Computer Engineering and Applications Vol. 8, No. 3, October 2019

The Implementation of Deep Neural Networks Algorithm for Malware Classification

Nurul Afifah[1*], Deris Stiawan[1], Siti Nurmaini[2], Firdaus[2]

[1]*Department of Informatics Engineering, Faculty of Computer Science, Universitas Sriwijaya*
[2]*Intelligent System Research Group, Universitas Sriwijaya*
**afifahnurul95@gmail.com*

ABSTRACT

Malware is very dangerous while attacked a device system. The device that can be attacked by malware is a Mobile Phone such an Android. Antivirus in the Android device is able to detect malware that has existed but antivirus has not been able to classify new malware that attacks an Android device. In this issue, malware classification techniques are needed that can grouping the files between malware or non-malware (benign) to improve the security system of Android devices. Deep Learning is the proposed method for solving problems in malware classification techniques. Deep Learning algorithm such as Deep Neural Network has succeeded in resolving the malware problem by producing an accuracy rate of 99.42%, precision level 99% and recall 99.4%.

Keyword: Malware, Benign, Android, Deep Neural Network.

1. INTRODUCTION

Malware is one of the most dangerous types of attacks for mobile phone, especially Android [1]. Malware infiltrated various applications in the Android application and encrypted someone's data to damage the entire mobile phone system [2]. Many types of malware are often used by someone to damage Android security systems such as Droidkungfu, Trojan, Ransomware, and others [3]. With various malware attacks in the mobile phone, a method that can solve malware problems is proposed with the Deep Neural Network and Convolutional Neural Network algorithms [4].

The malware detection technique uses Convolutional Neural Network algorithms generally used for malware extracted into the form of images or data that are visualized first [5]. To implement an algorithm in malware classification it is necessary to first look at the dataset obtained because the contents of the dataset determine which algorithm is suitable for implementation [6]. For the Deep Neural Network algorithm carried out by [6], the accuracy rate is higher at 93.67% with balanced data. In the result get high accuracy results obtained using the Deep Neural Network algorithm, it means that the Deep Neural Network algorithm is very suitable to be implemented in terms of malware classification techniques because it has good accuracy when implemented [6]. In part 1 this paper discusses the introduction of the contents of the paper. Part 2 discusses the dataset and the proposed method. Section 3 discusses preprocessing data and drawing DNN models. Section 4 discusses evaluating the performance of the ROC curve and a confusion matrix. The last section discusses the overall conclusion.

2. DEEP NEURAL NETWORK

Deep Neural Network (DNN) is a neural network-based algorithm that can be used for decision making that has more than one hidden nerve layer. This algorithm is the development of intelligence namely ANN algorithm (Artificial Neural Network) [6]. To get high level in accuracy of this algorithm, data needs to be trained first [8]. DNN consists of several layers and neurons in each layer. Both of these cannot be determined using definite rules and apply differently to different data [9]. In the development of Deep Learning, it was found that to overcome backpropagation shortcomings in handling complex data, a function is needed to transform input data into a form that is easier for backpropagation to understand. This triggers the development of Deep Learning wherein one model are given several layers to transform data before the data is processed using the classification method. This triggers the development of the neural network model with a number of layers above three. But due to the initial layer function as a feature extraction method, the number of layers in a DNN does not have universal rules and applies differently depending on the dataset used [10]. Here's the backpropagation theorem.

$$y(vi) = \tanh(vi) \qquad (1)$$

Here *y*i is the output of the i (neuron) and vi is the weight sum of the input connections

$$y(vi) = \left(1 + e^{-vi}\right) \qquad (2)$$

The nodes we have are based on corrections that reduce errors in the entire output, given by,

$$\varepsilon(n) = \frac{1}{2} \sum e^2(n) \qquad (3)$$

The change in each weight is,

$$\Delta wji(n) = -\mu \frac{\partial \varepsilon(n)}{\partial vj(n)} \, yi(n) \qquad (4)$$

The derivative to be calculated depends on the induced local field *vj*. The output of this derivative node can be simplified to,

$$-\frac{\partial \varepsilon(n)}{\partial vj(n)} = ej(n)\emptyset(vj(n)) \tag{5}$$

This depends on the change in weights of the k nodes which represents the output layer:

$$-\frac{\partial \varepsilon(n)}{\partial vj(n)} = \emptyset(vj(n)) \sum_k -\frac{\partial \varepsilon(n)}{\partial vk(n)} wkj(n) \tag{6}$$

DNN have three part of layer such an input, hidden and output layer. These cannot be determined using definite rules and apply differently to different data. The feedforward process works by multiplying the neurons so that many networks are formed which add to the variant of data to be studied. The results of the feedforward process are weights that will be used to evaluate the process of neural networks. While the testing process is a classification process using weights and biases from the results of the training process. So the result of this process is the accuracy of the classification performed. With weight and bias, a new feedforward process is applied to produce an output layer for inputting.

3. MATERIAL AND METHOD

3.1 DATASET

The dataset is obtained from the malgenome dataset project [7]. The malware and benign files obtained amount to 7414 data with the number of malware files 3707 and benign files 3707 [7]. Then the dataset is labeled X and Y. The data is then processed by preprocessing first. Then it will enter the DNN classification process. Can be seen in Table 1.

TABLE 1.
Dataset Description

	Transact	bindService	onServiceConnected	Service Connection	Read_sms
Count	3799.0	3799.0	3799.0	3799.0	3799.0
Mean	0.45224	0.467228	0.470387	0.500921	0.256383
Std	0.4977	0.4989	0.4991	0.4991	0.500065
Min	0	0	0	0	0
25%	0	0	0	0	0
50%	0	0	0	0	0
75%	1	1	1	1	1
max	1	1	1	1	1

3.2 DATA PREPROCESSING

The dataset was used in this malware detection technique is 7414 files. Because the dataset is imbalanced, it is necessary to do a dataset resampling technique using

SMOTE [8]. Data imbalance occurs when the number of objects in a data class is more than the other classes. Data classes with more objects are called major classes while others are called minor classes. The effect of unbalanced data usage to make a very large model on the results of the model obtained. Processing algorithms that ignore data imbalances will tend to be covered by major classes and ignoring minor classes [8].

The SMOTE method is proposed as one of the solutions to handling unbalanced data with different principles with the oversampling method that has been proposed previously. If the oversampling method is based on multiplying random observations, the SMOTE Method increases the number of minor class data to be equivalent to the major class by generating artificial data [8]. Artificial data generation on a numerical scale differs from categorical.

Numerical data is measured by the proximity to Euclidean distance, while categorical data is simpler, namely the mode value. The distance calculation between examples of minor classes whose variables are categorical in scale is done by the formula Value Difference Metric (VDM) in Equation 7 and 8.

$$\Delta(X,Y) = (WxWy \textstyle\sum_{i=1}^{n} \delta(xi, yi)^r) \tag{7}$$

$$\delta(V1,V2) = \textstyle\sum_{i=1}^{n} \mid \frac{C1i}{C1} - \frac{C2i}{C2} \mid k \tag{8}$$

Using the SMOTE approach, the amount of data is added to become 7414 data. The addition of the above data is useful for changing data that is initially imbalanced to be balanced.

3.3 DEEP NEURAL NETWORK ARCHITECTURE

The implementation of DNN hidden layer is recommended above 3 layers because to increase the value of accuracy in training and testing. In terms of the classification of malware, the following DNN network architecture is proposed as shown in Figure 1.

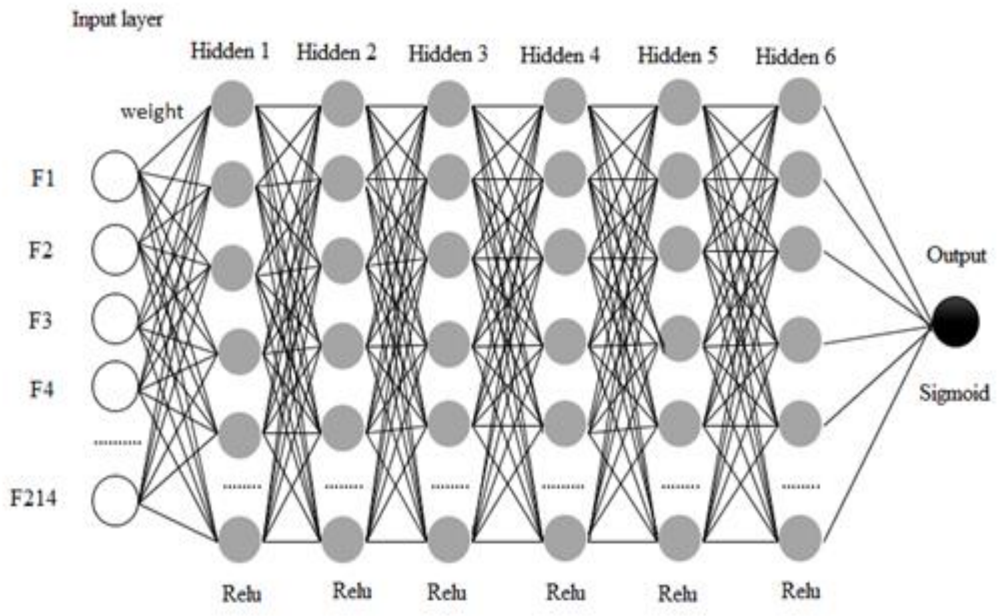


FIGURE 1. DNN network architecture

Figure 1 shows a series of DNN network architecture consisting of input layers, hidden layers, and output layers. There are 214 features with 75 epochs. At the input layer, there are 214 input. In the first hidden layer until last hidden layer, there are 150 nodes with ReLu activation function. In output layer use the sigmoid activation function. The difference in this activation function is due to the different roles of each unit. It has a role in accelerating the convergence process. Relu only creates a threshold for zero numbers.

Sigmoid in the output layer because it converts the value to non-linear and has a value of 0 and 1 and in this study applying binary crossentropy , the final result is 0 and 1 which means the sigmoid function must be placed at the output layer. To start the classification process, it is preceded by a data training process.

1. Create input layer (xi), hidden layer (z_inj), wight (vij)

$$z\_inj = xi * vij$$

$$z\_inj = z\_inj + vij$$

$$zj = f(z\_inj) \tag{9}$$

2. Create output in (y_ink), output value in hidden layer(zj), weight (wjk)

$$y\_ink = zj * wjk$$

$$y\_ink = y\_ink + wjk$$

$$\text{yk}=\text{f(y\_ink)} \tag{10}$$

3. Determine error in the output layer by using relu used cross-entropy method

$$Loss = -\left(\frac{1}{n}\right)(output\ x \log yi) + (1 - output)\ x \log(1 - yi) \tag{11}$$

4. Calculate the loss function to find the gradient by partial derivative of the sigmoid by use chain rule

$$\mathrm{f}'^{(\mathrm{sigmoid})} = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}}$$

$$\mathrm{f}'^{(\mathrm{sigmoid})} = \mathrm{yi} * (1 - \mathrm{yi})$$

$$\mathrm{f}'^{(\mathrm{sigmoid})} = -\mathrm{yi} * \mathrm{yj}$$

$$\delta\mathrm{k} = (\mathrm{tk} - \mathrm{yk})\mathrm{f}'(\mathrm{y\_ink}) \tag{12}$$

5. Update all weight

$$\Delta\mathrm{wjk} = \alpha\delta\mathrm{k}\ \mathrm{zj} \tag{13}$$

6. Error calculation in each weight

$$\delta_{\mathrm{inj}} = \textstyle\sum_{\mathrm{k}=1}^{\mathrm{m}} \delta\mathrm{k}\ \mathrm{wjk} \tag{14}$$

$$\delta\mathrm{j} = \delta_{\mathrm{inj}}\ \mathrm{f}'(\mathrm{z}_{\mathrm{inj}}) \tag{15}$$

7. Update the bias

$$\Delta\mathrm{vij} = (\alpha\delta\mathrm{j}\ \mathrm{xi}) \tag{16}$$

8. Update all parameters (weight and bias)

$$\mathrm{wjk(new)} = \mathrm{wjk(old)} + \Delta\mathrm{wjk} \tag{17}$$

$$\mathrm{vij(new)} = \mathrm{vij(old)} + \Delta\mathrm{vij} \tag{18}$$

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

The training process is a stage where the DNN is trained to obtain high accuracy from the classification carried out. This stage takes 70% of the sample dataset with split validation 0.33 and 75 epochs. This stage consists of feedforward and backpropagation processes. The feedforward process works by multiplying its neurons so that many networks are formed which add variants of data to be studied. The results of the feedforward process are weights and loss function. Loss function is used to backward process to get gradient. Gradient is used to minimize error in backpropagation process. While testing is the classification process using weights and biases from the results of the training process. In the testing stage, the process is to test all the training data. So the final result of this process is accuracy in training data, accuracy in testing data, and all the parameter performance. With the weights and biases, the forward process is applied which then produces the output layer. The output is fully connected layer to get the error loss function. The stage of the proposed algorithm as shown in Figure 2.

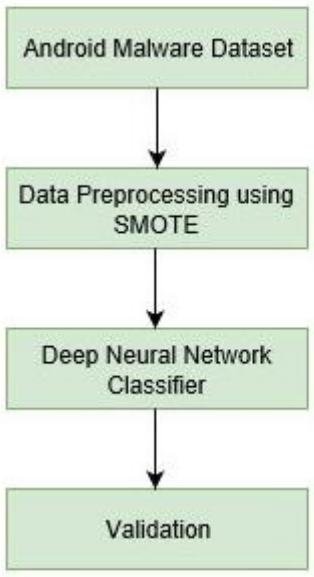

Android Malware Dataset
Data Preprocessing using SMOTE
Deep Neural Network Classifier
Validation

FIGURE 2. The stage of classification process

3.4 PERFORMANCE OF MALWARE CLASSIFICATION ALGORITHM

At this stage, all data from the experiment will be validated. Validation aims to determine whether the simulation of the benign ransomware classification system is in accordance with the predictions that have been made. This validation process will be carried out using a confusion matrix. This matrix describes the classification performance of experimental data that contains data prediction information.

In measuring performance using confusion matrix, there are four terms of representation of the results of the classification process, namely True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) in Table 2.

TABLE 2.
Confusion Matrix

Confusion Matrix	
True Positive (TP)	Positive data detected correctly
False Negative (FN)	Negative data detected positive data
False Positive (FP)	Negative data detected correctly
True Negative (TN)	Positive data but detected negative data

Based on the Table 1 describe the confusion matrix of data. True positive indicate that normal file but detected as a malware. False negative indicate the malware but detected in normal file. False positive indicate that malware detected correctly. And the last is true negative is normal file but detected malware.

Accuracy score is describe the system can classify data correctly or not. Accuracy is comparison of true positive and true negative with overall data such true positive, true negative, false positive and false negative. The formula of accuracy is in the Equation 19.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{19}$$

The precision score represents the number of correctly classified positive categorized data divided by the overall data classified as positive. What we want is high precision and recall, but this is very rare. By testing a series of models and plotting their precision and recall, their curves will give you an idea of the ideal tradeoff you need to go to.

To find out whether the proposed algorithm is biased towards a positive if we have very low precision, but very high recall. Precision can be obtained by Equation 20.

$$Precision = \frac{TP}{TP+FP} \tag{20}$$

The score of recall indicate the percentage of positive category data correctly classified by the system. Recall can be obtained with the Equation 21.

$$Recall = \frac{TP}{TP+FN} \tag{21}$$

Meanwhile, specificity formula is comparison of true negative with true positive and false positive. Specificity formula in Equation 22.

$$Specificity = \frac{TN}{TN+FP} \tag{22}$$

4. EXPERIMENTATION AND RESULTS

The results in the classification of malware and benign files obtained using the Deep Neural Network algorithm produces a very high level of accuracy with an accuracy of 99.42%. The following graph shows the accuracy of training and testing as shown in the following Figure 3.

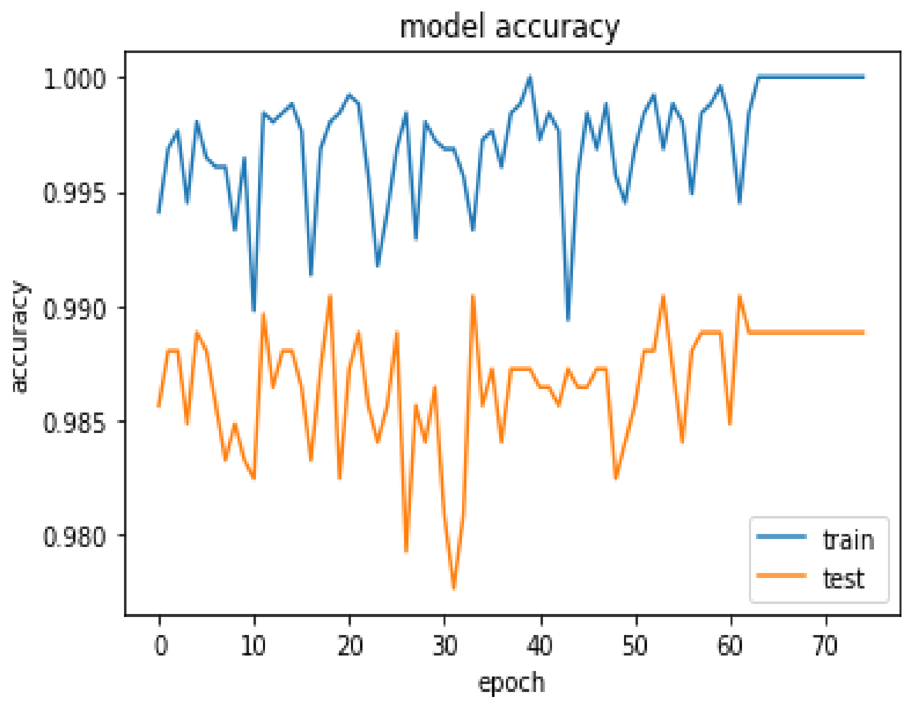


FIGURE 3. Training and Testing Accuracy Curves

Based on the Figure 3 in accuracy of the classification system. In training process get 0.991 until 0.999 in accuracy plot. But in testing process get the 0.960 until 0.990 in accuracy plot. The plot of the accuracy and loss charts show good results. The best score in the plot analysis is if the training process and testing process result the convergen graph.

Convergen is very need to be in graphic result. It seems the result is good or not. accuracy model and loss model obtained from the training and testing process The training process uses the relu activation function in the hidden layer and the sigmoid activation function in the output layer. The loss function used is binary crossentropy and uses Adam optimizer. The plot of the accuracy and loss charts show good results. the plot of loss function as shown in Figure 4.

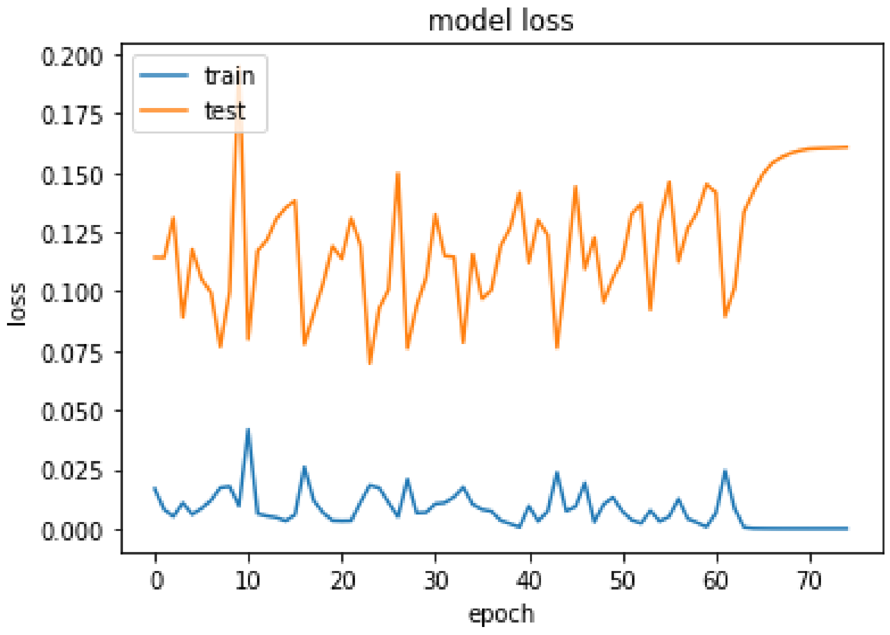


FIGURE 4. Loss Curve

Based on the Figure 4 show that loss in training process in 0.010 until 0.025. in testing process loss stay in 0.075 until 0.2. it show that the process is still good in classification. But it seems that the value is poor in amalysis. The value of the resulting ROC curve is equal to 97% as shown in Figure 5.

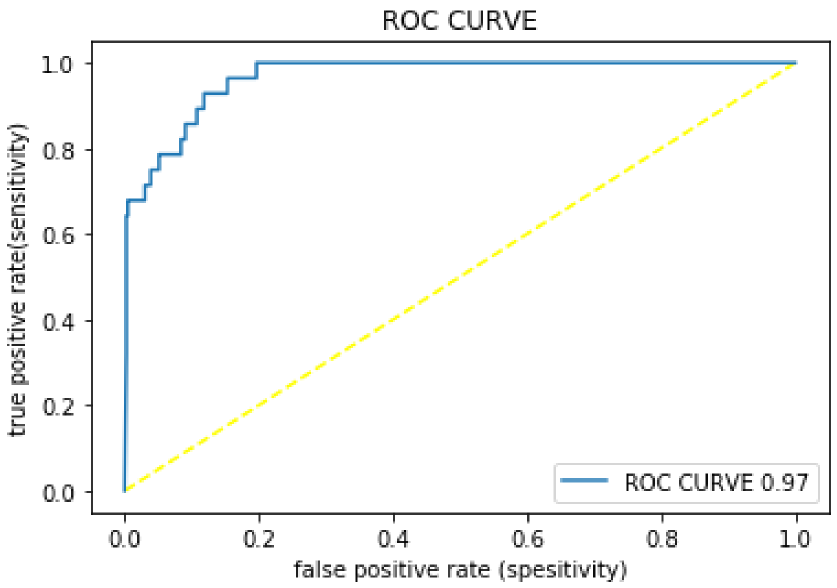


FIGURE 5. ROC Curve

Confusion matrix value is comparison of predict label and actual from confusion matrix. The results of the confusion matrix calculation based on trials in the Jupiter notebook show good results, as shown in Table 3.

TABLE 3.
Confusion Matrix Value

Confusion Matrix Value			
True Positive (TP)		False Positive (FP)	
Reality	: Malware	Reality	: Benign
Predicted	: Malware	Predicted	: Malware
Number of TP result	: 1113	Number of TP result	: 14
False Negative (FN)		True Negative (TN)	
Reality	: Malware	Reality	: Benign
Predicted	: Benign	Predicted	: Benign
Number of TP result	: 0	Number of TP result	: 1098

Based on the Table 3 shows that obtained True positive is 1113, True Negative is 1098, False positive is 14 and False Negative 0. The next step is calculate the precision and recall. The value of the Precision and Recall results are very good. As shown in Table 4.

TABLE 4.
Precision and Recall Value

Precision and Recall	
Precision	99.6%
Recall	99.4%

The precision value obtained is 99.6% while for the recall value is 99.4%. To get a curve of precision and a good recall is to calculate from the formula precision and recall. A confusion matrix for two classes is used binary crossentropy is shown four analysis. The first is true positive value is normal file with correctly classification. False positive value is from malware file detected correct. True negative is malware file detected correctly in classification and False negative is malware data detected normal file and normal file detected malware. To see the precision and recall curves can be seen in Figure 6 with average precision 0.60

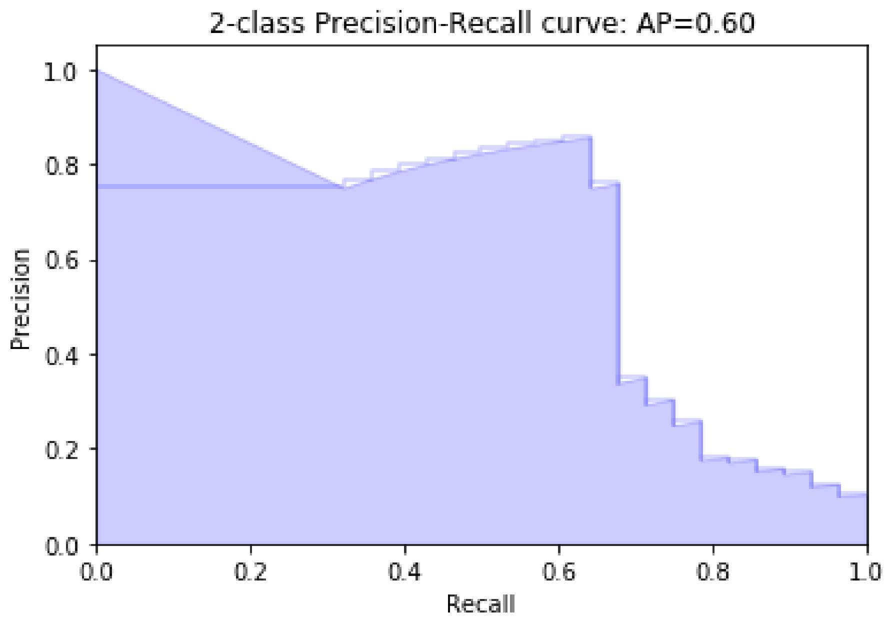


FIGURE 6. Precision and Recall Curve

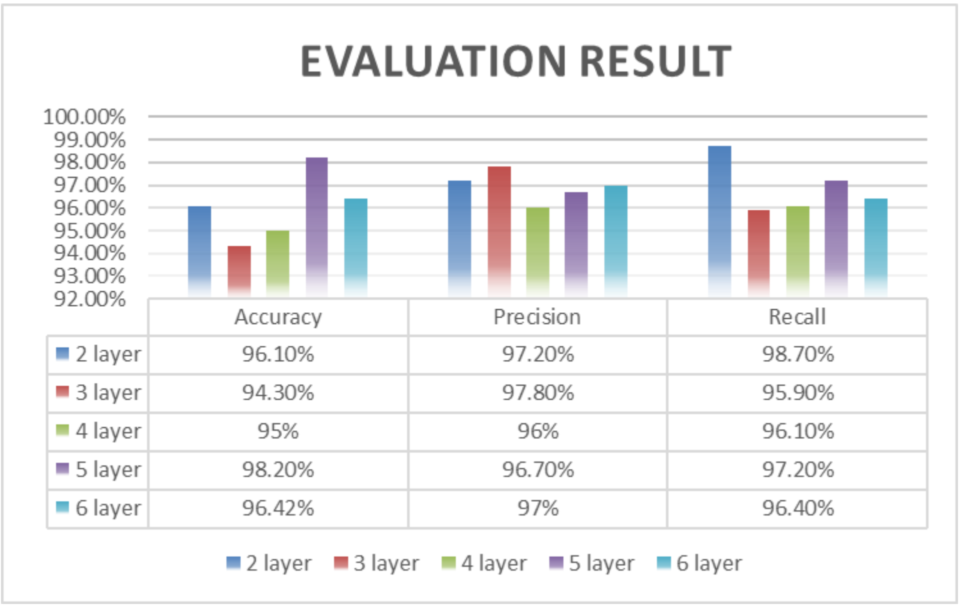
The development of the DNN model that is implemented is very capable of detecting malware. The DNN model produces the best performance. But the data processed in this case includes imbalanced data and preprocessing stage needs to be done before the training, testing and data classification stages are carried out. This is obtained from the results of the accuracy and confusion matrix that initially overfitting, then by using SMOTE the data increases. The value of the accuracy and confusion matrix results are very good. As shown in Table 5 below.

TABLE 5.
Evaluation of Results

Hidden Layer	Accuracy	Precision	Recall
2 Layer	96.1%	97.2%	98.7%
3 Layer	94.3%	97.8%	95.9%
4 Layer	95%	96%	96.1%
5 Layer	94.9%	96.7%	97.2%
6 Layer	99.4%	99.6%	99.4%

From the results of experiments with several hidden layers can be seen that the more hidden layers, the better the level of accuracy, precision, and recall. This proves that the classification results successfully group files between malware and benign. The dataset is imbalanced. The results of 99% precision and 99.4% recall.

To see the evaluation result curves of accuracy, precision, and recall can be seen in Figure 7.



	Accuracy	Precision	Recall
🟦 2 layer	96.10%	97.20%	98.70%
🟥 3 layer	94.30%	97.80%	95.90%
🟩 4 layer	95%	96%	96.10%
🟪 5 layer	98.20%	96.70%	97.20%
🟦 6 layer	96.42%	97%	96.40%

FIGURE 7. Evaluation Result Graph

5. CONCLUSION

Preprocessing using SMOTE for imbalanced data and the DNN classification method is very reliable for determining the correctness and classification of malware and benign. This is proven by the accuracy of 99.4%. Changes in layer level greatly affect the level of accuracy when training data, and therefore must be careful in changing the layer level. By using good and optimal training data, a subset of the training data will also produce a good classification. However, because data is imbalanced, the results of precision and recall are the most important in terms of performance with a value of 99% precision and 99.4% recall.

REFERENCES

[1] Nix. R and Zhang. J, "Classification of Android Apps and Malware Using Deep Neural Networks" in *IEEE*, 2017, pp. 1871-1878.
[2] Bulut. I and Yavuz. A "Mobile Malware Detection Using Deep Neural Network" in *IEEE*, 2017., 978-1-5090-6494-6/17.
[3] Y. Du, J. Wang and Q. Li, "An android malware detection approach using community structures of weighted function call graphs" in *IEEE*, 2017, pp. 17478-17486.
[4] L. Wei, W. Luo, J. Weng et al, "Machine Learning-Based Malicious Application Detection of Android" in *IEEE*, vol 5, 2017, pp. 25591-25601.

[5] F. Martinelli, F. Marulli, F. Mercaldo "Evaluating Convolutional Neural Network for Effective Mobile Malware Detection" in *Proceedia Computer Science Elsevier,* vol 112, 2017, pp. 2372-2381.
[6] Karbab E. B, Debbabi, Derhab A., and Mouheb D, "Maldozer: Automatic Framework for Android Malware Detection Using Deep Learning" *in Proceeding of the Fifth Annual DFRWS Europe Elsevier,* 2018, pp. S48-S59.
[7] http://www.malgenomeproject.org/
[8] Verbiest N, Ramentol E, Cornelis C, Herrera F, "Preprocessing Noisy Imbalanced Dataset using SMOTE enhanced with Fuzzy Rough Prototype Selection" in *Elsevier,* 2014., pp. 511-517.
[9] D. Stiawan, Sandra. S, Alzahrani E, Budiarto. R, "Comparative analysis of K-means method and Naïve Bayes method for brute force attack visualization". IEEE, pp. 177-182, 2017.
[10] S. Nurmaini, N. L. Husni, A. Silvia, E. Prihatini, I. Yani, "Swarm Intelligence in Bio-Inspired Perspective: A Summary" no. June, pp.105-120, 2018.