Penerbit Akademia Baru

Journal of Advanced Research Design

Journal homepage:
https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Malware Detection in Windows Using Deep Learning Classification Approach

Mohd Faris Mohd Fuzi[1,*], Aishah Anuar[1], Mohammad Hafiz Ismail[1], Mohamad Yusof Darus[2], Tajul Rosli Razak[2], Nurul Huda Nik Zulkipli[3], Evizal Abdul Kadir[4]

[1] College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Perlis Branch, Arau Campus, 02600 Arau, Perlis, Malaysia
[2] College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia
[3] College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Melaka Branch, Jasin Campus 77300 Merlimau, Melaka, Malaysia
[4] Department of Informatics Engineering, Universitas Islam Riau, Pekanbaru 28284, Indonesia

ARTICLE INFO

Article history:
Received 23 March 2025
Received in revised form 22 April 2025
Accepted 23 September 2025
Available online 5 November 2025

Keywords:
Windows Malware; Malware Detection; Deep Learning; Classification

ABSTRACT

Microsoft Windows is the most common operating system, and because of its global popularity, it is also the most popular platform for hackers to target. It is also susceptible to security flaws. According to the Common Vulnerabilities and Exposures (CVE) database, which tracks known system vulnerabilities, Microsoft had over 660 dangerous security holes, 357 of which were associated with Windows 10. Thus, users may also be at risk because of security flaws in the Windows applications they employ or because of attacks on connected devices. Windows malware has been a major threat to computer software for decades, putting millions of people in danger. An attacker creates it to disrupt computer operations, gather sensitive information, or gain access to private computer systems. The increasing number of zero-day vulnerabilities and the rapid growth of Windows malware require efficient and accurate malware detection. Thus, this paper discusses Windows malware detection using a deep learning classification approach. In this study, the samples of Windows malware were analysed using malware analysis tools such as HashMyFiles and CFF Explorer. Subsequently, the malware visualisation was used to convert the binaries of malware files to generate a grayscale dataset. The classification process implemented using CNN and RNN for malware detection was being evaluated. Using the Metric Formula Definition Accuracy, the performance of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) malware detection models in Windows has been tested. According to the models, CNN is doing better, providing an accuracy of 97.5 percent in detecting malware, whereas RNN provides an accuracy of 88.5 percent, respectively. This study evaluated the accuracy performance between the CNN and RNN architecture models.

1. Introduction

Malware has been a major threat to computer software for decades, putting millions of people in danger. The number and variety of Windows malware have rapidly increased in recent years.

* *Corresponding author.*
E-mail address: farisfuzi@uitm.edu.my

Malware possesses the ability to infiltrate the system and can perform a variety of malicious actions, including stealing information, altering file information, and disguising itself as a legitimate file. The malware types include ransomware, adware, spyware, rootkits, keyloggers, worms, and viruses [1]. According to the AV-TEST Institute, the number of new malware and potentially unwanted application (PUA) registrations exceeds 450,000 per day. Based on the SonicWall report in the first half of 2021, there were 304.7 million ransomware attacks worldwide, a 151% increase since 2020. In 2021, ransomware attacks affected 80% of organisations [2]. The organisation is concerned about the ransomware attack because it uses encryption to hold a victim's data hostage. As a result, the organisation's data remains encrypted, preventing them from accessing files, databases, or applications. Users may also be at risk due to security flaws in their applications or attacks on connected devices. In IoT (Internet of Things) environments, these threats can lead to severe consequences, ranging from the compromise of personal data to the disruption of essential services [3].

Thus, malware analysis is required to discover the malware's nature and purpose, as well as the attacker's aim and motive. Malware analysis is the investigation into the behavior and purpose of a suspicious file or URL. The analysis results assist in detecting and mitigating potential threats. In addition, it's crucial to determine the method of system hacking and the extent of its impact. The existing malware analysis techniques use static and dynamic analysis. Static analysis is the process of analysing a binary file without running it. Meanwhile, dynamic analysis is the process of running a suspicious binary in a controlled environment and monitoring how it behaves [4][5][6].

The increasing number of zero-day vulnerabilities and the rapid growth in malware quantity require efficient and accurate malware detection. Each year, millions of new malware varieties are detected by anti-malware vendors. Thus, the researchers must quickly develop new intelligent malware detection systems to halt this trend. Many detection methods have been proposed in the past few decades [7]. Malware may be detected in a variety of ways using a variety of techniques. However, using traditional malware detection techniques is limited by the number of detection rules that need to be set manually [8]. In addition, malware detection projects now involve artificial intelligence using machine learning. Machine learning (ML) is a subfield of artificial intelligence (AI) and computer science that focuses on using data and algorithms to mimic how humans learn, gradually improving its accuracy [4][9][10][11][12]. Machine learning may be used to detect the existence of malware and work by categorising a set of data into several categories. Classification is used to distinguish between two types of entities, which are benign and malicious. Many ML algorithms have been used, like support vector machines (SVM) [13, 14,15], K-nearest neighbor (KNN) [16, 17], Bayesian estimation [18, 19], genetic algorithms [20], etc., to build malware detection systems. As a result, machine learning is suitable for malware detection because it has the highest detection accuracy. However, machine learning is a traditional way to detect malware.

Nowadays, deep learning can accurately and mostly perfectly detect malware since it rapidly increases. Deep learning has gained popularity in recent years because of its higher accuracy when compared to regular machine learning. Every day, cyber criminals devise new schemes to breach networks and steal sensitive data. As a result, finding new ways to secure information has become a challenge. Many strategies have been used to identify malware threats using deep learning methods [21][22][23][24]. The most important aspect to emphasise is the accuracy of malware detection. This is because the accuracy of detection validates the effectiveness of this strategy. If the accuracy is low, the malware is difficult to detect. The issue of malware detection remains unresolved. Thus, the purpose of this study is to apply deep learning techniques using the CNN and RNN architectures to investigate the accuracy of malware detection.

2. Related Works

In one of the previous studies, Hossain et al. [25] proposed a malware detection approach using three different neural network models, which are CNN, Long-Short Term Memory Network (LSTM) and Gated Recurrent Unit (GRU). This study compared the performance of various models in detecting malware and benign applications. The goal of this study is to improve malware detection methods and protect users from security threats as the number of malicious applications grows at a rapid rate. After comparing this neural network, they realised CNN was the best solution because it performed better in image classification tasks to detect malicious or benign files.

Meanwhile, Agrawal et al. [26] proposed detecting ransomware by combining the Attended Recent Inputs (ARI) cell with Long Short-Term Memory (LSTM) networks, which they called ARI-LSTM. They enhance the LSTM cell by incorporating an ARI mechanism and utilise the resulting neural network for ransomware sequence learning. They test ARI-LSTM on a ransomware dataset for the Windows operating system to show that it improves the performance of an LSTM in finding ransomware from emulation sequences.

Another researcher, Choi et al. [27], proposed a deep learning model for malware detection using malware images. In this study, they use CNN for malware image recognition. Firstly, they construct images from both benign files and malware, as each type of malware has a corresponding image. Additionally, they can obtain the picture faster than API sequences. Second, they identify malware by applying a deep learning model based on CNN, since the CNN model learns characteristics from photos. In this project, they achieved the highest detection accuracy.

The following study is from S.L. and C.D. [28], whose proposed CNN-based Windows malware detector consists primarily of two phases, which are training and prediction. During the training phase, the malware detector is trained to recognise and categorise unknown PE files so that it can later detect malicious software. The behaviour-based feature extractor, the Feature Selector, the Final Features Set, the Image Generator, and the CNN are the essential modules that are utilised during the training phase. This paper proposed, implemented, and evaluated a CNN-based Windows malware detector using a dataset of 7,433 MIST files. The best 676 N-grams suggested by each Feature Selection Technique, including Chi-Square, Information Gain, Mutual Information, and Relief, were considered, and 7,433 N-gram files were used to generate 7,433 images, which were used to evaluate the performance of the proposed CNN-based malware classification method. The paper measured the effectiveness of the proposed method and compared it to six machine learning-based classifiers to determine which one performed the best. According to the obtained empirical results, the performance of the proposed method was superior to that of the six selected machine learning-based classifiers. Additionally, using 50 epochs, the proposed method achieved a malware detection accuracy of 97.968 percent for the N-grams recommended by the Relief Feature Selection Technique.

Finally, in a study by Jha et al. [29], they proposed an efficient recurrent neural network (RNN) to detect malware. Artificial neural networks subcategorize RNNs, which interconnect to form a directed graph and a temporal sequence. In this paper, they present the results of several experiments they ran using various hyperparameter values. Their testing revealed that when using RNN for malware classification, the step size matters more than the input size. They evaluated RNN's performance with three different feature vectors, using hyperparameters to support RNN's proof-of-concept as an effective method for malware detection. The three feature vectors are "Hot Encoding feature vector," "random feature vector," and "Word2Vec feature vector." They also ran a pairwise t-test to see if the results were statistically significant with respect to one another. Their findings demonstrate that among the three feature vectors, the RNN with the Word2Vec feature vector had the highest Area Under the Curve (AUC) value and acceptable variance. According to their empirical

analysis, the researchers found that the Word2Vec model's skip-gram architecture feature vectors in RNNs are the best and most stable at finding malware.

3. Methodology

There are five phases involved in this research: malware and benign software collection, malware analysis, malware visualisation, model training and validation, and model performance evaluation. Figure 1 shows the Windows malware detection model flowchart.

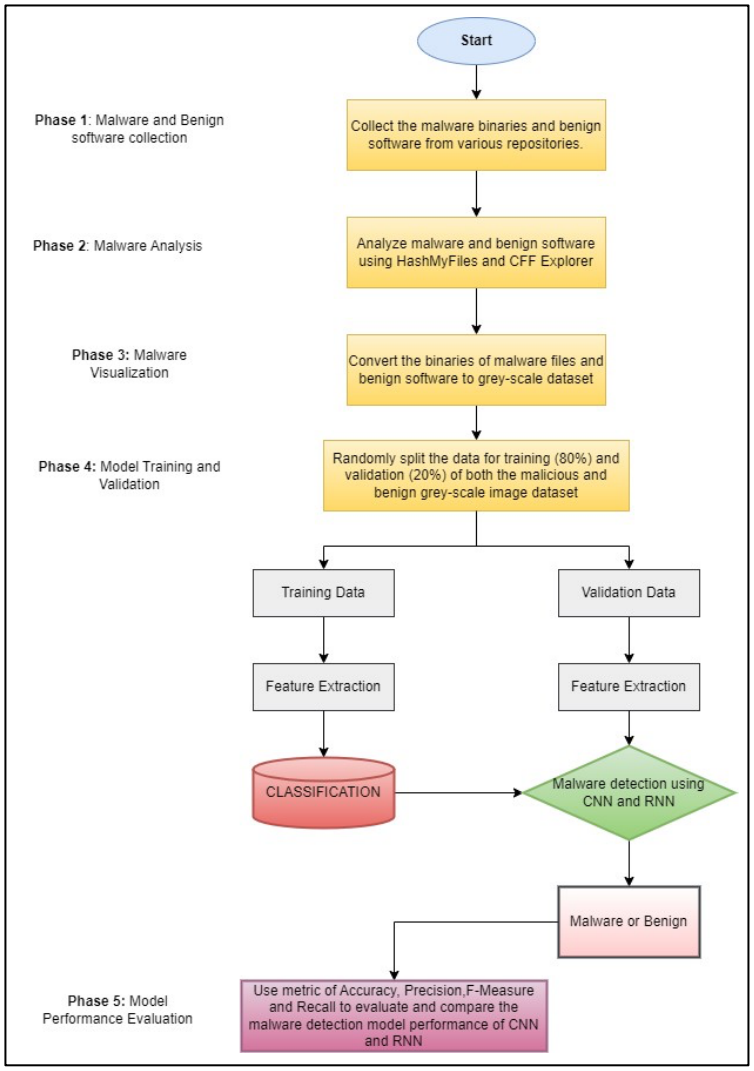

Fig. 1. Malware Detection Model Flowchart

3.1 Malware and Benign Software Collection

This dataset contains a total of 1000 samples: 600 distinct samples of malicious software and 400 distinct samples of benign software. The benign software dataset was obtained from the website en.softnic.com. This collection includes a total of 400 different types of software. This software is entirely safe to run on the PC. The malware collection includes a total of 600 different types of malware families. Unwanted software, known as malware, infiltrates computers without authorization with the intention of causing harm and disrupting computer networks and systems. Once it has been exploited, it will cause harm to our computer.

3.2 Malware Analysis

This study employed malware analysis, a method that uses static analysis to identify and analyse suspicious files on endpoints and within networks. To understand the software's behaviour, the method consists of extracting features. The benign software and malware have been analysed using HashMyFiles and CFF Explorer to extract the feature selection and generate the dataset in a Comma-Separated Value (CSV) file.

3.3 Malware Visualization

In this phase, the process involved malware visualization. These malware images were generated using binaries. The benign and malicious files have been disassembled. Each of the files will extract the md5 and sha256 that contain hexadecimal values. After that, the hexadecimal values will be converted to binary. It generated the grayscale image from the binary. Figure 2 shows the process of malware visualisation.

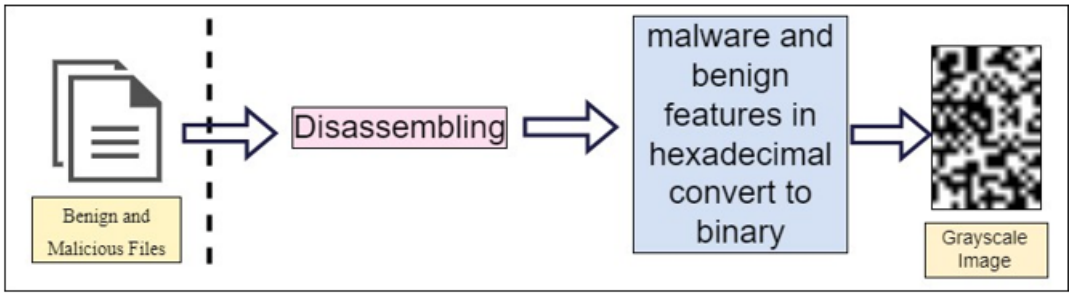

Fig. 2. Malware visualisation into a grayscale image

3.4 Model Training and Validation

This phase constructed a deep learning classification model utilising the CNN architecture in order to train and test the model on a dataset consisting of malware and benign grayscale images. The dataset of grayscale images containing both harmful and benign images will be randomly divided for training (80%) and validation (20%) purposes. The effectiveness of the malware detection model that was created as a result will be assessed. This procedure has been written in the Python programming language and built with Anaconda. Figure 3 shows the CNN training script that is used to train the benign and malware images. It trained based on the epochs that have been set to learn both images.

```
history=model.fit(
    x=training_aug,
    epochs=EPOCHS,
    validation_data=(testX,testY),
    batch_size=training_aug.batch_size,
    steps_per_epoch = training_aug.n // training_aug.batch_size,
    callbacks = cb
)



model.save_weights(MODEL_SAVE_WEIGHT)
model.save(MODEL_SAVE)
```

Fig. 3. CNN training script

Figure 4 represents the calculation of the CNN accuracy. It calculates the accuracy, precision, recall, F-measure, Cohen Kappa score, and confusion matrix. Finally, it displayed a classification report with training data details.

```
accuracy_sc = accuracy_score(valid_labels,ypred)
precision =precision_score(valid_labels,ypred,average='weighted')
recall =recall_score(valid_labels,ypred,average='weighted')
f1 =f1_score(valid_labels,ypred,average='weighted')
kappa = cohen_kappa_score(valid_labels,ypred)
matrix = confusion_matrix(valid_labels,ypred)

classreport = classification_report(valid_labels,ypred
                                    ,target_names=imagenet_labels)

print("Accuracy : {:.4f}, Precision : {:.4f}, Recall :
      {:.4f}".format(accuracy_sc,precision,recall))
print("F1: {:.4f}, Cohen-Kappa : {:4f}".format(f1, kappa))

print("\n\n=== Confusion matrix === ")
print(matrix)

print("\n\n=== Classification Report ===\n")
print(classreport)
```

Fig. 4. CNN Accuracy Calculation

Figure 5 represents the RNN model scripting, which will have several layers. The input connection within the LSTM nodes is eligible for dropout, which can be applied as necessary. The data on the input connection to each LSTM block is excluded from node activation and weight updates if there is a dropout on the input. This means that for a given probability, the data will be dropped.

```
# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 10,activation='tanh', return_sequences = True,
                   input_shape = (X_train.shape[1], 1)))

# Adding a second RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fifth RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))
```

Fig. 5. RNN model scripting

Figure 6 shows how the RNN architecture has been combined using the Adam optimizer. It is an alternative to stochastic gradient descent as an optimisation technique for training deep learning models. This RNN architecture used regressor.fit to fit the training set.

```
# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error' ,metrics=['accuracy'])


# Fitting the RNN to the Training set
t = regressor.fit(X_train, y_train, epochs = 8, batch_size = 32, validation_data=(X_test,y_test))
```

Fig. 6. RNN training script

3.5 Model Performance Evaluation

The goal of any model evaluation should be to establish whether or not the performance of a model meets the objectives it was designed to achieve. However, when it comes to tasks involving the detection or classification of malware, class-unbalanced classification problems occur naturally. This means that the number of benign software samples is significantly greater than that of harmful software, and the quantity of malware samples within large families is far greater than those within rare families. As a consequence of this, the effectiveness of the malware detection or classification model needs to be evaluated using the right metrics in order to provide a complete picture. The following metrics of evaluation are being used in these studies, True Positive (TP), True Negative (TN), False Positive (FP), False Negative, Accuracy, Precision, Recall and F-Measure.

4. Result and Discussion

4.1 Result and analysis for CNN

Figure 7 shows the confusion matrix graph that contains true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The true positive for this validation set is 100%. It means this validation set correctly detects or classifies the grayscale image as malware. The true negative is 94%, which means the validation set is correctly detected as benign and not classified as malware. The false positive rate is less than 10%, which is good since the validation set is less likely to be incorrectly detected as malware or incorrectly classified as malware, while the grayscale image sample is benign. The false negative is 0%, which means the validation set does not incorrectly detect the malware grayscale image as a benign sample.

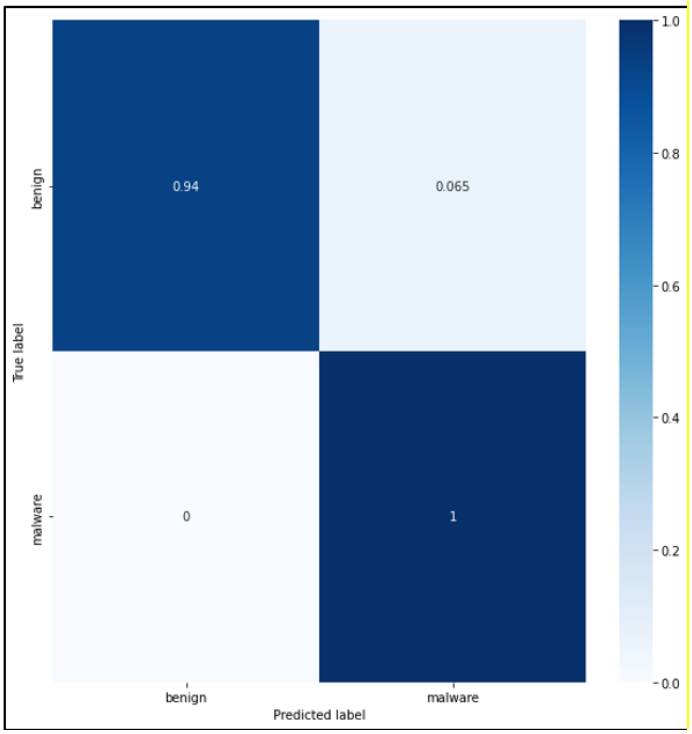

Fig. 7. Confusion matrix graph for CNN architecture.

Figure 8 shows the accuracy value for every grayscale image in the validation set. The validation of the dataset was successful, as indicated by the green colour label that is located above the image. On the other hand, the red colour indicates that the dataset was categorised incorrectly.

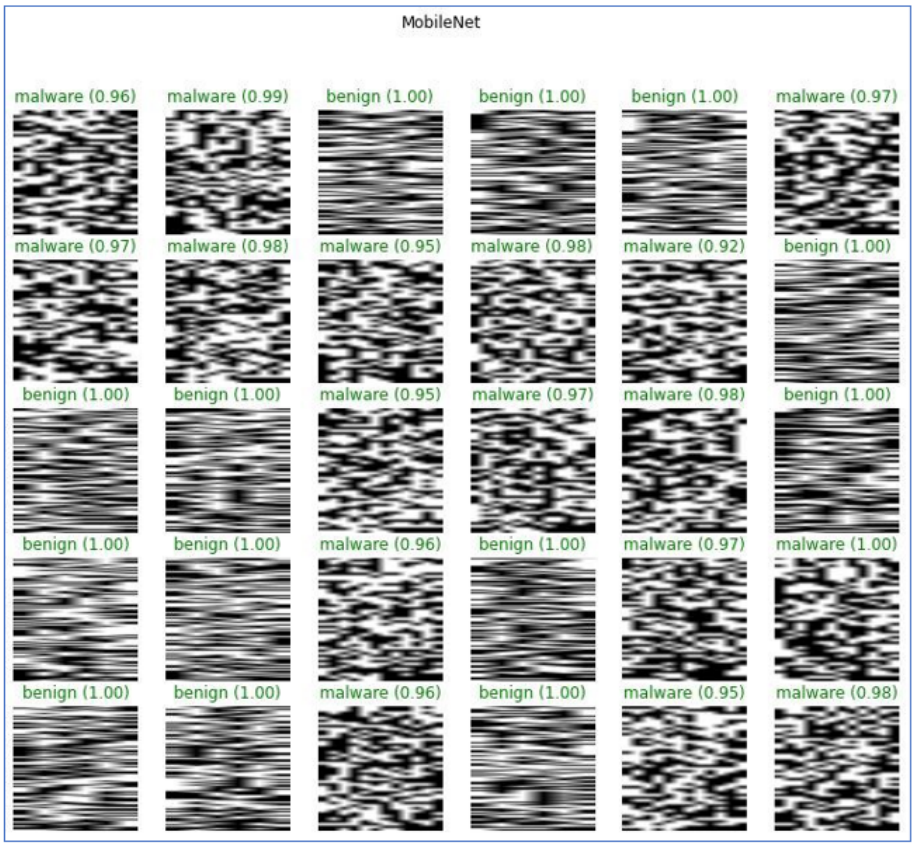

Fig. 8. Accuracy value for each grayscale image

4.2 Result and analysis for RNN

Figure 9 shows the confusion matrix graph that contains true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The true positive for this validation set is 62%. It means this validation set correctly detects or classifies the features as malware. The true negative is 26.5%, which means the validation set is correctly detected as benign and not classified as malware. The false positive rate is less than 12%, which is good since the validation set is less likely to be incorrectly detected as malware or incorrectly classified as malware, while the feature sample is benign. The false negative is 0%, which means the validation set does not incorrectly detect the malware features as a benign sample.

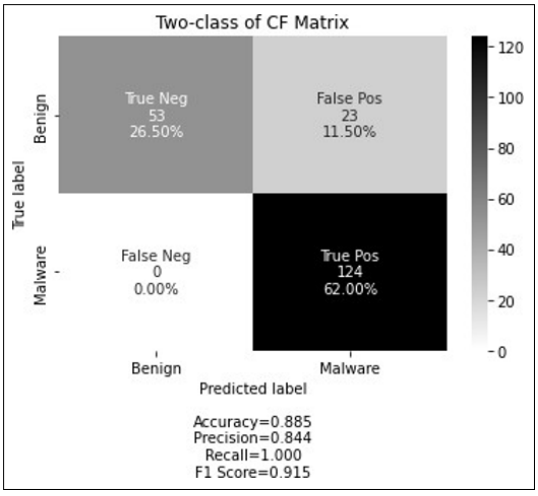

Fig. 9. Confusion Matrix for RNN architecture

4.3 Comparison between CNN and RNN

According to Table 1, CNN has a 97.5 percent accuracy, whereas RNN has an 88.5 percent accuracy. Since then, CNN has improved its ability to determine whether a file is malicious or benign. This is because CNN layers contain many convolutional filters that evaluate the complete matrix of features and minimise spatial size. This makes CNN a very convenient and suitable network for categorising malware and benign data. RNN are less accurate than CNN due to memory-bandwidth-restricted computations that minimise the utilisation of neural network implementation.

Table 1
Comparison between CNN and RNN architecture

Model (%)	CNN	RNN
Accuracy	97.5	88.5
Precision	97.6	84.4
Recall	97.5	100
F1-Score	97.5	91.5

5. Conclusions

In this study, the purpose is to construct a deep learning classification model based on the extracted features from sampled malware and benign software. The feature is being converted from hexadecimal to binary. After that, it will be converted to a grayscale image using Python code. The second purpose of this study is to evaluate the accuracy of the resulting malware detection model. This study used training and validation to achieve accuracy. For training purposes, use 80% of the combined grayscale image from benign and malicious software. For the validation, we will use 20% of the grayscale image dataset. The accuracy of the CNN architecture approach is 97.5%. Meanwhile, the accuracy of the RNN architecture is 88.5%. This shows CNN is more accurate in determining whether the image is malicious or benign.

For future works, several suggestions can be made to enhance its quality and accuracy. To improve this project, we encourage the gathering of additional feature selections. This is because the system has the capability to generate large grayscale images. Aside from that, this study needs to make use of a larger dataset to get a higher level of accuracy and implement using another various technique of deep learning.

Acknowledgement
The author hereby acknowledges the financial support from University Technology MARA (UiTM) under 600-RMC 5/3/GPM (058/2022).

References
[1] Taib, A.M. & Azman, N.N.K. (2023). Experimental Analysis of Trojan Horse and Worm Attacks in Windows Environment. Journal of Advanced Research in Computing and Applications, 13(1), 1–9. Retrieved from https://www.akademiabaru.com/submit/index.php/arca/article/view/5002.
[2] Segal, E. (2022, February 9). A majority of surveyed companies were hit by ransomware attacks in 2021-and paid ransom demands. Forbes. Retrieved April 20, 2022, from https://www.forbes.com/sites/edwardsegal/2022/02/03/a-majority-of-surveyed-companies-were-hit-by-ransomware-attack-in-2021-and- paid-ransom-demands/?sh=7f2a9598b8c6

[3] Mahmoud Ahmed, M. A., Idris, S. A., Md Yunus, N. A., Mohd Ali, F., & Sofian, H. (2024). CyberShield Framework: Attacks and Defense Modelling for Cybersecurity in Internet of Things (IoT). International Journal of Computational Thinking and Data Science, 4(1), 30–39. https://doi.org/10.37934/ctds.4.1.3039
[4] Dilhara, S. (2021). Classification of Malware using Machine Learning and Deep learning Techniques. International Journal of Computer Applications. 183. 12-17. https://doi.org/10.5120/ijca2021921708
[5] Yu, B. & Fang, Y., Yang, Qiang., Tang, Y. & Liu, L. (2018). A Survey of Malware Behavior Description and Analysis. Frontiers of Information Technology & Electronic Engineering. 19. 583-603. https://doi.org/10.1631/FITEE.1601745
[6] Chakkaravarthy, S., Sangeetha, D. & Vaidehi, V. (2019). A Survey on Malware Analysis and Mitigation Techniques. Computer Science Review. 32. 1-23. https://doi.org/10.1016/j.cosrev.2019.01.002
[7] Tayyab, U., Khan, F. B., Durad, M. H., Khan, A. & Lee, Y. S. (2022). A Survey of the Recent Trends in Deep Learning Based Malware Detection. Journal of Cybersecurity and Privacy. 2. 800-829. https://doi.org/10.3390/jcp2040041.
[8] Xing, X., Jin, X., Elahi, H., Jiang, H., & Wang, G. (2022). A malware detection approach using Autoencoder in deep learning. IEEE Access, 10, 25696–25706. https://doi.org/10.1109/access.2022.3155695
[9] Brown, S. (2021, April 21). Machine Learning, explained. MIT Sloan. Retrieved June 10, 2022, from https://mitsloan.mit.edu/ideas-made-to-matter/machine-learningexplained
[10] Mohd Fuzi, M.F., Mohd Shahirudin, S., Abd Halim, I.H. and Jamaluddin, M.N.F. (2023). Comparison of Supervised Machine Learning Algorithms for Malware Detection . Journal of Computing Research and Innovation. 8, 2 (Sep. 2023), 67–73. DOI: https://doi.org/10.24191/jcrinn.v8i2.329
[11] Poornima, S. & Mahalakshmi, R. (2024). Automated Malware Detection using Machine Learning and Deep Learning Approaches for Android Applications. Measurement: Sensors, Volume 32, 100955, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2023.100955.
[12] Aburashed, L., AL Amoush, M., & Alrefai, W. (2024). SQL Injection Attack Detection using Machine Learning Algorithms . Semarak International Journal of Machine Learning , 2(1), 1–12. https://doi.org/10.37934/sijml.2.1.112
[13] Vaidya, A., Pande, M., Shankrod, S., Dorkar, T. & Aundhakar, S. (2023). International Research Journal of Modernization in Engineering Technology and Science. Vol 05. Issue 03. DOI : https://www.doi.org/10.56726/IRJMETS34910.
[14] Singh, P., Borgohain, S. K. & Kumar, J. (2022). Performance Enhancement of SVM-based ML Malware Detection Model Using Data Preprocessing. 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET). pp. 1-4, DOI: https://doi.org/10.1109/ICEFEET51821.2022.9848192.
[15] Wadkar, M., Troia,F.D. & Stamp, M. (2020). Detecting Malware Evolution using Support Vector Machine. Expert Systems with Applications. Volume 143. 113022. ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2019.113022.
[16] Supriyanto, C., Rafrastara, F. A., Amiral, A., Amalia, S. R., Fahreza, M. D. A. & Abdollah, M. F. (2024). Malware Detection Using K-Nearest Neighbor Algorithm and Feature Selection. JURNAL MEDIA INFORMATIKA BUDIDARMA. Volume 8. Number 1. Page 412-420. DOI: https://doi.org/10.30865/mib.v8i1.6970
[17] Assegie, T.A. (2021). An Optimized KNN Model for Signature-Based Malware Detection. International Journal of Computer Engineering in Research Trends (IJCERT). Vol.8, Issue 02, pp. 46-49, https://ijcert.org/ems/ijcert_papers/V8I206.pdf.
[18] Castillo-Zúñiga, I.; Luna-Rosas, F.; Rodríguez-Martínez, L.; Muñoz-Arteaga, J.; López-Veyna, J.; Rodríguez- Díaz, M.A. (2020). Internet Data Analysis Methodology for Cyberterrorism Vocabulary Detection, Combining Techniques of Big Data Analytics, NLP and Semantic Web. International Journal on Semantic Web and Information Systems (IJSWIS). 16(1). 69–86. DOI: https://doi.org/10.4018/IJSWIS.2020010104
[19] Ramadhan, B., Purwanto, Y. & Ruriawan, M. F. (2020). Forensic Malware Identification Using Naive Bayes Method. International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia. pp. 1-7, DOI: https://doi.org/10.1109/ICITSI50517.2020.9264959.
[20] Yildiz, O. & Dogru, I.A. (2019). Permission-based Android Malware Detection System using Feature Selection with Genetic Algorithm. International Journal of Software Engineering and Knowledge Engineering. Vol. 29. No. 02. pp. 245-262. https://doi.org/10.1142/S0218194019500116.
[21] Bensaoud, A., Kalita, J. & Bensaoud, M. (2024). A Survey of Malware Detection using Deep Learning. Machine Learning with Applications. Volume 16. 100546. ISSN 2666-8270. https://doi.org/10.1016/j.mlwa.2024.100546.
[22] Alomari, E.S., Nuiaa, R.R., Alyasseri, Z.A.A., Mohammed, H.J., Sani, N.S., Esa, M.I. & Musawi, B.A. (2023). Malware Detection Using Deep Learning and Correlation-Based Feature Selection. Symmetry 2023. 15. 123. https://doi.org/10.3390/sym15010123
[23] Shaukat, K., Luo, S. & Varadharajan, V. (2023). A Novel Deep Learning-based Approach for Malware Metection.Engineering Applications of Artificial Intelligence. Volume 122. 106030. ISSN 0952-1976. https://doi.org/10.1016/j.engappai.2023.106030.

[24] He, K. & Kim, D. S. (2019). Malware Detection with Malware Images using Deep Learning Techniques. 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 95-102. DOI: https://doi.org/10.1109/TrustCom/BigDataSE.2019.00022

[25] Hossain, H., Kayum, S. I., Paul, A., Rohan, A. A., Tasnim, N., & Hossain, M. I. (2021). Malware detection using Neural Networks. 2021 5th International Conference on Electrical Information and Communication Technology (EICT). https://doi.org/10.1109/eict54103.2021.9733457

[26] Agrawal, R., Stokes, J. W., Selvaraj, K., & Marinescu, M. (2019). Attention in recurrent neural networks for ransomware detection. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/icassp.2019.8682899

[27] Choi, S., Jang, S., Kim, Y., & Kim, J. (2017). Malware detection using malware image and Deep Learning. 2017 International Conference on Information and Communication Technology Convergence (ICTC). https://doi.org/10.1109/ictc.2017.8190895

[28] S.L, S. D., & C.D, J. (2021). Windows malware detector using convolutional neural network based on visualization images. IEEE Transactions on Emerging Topics in Computing, 9(2), 1057–1069. https://doi.org/10.1109/tetc.2019.2910086.

[29] Jha, S., Prashar, D., Long, H. V., & Taniar, D. (2020). Recurrent neural network for detecting malware. Computers & Security, 99, 102037. https://doi.org/10.1016/j.cose.2020.102037