Computer Science and Information Technologies
Vol. 6, No. 1, March 2025, pp. 68~79
ISSN: 2722-3221, DOI: 10.11591/csit.v6i1.p68-79

Detection of android malware with deep learning method using convolutional neural network model

Reza Maulana[1], Deris Stiawan[1], Rahmat Budiarto[2]
[1]Department of Computer Science, Sriwijaya University, Palembang, Indonesia
[2]College of Computing and Information, Al-Baha University, Al Baha, Saudi Arabia

Article Info

Article history:

Received Nov 19, 2024
Revised Feb 6, 2025
Accepted Feb 18, 2025

Keywords:

Android malware
Classification
Convolutional neural network
Deep learning
Pattern recognition

ABSTRACT

Android malware is an application that targets Android devices to steal crucial data, including money or confidential information from Android users. Recent years have seen a surge in research on Android malware, as its types continue to evolve, and cybersecurity requires periodic improvements. This research focuses on detecting Android malware attack patterns using deep learning and convolutional neural network (CNN) models, which classify and detect malware attack patterns on Android devices into two categories: malware and non-malware. This research contributes to understanding how effective the CNN models are by comparing the ratio of data used with several epochs. We effectively use CNN models to detect malware attack patterns. The results show that the deep learning method with the CNN model can manage unstructured data. The research results indicate that the CNN model demonstrates a minimal error rate during evaluation. The comparison of accuracy, precision, recall, F1 Score, and area under the curve (AUC) values demonstrates the recognition of malware attack patterns, reaching an average of 92% accuracy in data testing. This provides a holistic understanding of the model's performance and its practical utility in detecting Android malware.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Deris Stiawan
Department of Computer Science, Sriwijaya University
Srijaya Negara Street, Palembang 30139, Indonesia
Email: deris@unsri.ac.id

1. INTRODUCTION

Android malware specifically targets devices running the Android operating system [1]. They can take various forms, including viruses, trojans, adware [2], and spyware [3]. Android malware can infect devices in various ways [4], such as by downloading malicious applications from third-party applications, visiting compromised websites, or exploiting vulnerabilities in the device's operating system. Once installed, malware can steal personal information [5], send unwanted SMS messages [6], make unauthorized phone calls [7], or perform other malicious actions [8].

The types of Android malware continue to grow and develop [9]–[12], with many new families and variants emerging. Cybercriminals are increasingly focusing on mobile malware, with many new families and variants emerging [13]. We expect this trend to persist in the future as cybercriminals increasingly target mobile devices. One method that Android malware uses is keylogging [14], [15], where it records every keystroke made on the device, including passwords, and other sensitive information. Another method is scraping [14], where malware collects information from various sources such as device contacts, calendars, and call logs, as well as from applications installed on the device. Another method is data exfiltration [16], [17], where malware transmits the gathered information to a remote server for analysis and malicious use [15].

Recent years have seen the development of deep learning methods for classifying malware attack patterns [18]–[24]. In several of these studies, malware detection on Android using deep learning methods has become a research hotspot [12], [13]; however, it lacks detail and comprehensiveness so this research continues to be developed nowadays [20]. The increasing threat of malware in cyberspace has made malware detection using deep learning methods a significant research field in recent years. A major challenge in malware detection is the ability of malware to evolve quickly [25], making it difficult for traditional detection methods to keep up. Deep learning methods have proven to be very effective in detecting malware [18], [26] due to their ability to automatically learn features from large data and adapt to emerging threats.

One of the main problems in deep learning-based malware detection is the need for large and diverse datasets to train models effectively. This can be a challenge, especially when dealing with the ever-evolving nature of malware, which can lead to a lack of relevant data for training. Additionally, the complexity of malware can make it difficult to identify the most effective features to extract from data, which can impact model performance. Therefore, this paper applies deep learning methods using convolutional neural network (CNN) to detect Android malware attack patterns.

This research will demonstrate the practical application of CNN to classify Android applications into malware and non-malware. This showcases the capability of CNN to effectively handle the problem of Android malware detection, which involves unstructured data. A key contribution of the study is the analysis of how different training data ratios and epochs impact the CNN model's effectiveness. This provides valuable insights into optimizing CNN performance for malware detection. The study highlights the ability of CNNs to process unstructured data (e.g., data extracted from Android applications) and identify complex patterns that distinguish malware from non-malware, emphasizing the suitability of deep learning for cybersecurity tasks. Tackles the issue of insufficient and imbalanced malware datasets by employing techniques of data augmentation, by proposing a new methodology for generating a diverse and representative dataset of Android malware samples. This contribution aids in mitigating the dataset limitations faced by other malware detection systems and measuring uses a detailed evaluation framework, including metrics such as accuracy, precision, recall, F1 score, and area under the curve (AUC).

2. METHOD

2.1. State of the art

Android malware detection with deep learning, has received important attention in recent years [27]–[29] due to the increasing sophistication of malware attacks on Android devices. Deep learning algorithms, especially those, which are based on recurrent neural networks (RNN), and CNN, have been successfully applied to detect malware in Android applications [30]. A combination of RNN and CNN harnesses the power of machine learning to analyze complex patterns in Android app behaviour, such as application programming interface (API) calls and system interactions, to identify malicious software.

One of the deep learning methods CNN is currently widely used in the fields of classification and pattern recognition, due to CNN's ability to perform feature extraction and classification in one network so that it can be implemented for any case. The approach using the CNN method to detect Android malware is very effective in analyzing structured data [31], such as API calls and system interactions, which can be represented as a series of vectors [24]. This method involves training a CNN model on a dataset of labelled Android apps, where the labels indicate whether the app is malicious or non-malicious. The trained model can then be used to classify new, unseen applications as malicious or non-malicious based on their behaviour.

Deep learning models have achieved high accuracy in detecting malware [32], [33], especially when compared to machine learning methods [34]. Deep learning models can process large amounts of data quickly, making them suitable for real-time malware detection on Android devices [34], [35]. The quality of training data is critical to the effectiveness of deep learning models. High-quality data with accurate labels is critical to achieving good performance. Integrating deep learning-based malware detection with existing security systems can improve the overall security of Android devices [34].

Based on research that has been conducted, Android malware detection using deep learning methods has shown promising results in detecting malware on Android devices. The use of CNN is effective in analyzing complex patterns in Android application behaviour [36]. The deep learning method in classifying Android malware involves the use of deep learning models to identify and separate malware applications from correct applications [37]–[39]. The static and dynamic features of malware applications and non-malware applications were combined, and a deep learning model with long short-term memory (LSTM) architecture was developed to identify malware [40]. In this research using the deep learning neural network method [38], the results show that the model developed has an accuracy of 99% precision and 99.4% recall. It means that deep learning methods is the best solution for recognizing Android malware patterns.

The Mapping research related to Android malware classification by the VOS Viewer application is shown in Figure 1. In Figure 1 the metadata is taken from dimensions so that 7 clusters are formed within the

mapping of research related to deep learning topics. In these 7 clusters, the highest researched from 2019 to 2023 is research related to malware and Android, which is shown in the blue and light blue clusters, meanwhile, Android malware detection is in the red cluster, and one of the least research focuses is the use of deep learning methods in Android malware classification. This means the research of the CNN method in deep learning is still in demand of research until this year [41], [42].

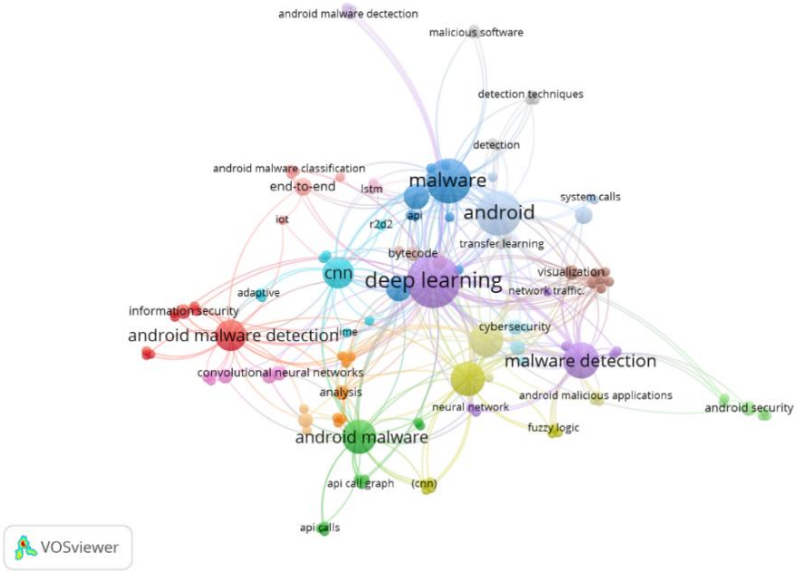


Figure 1. Network visualization android malware classification

2.2. Research framework

At this phase, the deep learning method uses a simple CNN algorithm, which will be applied to image processing. This research focuses on analyzing attack patterns that occur on Android devices by grouping the malware variants into Variant X (malware), and Variant Y (non-malware), then applying the deep learning method using the CNN algorithm to detect malware attack patterns. The steps of this research are shown in Figure 2.

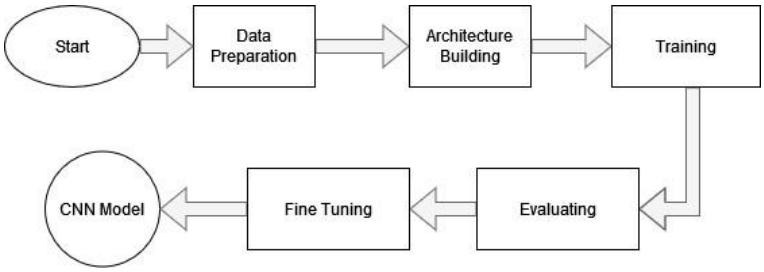


Figure 2. Research steps

2.2.1. Data preparation

In this phase, the data is collected in the form of image files which contain malware and benign attack patterns detected on Android devices, the dataset has been converted from a binary file into a grayscale image 2D as seen in Figure 3. The binary file is converted to a grayscale image, using a hexadecimal representation of the binary content, and then the file is converted into an image in PNG format. The dataset for malware attack patterns was obtained via Google Dataset with a total of 6,012 data samples with dimensions of 256×256 pixels. The data was divided into 3,000 sample data for malware and 3,012 sample data for non-malware.

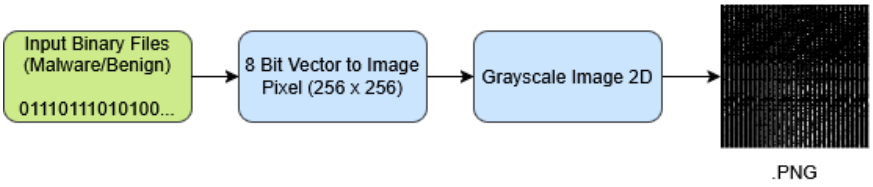


Figure 3. Step binary file to convert image

2.2.2. Data preprocessing

Preparing the data by importing the dataset folder on the storage drive, and dividing the data based on the ratio of training and testing data with comparison 80:20. Then data resizing to 128×128 pixels. The last data saved will be augmented to increase the number of datasets and provide a wide variety of Android malware patterns. Data augmentation is a technique to increase the variety of training data by modifying the original images so that the model can learn better and improve generalization. The augmentation in data preprocessing steps can be shown in Figure 4.

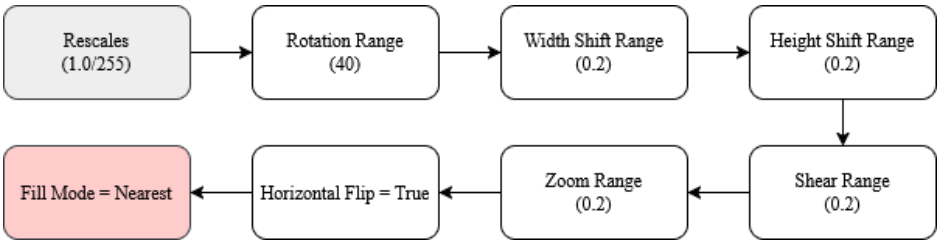


Figure 4. Augmentation in data preprocessing

The augmentation data using an image data generator for each parameter step by step in Figure 4 can be explained in detail as follows.

- Rescale=1.0/255: Used for pixel normalization. The initial image pixel values are 0-255 (standard for RGB images). With rescale=1.0/255, the pixel values are converted to the range 0-1, which often makes model training more stable and converges quickly.
- Rotation_range=40: Sets a random rotation of the image within a certain degree range. In this case, the image will be rotated randomly between -40 and +40 degrees.
- Width_shift_range=0.2: Performs a horizontal shift on the image. It means the image can be shifted horizontally 20% from the original image width.
- Height_shift_range=0.2: Performs a vertical shift on the image. The image can be shifted vertically 20% from the original image height.
- Shear_range=0.2: Provides a shearing transformation on the image. Shearing shifts one part of the image in a certain direction, producing a skewed effect. 0.2 means the image will be changed randomly within a certain shearing angle.
- Zoom_range=0.2: Sets a random zoom on the image, zooming in or out of the image. 0.2 means the image can be zoomed in or out up to 20% of its original size.
- Horizontal_flip=True: Performs a horizontal flip on the image randomly, so the image can be mirrored horizontally. Useful for making the model unbiased to the orientation of the image.
- Fill_mode='nearest': Determines how to fill empty areas that may appear due to rotation, shifting, or zooming.

2.2.3. Architecture building

In the model selection stage, namely deep learning by adding convolutional layers, pooling layers, fully-connected layers, and dropout layers in the CNN model architecture used. So that the data will form a CNN classification model based on the specified data labels. The input data will be mapped into 2 (two) groups with the dataset divided into malware and non-malware labels. Next, the data that has been formed will be added to a convolution layer which is the core network for finding patterns of malware attacks, so that a one-dimensional vector will be formed. Next, the data will enter the pooling layer which is used to reduce the dimensions of the feature maps produced by the previous convolutional layer. So, it will reduce the

occurrence of overfitting. Then the next data will be forwarded to the fully connected layer to produce fully classified data output. The simple model of CNN used in this research is shown in Figure 5.

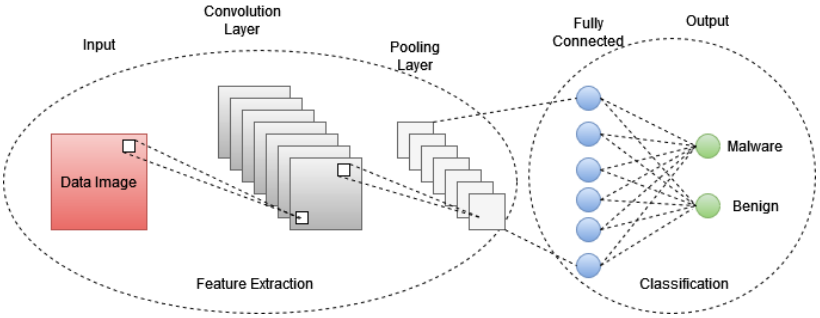


Figure 5. Simple architecture of CNN

The start step is adding a convolutional layer from the input shape with 32 filters of size 3×3. Rectified linear unit (ReLu) activation is used to add non-linearity. Input shape is used only in the first layer to determine the shape of the model input. A pooling layer that reduces the dimensionality of the features by taking the maximum value from a 2×2 area in the previous layer's output, helps reduce computational complexity. Next, add flattening to convert the 2D output of the previous layer into one dimension. Then add a fully-connected layer with 128 units and activate ReLu. This layer captures the pattern of features that have been extracted by the convolutional layer. Adding dropout with a probability of 50% is used to prevent overfitting by reducing the model's dependence on certain neurons. This function returns a simple CNN model that can be used for image classification. The step of the simple CNN model applied is shown in Figure 6.

```
def build_simple_cnn(input_shape=(64, 64, 3), num_classes=1):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),

        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='sigmoid' if num_classes == 1 else 'softmax')
    ])
    return model
```

Figure 6. CNN models applied

2.2.4. Training and testing

In this phase, data will separate in 80:20 which contains 4,809 data testing and 1,203 data training belonging to 2 classes of malware and non-malware patterns. In the training phase is used train generator. Training data generators that provide image data in batches, typically use “ImageDataGenerator” from augmentation. The training phase uses the model.fit () function, which trains the model and records the loss and accuracy values, and then evaluation is carried out using the validation generator function on the test data to see the actual model performance.

2.2.5. Fine tuning

Fine-tuning is adjusting a pre-trained model for optimal performance on a new task or dataset. It is a subset of transfer learning techniques, where a model learned on a large dataset is reused for a different task with a smaller or specialized dataset. The stages in fine-tuning carried out in this study are starting with the pre-trained model, and then freezing the Initial layer. The initial layers of the pre-trained model often learn

basic features such as edges, corners, and textures that are generally applicable to many types of images. Then adjust the final layer, train with a lower learning rate and evaluate the results. During fine-tuning, the model is trained with a smaller learning rate to make subtle adjustments to the parameters without destroying previously learned features. The fine-tuning step in this research is shown in Figure 7.

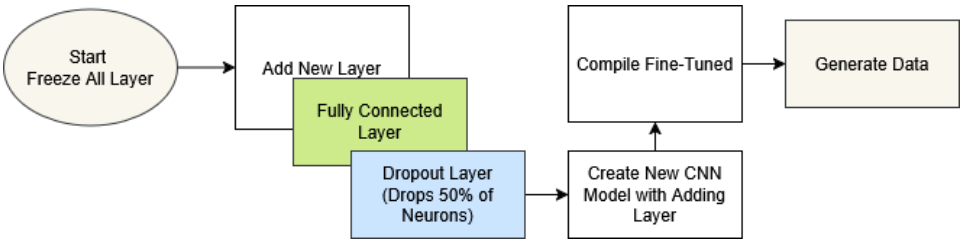


Figure 7. Fine tuning step

2.2.6. Evaluation

In this evaluation stage, the performance of the CNN classifier is measured by using a confusion matrix as a measurement metric. Measurement metrics in deep learning are used to assess a model's performance throughout training and evaluation. They provide insights into how effectively the model learns and adapts to new data. In this stage, the values of accuracy, precision, recall, and F1 score will be measured by calculating the true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The measurement metrics from evaluation stage can be found in (1) to (4).

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\ Score = 2\ x\ \frac{(Recall\ x\ Precision)}{(Recall + Precision)} \quad (4)$$

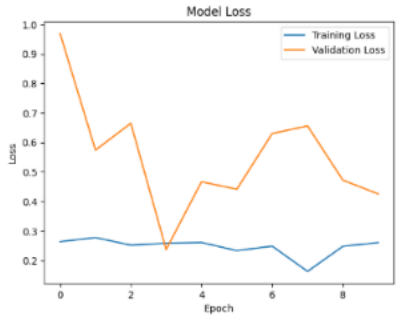
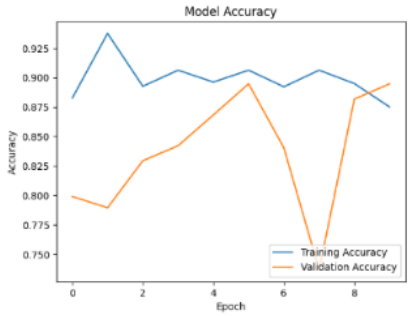
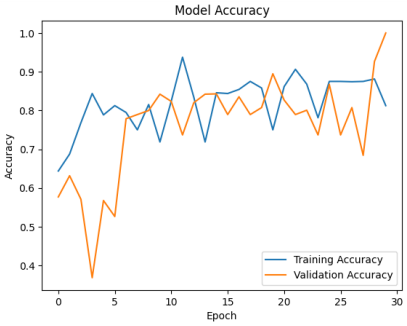
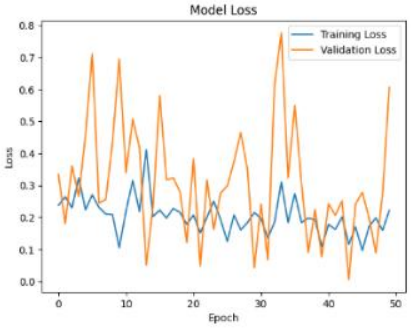
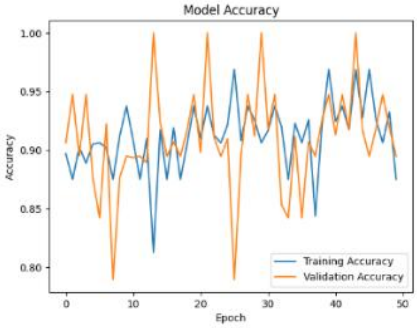
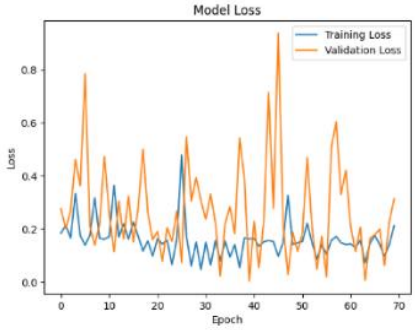
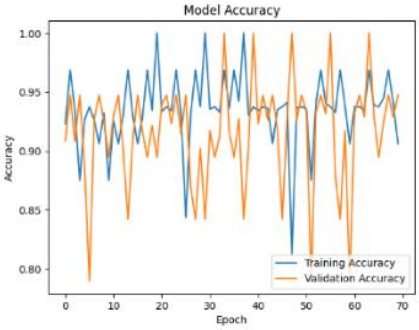
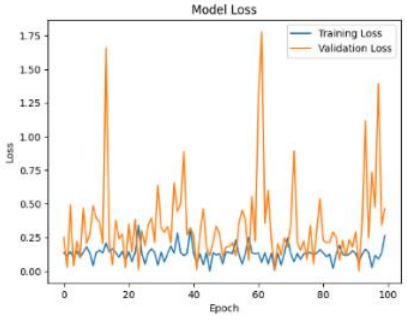
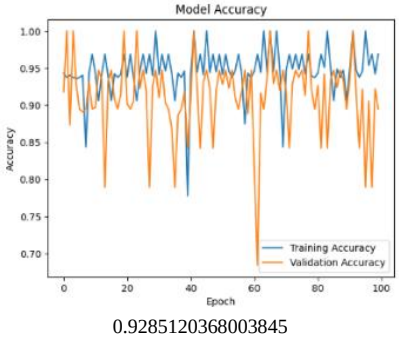
3. RESULTS AND DISCUSSION

The results and analysis of the data will be discussed in this part. The CNN algorithm has been applied to classify malware (X) and non-malware (Y) variants. After implementing the algorithm, the system that has been created will be tested. Testing on the system aims to measure the performance of the model implemented on the dataset. In the testing stages, a different number of epochs will also be used when training the previously designed model.

The number of epochs used is 10, 30, 50, 70 and 100. The simple CNN model applies additional layers to deep learning by providing several basic layers, namely convolutional layers (conv), pooling layers (pool), fully connected layers and dropout layers using ReLu. In the processing data stage, the augmented dataset was implemented. The data transformed to 128×128 from 256×256 pixels. The next stage is calling the model function, i.e.: Simple CNN by applying the neural network cross-entropy loss function and calling the Adam optimizer function. Then the next stage is to carry out training data by entering the epoch value several times as a comparison for better accuracy. In the validation stage of the testing results, several functions are used to determine accuracy values. Several functions are used, including to display comparisons of total loss, accuracy, confusion matrix, precision values, accuracy, recall, F1 score, and receiver operating characteristic (ROC)-AUC.

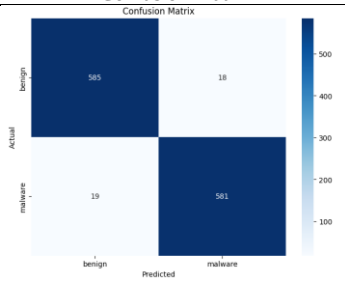
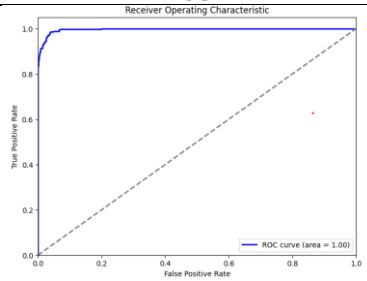
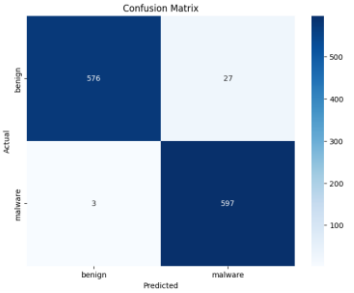
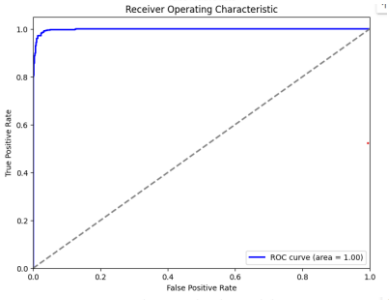
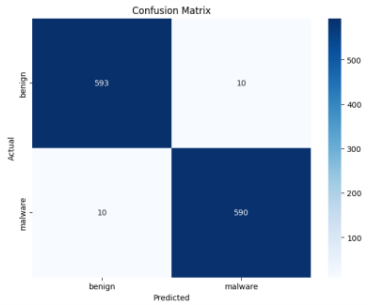
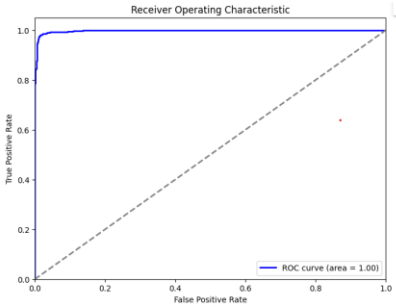
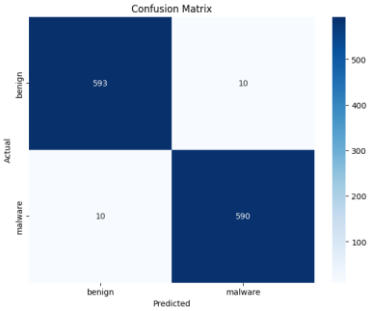
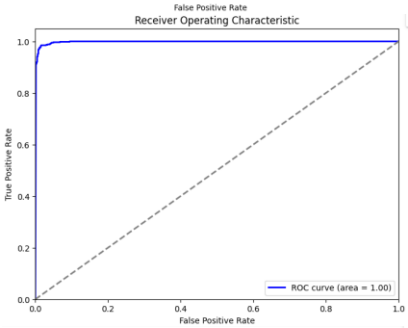
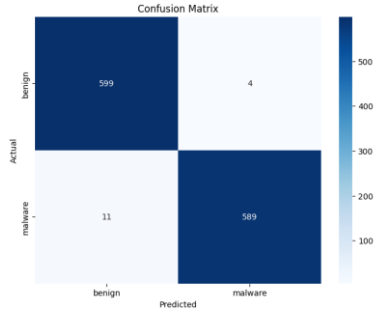
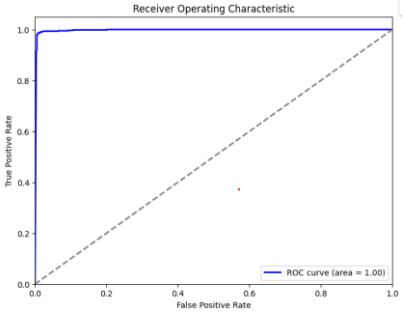
The accuracy of the predictions in relation to the total testing data was assessed in this study using the accuracy and loss variables, as indicated in Table 1. While the recall is generated to test the CNN model's error in identifying patterns other than malware, precision is generated to measure the CNN model's mistake in predicting the proper object. It is anticipated that the model in this study will have greater recall and accuracy values than precision values. Google Collabs software was used as a tool to analyze previously obtained datasets which used Python programming language, connected to the runtime with the T4 GPU for running the process quickly and setting the RAM using NVIDIA graphics. After running the process with the simple CNN model, evaluate the model by measuring validation loss and accuracy in any epoch as shown in Table 1.

Table 1. Result and comparison between validation loss and validation accuracy

Total Epoch	Validation Loss	Validation Accuracy
10	 0.5306247472763062	 0.864505410194397
30	 0.25273627042770386	 0.9268495440483093
50	 0.2981289029121399	 0.916043221950531
70	 0.2427741140127182	 0.9318370819091797
100	 0.26761937141418457	 0.9285120368003845

From Table 1, the validation loss and accuracy results show the different values within any epochs (10, 30, 50, 70, and 100), with the best validation start in 30 epochs with a validation loss value of 0.2 and an accuracy score of 0.93. In this case, the tested dataset has no underfitting or overfitting so, it can be used for better system deployment. The result showed that the simple CNN model can still detect malware attack patterns through data classification for malware and non-malware classes. The comparison of the confusion matrix and ROC graph in any epoch is shown in Table 2.

Table 2. Result of confusion matrix and ROC

Total Epoch	Confusion Matrix	ROC
10		
30		
50		
70		
100		

From Table 2, The confusion matrix shows scores TP, FN, TN, and FP in malware prediction has good balance in classification performance for benign and malware samples, with the error rates distributed proportionally. The high recall suggests that the model prioritizes detecting malware over avoiding false positives. This is typically ideal for malware detection systems. For the comparison of accuration score in any epoch can be seen in Figure 8.

From Figure 8, Accuracy score starts at 86.45% with 10 epochs and increases to 93.18% at 70 epochs, at 100 epochs accuracy slightly drops to 92.85%, indicating potential overfitting beyond 70 epochs. Precision remains high across all epochs, ranging from 95.67% (30 epochs) to 99.33% (100 epochs). Precision improves significantly after 30 epochs and reaches its highest value at 100 epochs. Recall improves from 96.83% (10 epochs) to 99.5% (30 epochs), peaks, and then stabilizes at 98.33% (70 epochs). This suggests the model performs well in identifying all true positives consistently as training progresses. The F1 score follows a similar pattern as accuracy. It increases significantly, peaking at 98.33% (70 epochs). A slight improvement occurs at 100 epochs (98.74%), driven by precision improvements. ROC-AUC values are consistently very high (above 99.65%) across all epochs, indicating the model's strong ability to distinguish between classes. The highest value (99.8%) is observed at 30 epochs, with marginal variations thereafter.

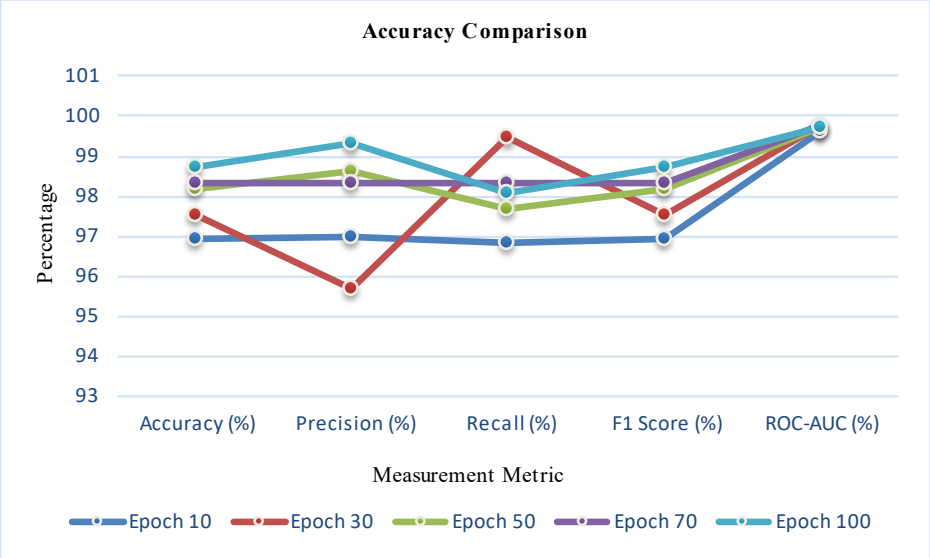


Figure 8. Comparison of accuration score

4. CONCLUSION

The deep learning method applied with the simple CNN model to detect Android malware patterns on Android devices is highly recommended with unstructured data. The pattern of malware attacks is classified with an average accuracy rate of 92%, the error percentage only reaches less than 10%. The CNN model applied is effective in classifying and detecting malware attack patterns. This establishes the effectiveness of the CNN model in detecting Android malware attack patterns and demonstrates its potential as a reliable malware detection method. Malware patterns on Android devices can be determined by adding other methods after the classification results are obtained using the simple CNN architecture. The CNN architecture with the correct composition of additional convolutional layers, pooling layers, fully connected layers, and ReLu will produce good accuracy. The research provides a robust CNN-based framework for detecting Android malware. It contributes to understanding the effectiveness of CNN models in handling unstructured data and optimizing performance through data ratio and epoch analysis. Additionally, it offers a comprehensive evaluation framework and achieves high detection accuracy, making it a valuable contribution to Android malware detection. By addressing the evolving nature of Android malware, this research provides a scalable and robust framework for periodic updates to cybersecurity tools. Detection of Android malware patterns can also be determined by adding other deep learning models to obtain better accuracy results. The classification results can be used as benchmark data in building system defences in cyber security, ensuring their relevance in combating emerging threats.

ACKNOWLEDGEMENTS

The authors thank the Computer Science Magister Program at Sriwijaya University, Palembang, South Sumatra, Indonesia, and all the support from the academic community of the Computer Science Magister Program.

FUNDING INFORMATION

No funding is involved in this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Reza Maulana	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	
Deris Stiawan	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓	✓	
Rahmat Budiarto	✓	✓								✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article [and/or its supplementary materials].

REFERENCES

[1] T. Sharma and D. Rattan, "Malicious application detection in android-A systematic literature review," *Computer Science Review*, vol. 40, 2021, doi: 10.1016/j.cosrev.2021.100373.

[2] P. Sreekumari, "Malware detection techniques based on deep learning," *Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020*, pp. 65–70, 2020, doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00023.

[3] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based android malware detection and variant classification," *Computational Intelligence*, vol. 38, no. 5, pp. 1748–1771, 2022, doi: 10.1111/coin.12532.

[4] A. Razgallah, R. Khoury, S. Hallé, and K. Khanmohammadi, "A survey of malware detection in Android apps: Recommendations and perspectives for future research," *Computer Science Review*, vol. 39, 2021, doi: 10.1016/j.cosrev.2020.100358.

[5] S. Sharma, R. Kumar, and C. R. Krishna, "RansomAnalysis: The evolution and investigation of android ransomware," *Proceedings of International Conference on IoT Inclusive Life (ICIIL 2019)*, vol. 116, pp. 33–41, Apr. 2020, doi: 10.1007/978-981-15-3020-3_4.

[6] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "EfficientNet convolutional neural networks-based Android malware detection," *Computers and Security*, vol. 115, 2022, doi: 10.1016/j.cose.2022.102622.

[7] S. Sharma, R. Kumar, and C. Rama Krishna, "A survey on analysis and detection of Android ransomware," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 16, 2021, doi: 10.1002/cpe.6272.

[8] M. Talal *et al.*, "Comprehensive review and analysis of anti-malware apps for smartphones," *Telecommunication Systems*, vol. 72, no. 2, pp. 285–337, 2019, doi: 10.1007/s11235-019-00575-7.

[9] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning," *Ad Hoc Networks*, vol. 101, 2020, doi: 10.1016/j.adhoc.2020.102098.

[10] L. Chen, C. Xia, S. Lei, and T. Wang, "Detection, traceability, and propagation of mobile malware threats," *IEEE Access*, vol. 9, pp. 14576–14598, 2021, doi: 10.1109/ACCESS.2021.3049819.

[11] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of Android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020, doi: 10.1109/ACCESS.2020.3006143.

[12] S. Peng, L. Cao, Y. Zhou, J. Xie, P. Yin, and J. Mo, "Challenges and trends of Android malware detection in the era of deep learning," *Proceedings - 2020 IEEE 8th International Conference on Smart City and Informatization, iSCI 2020*, pp. 37–43, 2020, doi: 10.1109/iSCI50694.2020.00014.

[13] N. Lachtar, D. Ibdah, and A. Bacha, "Toward mobile malware detection through convolutional neural networks," *IEEE Embedded Systems Letters*, vol. 13, no. 3, pp. 134–137, 2021, doi: 10.1109/LES.2020.3035875.
[14] D. Waterson, "Managing endpoints, the weakest link in the security chain," *Network Security*, vol. 2020, no. 8, pp. 9–13, 2020, doi: 10.1016/S1353-4858(20)30093-3.
[15] J. Hubbard, G. Bendiab, and S. Shiaeles, "IPASS: a novel open-source intelligence password scoring system," *Proceedings of the 2022 IEEE International Conference on Cyber Security and Resilience, CSR 2022*, pp. 90–95, 2022, doi: 10.1109/CSR54599.2022.9850311.
[16] K. Chung, P. Cao, Z. T. Kalbarczyk, and R. K. Iyer, "StealthML: data-driven malware for stealthy data exfiltration," *Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience, CSR 2023*, pp. 16–21, 2023, doi: 10.1109/CSR57506.2023.10224946.
[17] J. King, G. Bendiab, N. Savage, and S. Shiaeles, "Data exfiltration: methods and detection countermeasures," *Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience, CSR 2021*, pp. 442–447, 2021, doi: 10.1109/CSR51186.2021.9527962.
[18] O. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021, doi: 10.1109/ACCESS.2021.3089586.
[19] A. Abusnaina *et al.*, "DL-FHMC: deep learning-based fine-grained hierarchical learning approach for robust malware classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3432–3447, 2022, doi: 10.1109/TDSC.2021.3097296.
[20] Z. Wang, Q. Liu, and Y. Chi, "Review of Android malware detection based on deep learning," *IEEE Access*, vol. 8, pp. 181102–181126, 2020, doi: 10.1109/ACCESS.2020.3028370.
[21] Y. Fang, Y. Gao, F. Jing, and L. Zhang, "Android malware familial classification based on DEX file section features," *IEEE Access*, vol. 8, pp. 10614–10627, 2020, doi: 10.1109/ACCESS.2020.2965646.
[22] I. U. Haq, T. A. Khan, and A. Akhunzada, "A dynamic robust DL-based model for Android malware detection," *IEEE Access*, vol. 9, pp. 74510–74521, 2021, doi: 10.1109/ACCESS.2021.3079370.
[23] N. Zhang, Y. an Tan, C. Yang, and Y. Li, "Deep learning feature exploration for Android malware detection," *Applied Soft Computing*, vol. 102, 2021, doi: 10.1016/j.asoc.2020.107069.
[24] A. Pektaş and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," *Soft Computing*, vol. 24, no. 2, pp. 1027–1043, 2020, doi: 10.1007/s00500-019-03940-5.
[25] O. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.
[26] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for Malware detection," *Entropy*, vol. 23, no. 3, 2021, doi: 10.3390/e23030344.
[27] M. Chen, Q. Zhou, K. Wang, and Z. Zeng, "An Android malware detection method using deep learning based on multi-features," *2022 IEEE International Conference on Artificial Intelligence and Computer Applications, ICAICA 2022*, pp. 187–190, 2022, doi: 10.1109/ICAICA54878.2022.9844642.
[28] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "A deep learning based android malware detection system with static analysis," *HORA 2022 - 4th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, 2022, doi: 10.1109/HORA55278.2022.9800057.
[29] A. Alzubaidi, "Sustainable android malware detection scheme using deep learning algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 12, pp. 860–867, 2021, doi: 10.14569/IJACSA.2021.01212104.
[30] A. Lakshmanarao and M. Shashi, "Android malware detection with deep learning using RNN from opcode sequences," *International Journal of Interactive Mobile Technologies*, vol. 16, no. 1, pp. 145–157, Jan. 2022, doi: 10.3991/IJIM.V16I01.26433.
[31] Y. Liu, G. Li, and Z. Jin, "Call graph based android malware detection with CNN," *Communications in Computer and Information Science*, vol. 861, pp. 72–82, 2019, doi: 10.1007/978-981-15-0310-8_5.
[32] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, no. 4, pp. 725–738, 2022, doi: 10.1007/s10207-022-00579-6.
[33] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1563–1578, 2021, doi: 10.1109/TIFS.2020.3025436.
[34] I. Almomani, A. Alkhayer, and W. El-Shafai, "An automated vision-based deep learning model for efficient detection of android malware attacks," *IEEE Access*, vol. 10, pp. 2700–2720, 2022, doi: 10.1109/ACCESS.2022.3140341.
[35] R. M. Sharma and C. P. Agrawal, "MH-DLdroid: a meta-heuristic and deep learning-based hybrid approach for android malware detection," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 4, pp. 425–435, 2022, doi: 10.22266/ijies2022.0831.38.
[36] M. S. Akhtar and T. Feng, "Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time," *Symmetry*, vol. 14, no. 11, 2022, doi: 10.3390/sym14112308.
[37] N. Afifah and D. Stiawan, "The implementation of deep neural networks algorithm for malware classification," *Computer Engineering and Applications Journal*, vol. 8, no. 3, pp. 189–202, 2019, doi: 10.18495/comengapp.v8i3.294.
[38] R. B. Hadiprakoso, I. K. S. Buana, and Y. R. Pramadi, "Android malware detection using hybrid-based analysis deep neural network," *2020 3rd International Conference on Information and Communications Technology, ICOIACT 2020*, pp. 252–256, 2020, doi: 10.1109/ICOIACT50329.2020.9332066.
[39] H. Il Kim, M. Kang, S. J. Cho, and S. Il Choi, "Efficient deep learning network with multi-streams for android malware family classification," *IEEE Access*, vol. 10, pp. 5518–5532, 2022, doi: 10.1109/ACCESS.2021.3139334.
[40] D. Stiawan *et al.*, "An improved LSTM-PCA ensemble classifier for SQL injection and XSS attack detection," *Computer Systems Science and Engineering*, vol. 46, no. 2, pp. 1759–1774, 2023, doi: 10.32604/csse.2023.034047.
[41] T. L. Nikmah, J. Jumanto, B. Prasetiyo, N. Fitriani, and M. A. Muslim, "Deep learning model implementation using convolutional neural network algorithm for default P2P lending prediction," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 9, no. 3, pp. 802–809, Aug. 2023, doi: 10.26555/jiteki.v9i3.26366.
[42] F. Fatimatuzzahra, L. Lindawati, and S. Soim, "Development of convolutional neural network models to improve facial expression recognition accuracy," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 10, no. 2, pp. 279–289, Jun. 2024, doi: 10.26555/jiteki.v10i2.28863.

BIOGRAPHIES OF AUTHORS

Reza Maulana is a student who will complete a master's degree in the Computer Science program at Sriwijaya University. The fields of concentration he is studying are networking and cyber security. He also works at a private company in Palembang, Indonesia, focusing on marketing managerial and services in the IT and networking fields. He has also joined in network research and filling workshops as a trainee, he has the competence and has been certified as a MikroTik Certified Network Associate (MTCNA) and MikroTik Certified Routing Engineer (MTCRE). He can be contacted at email: reza.javas@gmail.com.

Deris Stiawan is a Professor in the Faculty of Computer Science University of Sriwijaya, Indonesia. He is a member of IEEE and since 2010 he has joined on Pervasive Computing Research Group (PCRG) Universiti Teknologi Malaysia. His professional profile has derived to computer and network security fields, focused on network attack and intrusion prevention/detection systems. In 2011, he holds Certified Ethical Hacker (C|EH) & Certified Hacker Forensic Investigator (C|HFI) licensed from EC-Council USA and Cisco Certified Networking Associate in 2005. He can be contacted at email: deris@unsri.ac.id.

Rahmat Budiarto received B.Sc. degree from Bandung Institute of Technology in 1986, M.Eng, and Dr. Eng in Computer Science from Nagoya Institute of Technology in 1995 and 1998 respectively. Currently, he is a full professor at College of Computing and and Information, Albaha University, Saudi Arabia. He was the chairman of Network Security Working Group at Asia Pacific Advanced Networks (APAN) from January 2006 to August 2008, and the chairman of Fellowship Committee at Asia Pacific Advanced Networks (APAN) from January 2007 to August 2008. He was the deputy director and co-founder of National Advanced IPv6 (NAv6) Center, under the Ministry of Energy, Water, and Communication, Malaysia. His research interests include IPv6, network security, wireless sensor networks and intelligent systems. He can be contacted at email: rahmat@bu.edu.sa.