displaying the results of the model training process which is represented through plotting in the form of lines, in order to display performance and evaluate model overfitting based on the value in each iteration. The following is a graphic on each model:



(a) (b)

Figure 7. (a) Accuracy Graph and (b) Loss Graph of VGG-16 Training Model in Scenario 2

In the VGG-16 model in Scenario 2, the accuracy value in each iteration is at a value of 85% to 98% and the distance between the two values is not too large even though the validation data is not very stable, which initially increases not too significantly in the accuracy value.

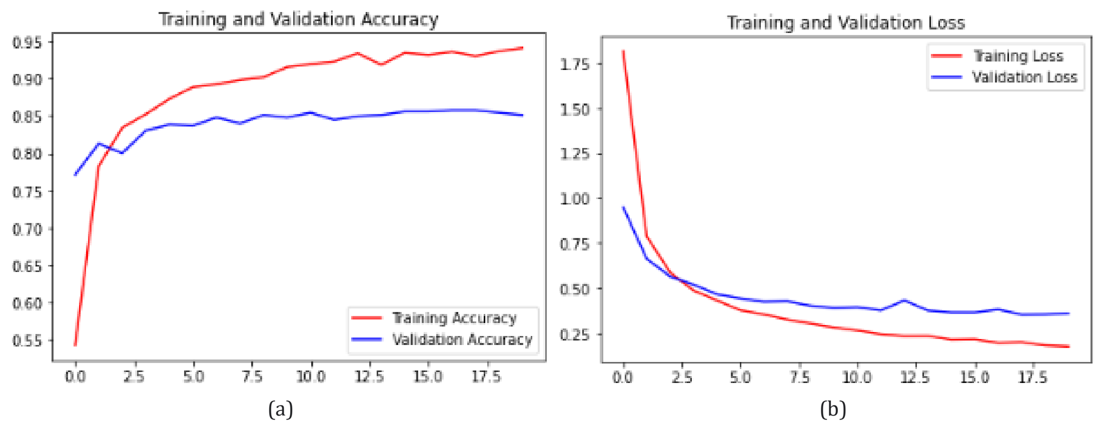


Figure 8. (a) Accuracy Graph and (b) Loss Graph of InceptionRestNet-V2 Training Model in Scenario 2

The InceptionRestNet-V2 model shown in Figure 8 shows that there is a significant increase in value at the beginning of the iteration and ends with a fairly stable value compared to the VGG-16 model, both in training data and data validation. After obtaining the accuracy value for each implemented model, the performance of each model in scenario 2 will be interpreted through the report in table 5 below:

Table 5. Model Evaluation Results in Scenario 2

No	Evaluation	VGG-16	InceptionResNet-V2
1	Accuracy	94.8%	85.1%
2	Precision	94.9%	82.7%
3	Recall	94.8%	85.1%
4	F1-Score	94.3%	83.3%

The model evaluation results in scenario 2 show better results when compared to scenario 1. In addition, the model evaluation value in the VGG-16 model has a higher value compared to the InceptionRestNet-V2 model, with an average percentage difference between the 2 the model is 10.65%.

3.3. Summary of Model Performance Results

After running 2 test scenarios, the resulting performance varies greatly. Table 6 is the model performance value based on the accuracy resulting from testing 2 scenarios. The highest accuracy value for the accuracy of the model for each scenario is obtained with a value of 94.8% by the VGG-16 model in scenario 2 and the lowest value is 87.5% by the InceptionRestNet-V2 model in scenario 2. Scenario 2 is a scenario with the implementation of a balanced dataset composition in each class. Whereas scenario 1 which implements an unbalanced dataset shows an accuracy below 90% in each model.

Table 6. Model Accuracy Results in Each Scenario

No	Scenario	VGG-16	InceptionResNet-V2
1	Scenario 1 (with Imbalance Dataset)	87.2%	87.5%
2	Scenario 2 (with Balance Dataset)	94.8%	85.1%

In addition to the highest accuracy value in the 2 test scenarios, the VGG-16 model in scenario 2 has the highest model accuracy rate when compared to the 4 previous research tests. Table 7 shows a comparison of the results of the accuracy of the VGG-16 model in previous studies with the results of the accuracy of the VGG-16 model in the 2 scenarios in this study.

Table 7. Comparison of Accuracy Model Results in Previous Research

No	Reference	Date	Dataset	VGG-16
1	[Dipendra]	2021	Malimg	88.40%
2	[Bensaoud]	2020	Malimg	15.92%
3	[Haris]	2023	Malimg	80.16%
4	[Goyal]	2022	API Calls	94%
5	Proposed Research (Scenario 1)	2023	Malimg	87.2%

No	Reference	Date	Dataset	VGG-16
6	Proposed Research (Scenario 2)	2023	Malimg	94.8%

When compared with previous studies, the VGG-16 model in scenario 2 of the proposed research has increased accuracy with the same model in previous studies. The VGG-16 model proposed in scenario 2 experiences an average increase in accuracy of 25.18%.

From the test results obtained, the VGG-16 model in scenario 2 using a balanced dataset has increased by 7.6% and the InceptionRestNet-V2 model in scenario 2 has decreased by 2.4%. If we analyze it, a more balanced composition of the dataset has a good performance effect on the VGG-16 model, whereas it has no effect on the InceptionResNet-V2 model, it even tends to decrease. In addition, the difference in architectural composition of the VGG-16 model with previous research also affects the performance of the model, but the use of a more balanced dataset still has a major effect on the performance of the VGG-16 model, this can be seen from the average value of the increase in accuracy.

4. CONCLUSION

Research on the classification of malware types based on malware bytemap images by proposing the implementation of the Deep Learning method. This study implements the VGG-16 and InceptionResNetV2 architectures to extract features from images in Malware images. Based on the 2 scenarios performed, the best model results achieved the highest accuracy of 94.8% on the VGG-16 model in scenario 2 and the lowest 85.1% on the InceptionResNet-V2 model in scenario 2. The test results for the 2 scenarios in this study stated that the data were imbalanced or being balanced has a big influence on the performance of the VGG-16 model, but not on the InceptionResNet-V2 model. Therefore, the big influence of the performance level of the InceptionRestNet-V2 model is not on the composition of the dataset, but on the architectural form of the model built.

It is hoped that in the future an automation system can be built to classify types of malwares that can recognize different types of malware in the form of bytecode images. To realize this system, it is necessary to analyze more datasets on malware images with the basic form of bytecode images. Suggestions for future research are to try using different methods for classifying malware images and with similar datasets and evaluating various other feature extraction methods designed for classification problems may give different results, especially on the combined architectural model of Inception structure and residual network connection.

ACKNOWLEDGEMENTS

This research was supported by the Department of Informatics of University of Muhammadiyah Malang, who gave the opportunity for us to do this research.

REFERENCES

[1] M. A. Hama Saeed, "Malware in Computer Systems: Problems and Solutions," *IJID (International J. Informatics Dev.*, vol. 9, no. 1, p. 1, 2020, doi: 10.14421/ijid.2020.09101.

[2] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," *Symmetry (Basel).*, vol. 14, no. 11, p. 2304, 2022, doi: 10.3390/sym14112304.

[3] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of Malware Threats and Techniques: A Review," *Int. J. Commun. Networks Inf. Secur.*, vol. 12, no. 3, pp. 326–337, 2020, doi: doi.org/10.17762/ijcnis.v12i3.4723.

[4] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware Detection Issues, Challenges, and Future Directions: A Survey," *Appl. Sci.*, vol. 12, no. 17, 2022, doi: 10.3390/app12178482.

[5] S. Choi, J. Bae, C. Lee, Y. Kim, and J. Kim, "Attention-based automated feature extraction for malware analysis," *Sensors (Switzerland)*, vol. 20, no. 10, pp. 1–17, 2020, doi: 10.3390/s20102893.

[6] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, and M. S. Javed, "Identification of malicious code variants based on image visualization," *Comput. Electr. Eng.*, vol. 76, pp. 225–237, 2019, doi: 10.1016/j.compeleceng.2019.03.015.

[7] M. J. Awan *et al.*, "Image-based malware classification using vgg19 network and spatial convolutional attention," *Electron.*, vol. 10, no. 19, 2021, doi: 10.3390/electronics10192444.

[8] U.-H. Tayyab, F. B. Khan, M. H. Durad, A. Khan, and Y. S. Lee, "A Survey of the Recent Trends in Deep Learning Based Malware Detection," *J. Cybersecurity Priv.*, vol. 2, no. 4, pp. 800–829, 2022, doi: 10.3390/jcp2040041.

[9] A. Thomas, P. M. Harikrishnan, P. Palanisamy, and V. P. Gopi, “Moving Vehicle Candidate Recognition and Classification Using Inception-ResNet-v2,” *Proc. - 2020 IEEE 44th Annu. Comput. Software, Appl. Conf. COMPSAC 2020*, pp. 467–472, 2020, doi: 10.1109/COMPSAC48688.2020.0-207.

[10] Q. Guan *et al.*, “Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: A pilot study,” *J. Cancer*, vol. 10, no. 20, pp. 4876–4882, 2019, doi: 10.7150/jca.28769.

[11] D. Pant and R. Bista, “Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning,” *ACM Int. Conf. Proceeding Ser.*, 2021, doi: 10.1145/3503047.3503081.

[12] A. Bensaoud, N. Abudawaood, and J. Kalita, “Classifying Malware Images with Convolutional Neural Network Models,” 2020, doi: 10.48550/arXiv.2010.16108.

[13] H. U. Sharif, N. Jiwani, K. Gupta, M. A. Mohammed, and M. F. Ansari, “a Deep Learning Based Technique for the Classification of Malware Images,” *J. Theor. Appl. Inf. Technol.*, vol. 101, no. 1, pp. 135–160, 2023, [Online]. Available: http://www.jatit.org/volumes/Vol101No1/12Vol101No1.pdf

[14] K. R. Goyal Manish, “AVMCT: API Calls Visualization based Malware Classification using Transfer Learning,” vol. 13, no. 1, pp. 31–41, 2022, doi: doi.org/10.52783/jas.v13i1.59.

[15] E. G. Winarto *et al.*, “Implementasi Arsitektur Inception Resnet-V2 untuk Klasifikasi Kualitas Biji Kakao,” *Proceeding KONIK (Konferensi Nas. Ilmu Komputer)*, pp. 132–137, 2021, [Online]. Available: https://prosiding.konik.id/index.php/konik/article/view/38

[16] P. Nugraha, A. Komarudin, E. Ramadhan, and D. Learning, “Deteksi Objek Dan Jenis Burung Menggunakan Convolutional Neural Network Dengan Arsitektur Inception Resnet-V2,” *INFOTECH J.*, vol. 8, pp. 47–55, 2022, doi: doi.org/10.31949/infotech.v8I2.2889.

[17] I. B. K. Sudiatmika and I. G. A. A. S. Dewi, “Indonesian Shadow Puppet Recognition Using VGG-16 and Cosine Similarity,” *Int. J. Informatics Comput. Sci.*, vol. 5, no. 1, pp. 1–6, 2021, doi: 10.30865/ijics.v5i1.2579.

[18] D. Albashish, R. Al-Sayyed, A. Abdullah, M. H. Ryalat, and N. Ahmad Almansour, “Deep CNN Model based on VGG16 for Breast Cancer Classification,” *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, pp. 805–810, 2021, doi: 10.1109/ICIT52682.2021.9491631.

[19] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.

[20] C. L. Fan and Y. J. Chung, “Design and Optimization of CNN Architecture to Identify the Types of Damage Imagery,” *Mathematics*, vol. 10, no. 19, 2022, doi: 10.3390/math10193483.

[21] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A Survey of the Recent Architectures of Deep Convolutional Neural Networks,” *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020, doi: 10.1007/s10462-020-09825-6.

[22] M. R. Alwanda, R. Putra, K. Ramadhan, D. Alamsyah, P. Studi, and T. Informatika, “Implementasi Metode Convolutional Neural Network Menggunakan Arsitektur LeNet-5 untuk Pengenalan Doodle,” *J. Algoritm.*, vol. 1, no. 1, 2020, doi: doi.org/10.35957/algoritme.v1i1.434.

[23] R. Nirthika and S. Manivannan, “Pooling in convolutional neural networks for medical image analysis : a survey and an empirical study,” *Neural Comput. Appl.*, vol. 34, no. 7, pp. 5321–5347, 2022, doi: 10.1007/s00521-022-06953-8.

[24] M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad, and M. Khan, “Comparative Analysis of Recent Architecture of Convolutional Neural Network,” vol. 2022, 2022, doi: doi.org/10.1155/2022/7313612.