

Investigating the Impact of Training and Testing Ratios on the Performance of an AI-Based Malware Detector using MATLAB

Carlo, Romero, PECE, M.ENG

Quezon City University

Quezon City, Philippines

Abstract.

This research investigates the impact of the training and testing ratios on the performance of an AI-Based Malware Detector using MATLAB. The experiments through MATLAB have shown that higher training percentage means that a larger portion of dataset for training the model have been used while a lower training percentage shows that a large portion of the dataset reserved for testing the model's performance. The exploration of the influence of training and testing ratios also have been able to determine the performance of an AI-Based Malware Detector. The results give to determining the relationship between training and testing ratios and the effectiveness of the malware detection system.

Keywords: Malware Detector; Artificial Intelligence; MATLAB Based System; AI-Based System

Introduction

The era of artificial intelligence opens the possibility of doing things in the blink of an eye. Although we are just starting to embrace this technology, sooner or later, all of the manual stuff will be a thing of the past. With so many applications using AI, the possibility of exploitation of IP addresses and other credentials will be much easier than before. Traditional signature-based detectors and behavior-based methods will be less effective in the coming days. As a result, there is a need for new approaches to malware detection that can identify AI-powered malware and protect computer systems from its harmful effects. This malware driven by AI will be more powerful, so security for end devices and networks must prioritize. Cybersecurity threats are also becoming increasingly common and sophisticated. Malware, in particular, has become a pervasive and damaging form of cyber attack that can compromise the security of computer systems and data. Traditional signature-based malware detection approaches are no longer effective against advanced and evolving malware. Therefore, developing new and more robust malware detection methods is of critical importance. These methods depend on the training and testing of datasets to evaluate the performance. The primary concern is the ratio of the training and testing on the performance of an AI-Based Malware Detector. To resolve this dilemma, an investigation of the impact of the testing and training ratio of the performance of an AI-powered malware detector is proposed. The objective of this study is to evaluate the effectiveness of our malware detector in detecting malware through determining the impact of the testing and training of datasets in a real-world setting. Specifically, we aim to compare the performance of our detector with existing malware detection methods and demonstrate its effectiveness in detecting unknown and new malware variants. The researcher utilizes an AI-powered malware detector using MATLAB, a popular software tool for scientific computing and machine learning. The researcher's goal is to evaluate the machine learning models that can accurately detect AI-powered malware by analyzing various features of malicious code. By leveraging the power of MATLAB, the researcher aims to create a robust and effective malware detection system that can adapt to new and emerging threats. In obtaining an AI-powered malware detection system, dataset collection from various sources is the first thing to do. The researcher will then preprocess and analyze the dataset to extract relevant features that could train and evaluate machine learning models. The researcher will explore various machine learning algorithms, including deep learning, and evaluates their performance in detecting AI-powered malware. The proposed AI-powered malware detector has several potential applications, including improving the security of computer networks. This proposal will also enhance the effectiveness of antivirus software and protect individuals and organizations from cyber threats. The results of this study can also provide insights into the capabilities of AI-powered malware and inform the development of new strategies for cybersecurity.

Methodology

The MATLAB Simulation is used in investigating the impact of the testing and training ratio of the performance of an AI-Based malware detection techniques.

A. MATLAB Code for Malware Detection

The effectiveness of the algorithm using MATLAB will greatly depend on the dataset quality. Based on the codes below, malware detection system uses more advanced and sophisticated techniques.

```
% Malware Detection
% Step 1: Load the dataset
load('malware_dataset.mat'); % Assuming you have a preprocessed dataset

% Step 2: Split the dataset into training and testing sets
splitRatio = 0.8; % 80% for training, 20% for testing
splitIndex = round(splitRatio * size(dataset, 1));
trainingData = dataset(1:splitIndex, :);
testingData = dataset(splitIndex+1:end, :);
% Step 3: Train the classifier
classifier = fitensemble(trainingData(:, 1:end-1), trainingData(:, end),
'AdaBoostM1', 100, 'Tree'); % Example classifier, you can choose a different one
% Step 4: Test the classifier
predictions = predict(classifier, testingData(:, 1:end-1));
% Step 5: Evaluate the performance
actualLabels = testingData(:, end);
accuracy = sum(predictions == actualLabels) / numel(actualLabels);
fprintf('Accuracy: %.2f%%\n', accuracy * 100);
% Step 6: Display the confusion matrix
confusionMat = confusionmat(actualLabels, predictions);
disp('Confusion Matrix:');
disp(confusionMat);
```

The code also assumes that a preprocessed dataset named `malware_dataset.mat` contained feature vectors and corresponding labels (0 for clean files and 1 for malware files). The code splits the testing and training of dataset using an 80:20 ratio. It then trains an ensemble classifier on the testing data. Accuracy percentage and Confusion Matrix can also be computed with the actual labels.

B. Dataset for Testing purposes using MATLAB

A dataset has been collected, created and loaded in MATLAB for Testing purposes.

```
% Example Malware Dataset - MATLAB Code
% Generate synthetic data
numSamples = 1000; % Number of samples in the dataset
numFeatures = 10; % Number of features for each sample
% Create feature matrix
features = rand(numSamples, numFeatures);
% Create label vector
labels = randi([0, 1], numSamples, 1); % Random labels (0 or 1) %
Combine features and labels into a dataset matrix
dataset = [features, labels];
% Save the dataset as a MATLAB .mat file
save('malware_dataset.mat', 'dataset');
```

The example code consists of 1000 samples, where each sample has 10 randomly generated features. The labels are randomly assigned as either 0 or 1. The feature matrix has dimensions $\text{numSamples} \times \text{numFeatures}$, and the label vector labels has dimensions $\text{numSamples} \times 1$. The dataset matrix `dataset` combines the features and labels into a single matrix with dimensions $\text{numSamples} \times (\text{numFeatures} + 1)$.

C. Research Design

The accuracy and performance of the Artificial Intelligence Model is also presented in this paper. The researcher used a framework that can train and test datasets for investigating its impact in determining the performance of an AI Based Malware detector.

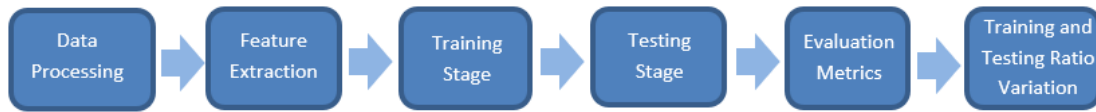


Fig. 1. Block Diagram of AI-Based Malware Detector

Figure 1 shows the Block Diagram of the AI Based Malware Detector. Initially, dataset must be prepared for analysis, which may include task such as feature selection or handling missing values. Then it compasses the techniques used to extract features from the malware samples, then at the training phase, a machine learning algorithm is used using training dataset. Model selection and training iterations were used in training dataset. Then, after the training stage, it will undergo under testing stage, where the trained model is evaluated using the testing dataset. The performance evaluation of the AI-Based Malware Detector can be done using the Evaluation Metrics. At the final stage, this section relates to the investigation of different training and testing ratios and its impact on the malware detector performance. The researcher will also train Artificial Intelligence Model to develop an effective malware detection model that can accurately distinguish between clean and malicious files. Convolutional Neural Network (CNN) is selected by the proponent as the artificial intelligence model. Since CNNs have been widely used in image classification, which can also be applicable in malware detection. The steps in using CNN for malware detection are shown in the figure 2.



Fig. 2. Using CNN for Malware Detection

Initially, it allows for preparing dataset of clean and with malware samples. It should be properly determined that must be classified clean or malware. Then dataset should be preprocess, making it suitable in a CNN training format. This can be applied to malwares particularly by resizing its images to a consistent size, normalizing pixel values

and by dataset augmentation. Rotation and cropping are effective in increasing its variability. Dataset should be divided into training and validation. The CNN architecture involved in this design is stacking convolutional layers, pooling layers and fully connected layers. During training, the model adjusts its internal weights using optimization techniques such as stochastic gradient descent (SGD) or Adam optimization to minimize the loss function. The model iteratively processes the training samples, updates the weights, and learns to classify clean and malware samples correctly. Also, fine tuning the hyperparameters of the CNN such as the learning rate, batch size and number of layers, helps the optimization of the performance model. Then the researcher evaluates the trained CNN using the testing set. Performance metrics such as accuracy, precision, recall, or F1 score are used to assess the model's effectiveness in malware detection. Once the client is satisfied with the performance of the CNN, this can be now deployed to classify new and unseen samples as clean or malware.

It is worth noting that CNNs might require a significant amount of training data and computational resources.

Overall, CNNs have shown promising results in various domains, including image-based tasks, and can be a powerful choice for AI-based malware detection. Based on the MATLAB Codes to train a CNN for malware detection:

```
% Load and preprocess the dataset (assuming you have preprocessed the
data)
% X_train: Training data (images)
% y_train: Training labels (0 for clean, 1 for malware)
% X_test: Testing data
% y_test: Testing labels
% Define the CNN architecture
layers = [ imageInputLayer([32 32 3]) % Input layer, assuming images are
RGB and of size 32x32
convolution2dLayer(3, 32, 'Padding', 'same') % Convolutional layer with 32
filters, filter size 3x3
reluLayer() % ReLU activation
maxPooling2dLayer(2, 'Stride', 2) % Max pooling layer with 2x2 pooling size and
stride 2
convolution2dLayer(3, 64, 'Padding', 'same') % Another convolutional layer
with 64 filters
reluLayer()
maxPooling2dLayer(2, 'Stride', 2)
fullyConnectedLayer(64) % Fully connected layer with 64 neurons
reluLayer()
fullyConnectedLayer(2) % Output layer with 2 neurons (clean or malware)
softmaxLayer() % Softmax activation for classification
classificationLayer() % Classification layer
];
% Specify training options
options = trainingOptions('adam', ...
'MaxEpochs', 10, ...
'MiniBatchSize', 32, ...
'ValidationData', {X_test, y_test}, ...
'Plots', 'training-progress');
% Train the CNN
net = trainNetwork(X_train, categorical(y_train), layers,
options); % Evaluate the trained CNN
```

```
y_pred = classify(net, X_test);  
accuracy = sum(y_pred == categorical(y_test)) / numel  
(y_test); fprintf('Accuracy: %.2f%%\n', accuracy * 100);
```

Results and Discussion

A. Confusion Matrix

The confusion matrix is a square matrix that provides a summary of the classifier's performance by counting the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The confusion matrix is typically used in binary classification problems, where each class is labeled as either positive or negative (in this case, clean or malware). It helps in understanding the classifier's ability to correctly classify the samples.

```
The confusion matrix is displayed using the 'disp' function in the MATLAB.  
confusionMat = confusionmat(actualLabels, predictions);  
disp('Confusion Matrix:');  
disp(confusionMat);  
Accuracy: 50.00%  
Confusion Matrix:  
53 43  
57 47
```

The top-left value (53) represents the number of true negatives (TN), indicating the number of clean files correctly classified as clean.

While the top-right value (43) represents the number of false positives (FP), indicating the number of clean files incorrectly classified as malware.

The bottom-left value (57) represents the number of false negatives (FN), indicating the number of malware files incorrectly classified as clean and the bottom-right value (47) represents the number of true positives (TP), indicating the number of malware files correctly classified as malware.

The accuracy of the classifier is calculated as the ratio of correct predictions to the total number of samples, resulting in an accuracy of 50.00%. It shows the number of samples that were correctly or incorrectly classified into their respective classes.

B. Dataset Training and Testing Results

The training and testing split to 50% each, it means that the researcher used 50% of the dataset for training the model and the remaining 50% for testing the model's performance. By running the code and computing the confusion matrix with this split, the confusion matrix and accuracy will reflect the performance of the model on the test dataset. The specific values in the confusion matrix and the accuracy will vary based on the characteristics of your dataset and the performance of your model. The confusion matrix will show the number of samples that fall into each category, indicating the true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP). The accuracy will represent the ratio of correct predictions to the total number of samples. By using a 50% training and 50% testing split, you can assess how well the model generalizes to unseen data and evaluate its performance in a more balanced manner. It provides a fair evaluation by testing the model on an independent dataset that was not used for training. The choice of the training and testing split is a design decision, and it depends on factors such as the size of the dataset, the complexity of the problem, and the desired trade-off between training and testing performance.

```
confusionMat = confusionmat(actualLabels, predictions);  
disp('Confusion Matrix:');
```

```
disp(confusionMat);
Accuracy: 52.20%
Confusion Matrix:
140 98
141 121
```

After training and testing the model, the confusion matrix is computed using the confusionmat function in MATLAB. The confusionmat function takes the actual labels (actualLabels) and predicted labels (predictions) as inputs and generates the confusion matrix.

The top-left value (140) represents the number of true negatives (TN), indicating the number of clean files correctly classified as clean.

The top-right value (98) represents the number of false positives (FP), indicating the number of clean files incorrectly classified as malware. The bottom-left value (141) represents the number of false negatives (FN), indicating the number of malware files incorrectly classified as clean.

The bottom-right value (121) represents the number of true positives (TP), indicating the number of malware files correctly classified as malware. In this case, the accuracy of the model is calculated as the ratio of correct predictions to the total number of samples, resulting in an accuracy of 52.20%.

The confusion matrix provides a detailed breakdown of the classification results. In this case, the confusion matrix reveals the following:

True Negatives (TN): The model correctly classified 140 clean files as clean. False Positives (FP): The model incorrectly classified 98 clean files as malware. False Negatives (FN): The model incorrectly classified 141 malware files as clean. True Positives (TP): The model correctly classified 121 malware files as malware. The accuracy of the model is calculated by summing the number of correct predictions (TN + TP) and dividing it by the total number of samples. In this case, the accuracy is determined to be 52.20%, indicating that the model's predictions were correct for approximately 52.20% of the total samples.

Then the training and testing is divided into 70% and 30%, respectively, meaning that 70% of the dataset is used for training the model, and 30% is used for testing the model's performance. After training and testing the model, the confusion matrix is computed and generates the confusion matrix.

The confusion matrix is then displayed below:

```
Accuracy: 49.33%
Confusion Matrix:
76 75
77 72
```

The confusion matrix reveals the following:

True Negatives (TN): The model correctly classified 76 samples as negative (e.g., clean files).

False Positives (FP): The model incorrectly classified 75 samples as positive (e.g., malware files) when they were actually negative.

False Negatives (FN): The model incorrectly classified 77 samples as negative when they were actually positive.

True Positives (TP): The model correctly classified 72 samples as positive.

The accuracy of the model is calculated by summing the number of correct predictions (TN + TP) and dividing it by the total number of samples. In this case,

the accuracy is determined to be 49.33%, indicating that the model's predictions were correct for approximately 49.33% of the total samples.

Review of Related Literature

A. Definition of Malwares, Artificial Intelligence (AI) and its applications in malware detection

Malwares, or malicious software, are programs that are created by hackers with the bad intention to compromise systems and premises. Unlike the unintentional errors in software, malware is used by hackers and attackers with wrong intent, variously using malwares such as viruses, worms, trojans, surveillance software (spyware) and ransomware (where the threat agent gets of your information and demands money to be able to get access back to your data).

In Figure 3, a volumetric attack is shown, the network is flooded with extreme traffic volumes at once which overloads the network, much the same way car traffic can overload city streets. According to Michael Karpowicz, director of research at NASK, "DDoS cyberattacks are meant to generate chaos, disrupt institutions, and of course cause financial losses. Traditional Malware Detection Methods. He also describes the problem using a familiar scenario. Imagine you're driving to the bank. There's little traffic and everything is going smoothly until you reach the final intersection. Suddenly, everything jams up and you must wait. That's a volumetric attack. Now imagine you have finally arrived at the bank and are standing in line for the teller. You notice that the customer in front of you has an unusual issue that requires the attention of every bank employee. Again, you must wait. That's an application attack. In a typical DDoS attack, a culprit uses many computers and online devices infected with a designed malware. These devices can include gadgets on the Internet of Things (IoT) devices.

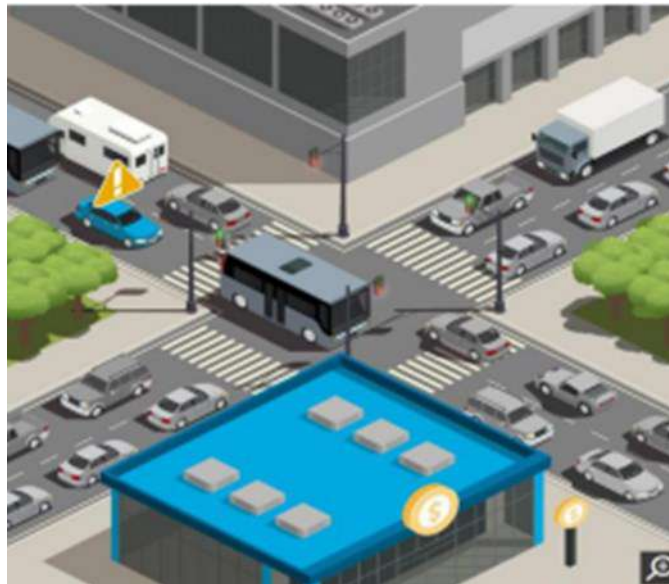


Fig. 3. Scenario of a Volumetric Attack

AI can be used in new innovations like blockchain, IoT, artificial intelligence, robots, and many other things. Security is one of the top issues for people who are using computer devices, and everyone wants to feel completely secure when traveling around every single day. (1)

B. Advantages and potential of using AI in malware detection

Hackers and cyber criminals are already updating their methodologies in detecting and intruding premises. The malwares are propagation by adversaries. There were many issues across the globe. Often it is found that malware is released in different countries for monetary gains. With the proliferation of malware spreading activities, it is made possible that now we have malware patterns that are used for training machine learning models. (2)

Artificial Intelligence (AI) can be used in advantage since hackers were also using AI in creating malicious codes. According to Sean Gallagher, a Principal Threat Researcher, from Sophos X-Ops, there are three Sophos AI projects that harness the model behind ChatGPT for better detection of malicious activity. The latest version of the Generative Pre-trained Transformer (GPT) model, GPT-3.5—the algorithmic brain of ChatGPT—has generated waves of both amazement and concern. Among those concerns is how it could be used for malicious purposes, including generating convincing phishing emails and even malware. According to Sophos AI Principal Data Scientist Younghoo Lee, the Sophos X-Ops researchers, have been examining ways to use an earlier version, GPT-3, as a force for good. He also presented some early insights into how GPT-3 could be used to generate human-readable explanations of attacker behavior.

C. Different Kinds of Malwares

To detect the malwares effectively, it is also important to know the different kinds of malwares. The malware needs to be detected before it affects a large number of systems. Recently, there have been made several studies on malware detection approaches. However, the detection of malware still remains problematic. Signature-based and heuristic-based detection approaches are efficient to detect known malware, but especially signature-based detection approach has failed to detect unknown malware. On the other hand, behavior based, model checking-based, and cloud-based approaches perform well for unknown and complicated malware; and deep learning-based, mobile devices based, and IoT-based approaches also emerge to detect some portion of known and unknown malware. (3)

Malwares exist in different forms; they are broadly categorized in following classes. They are not mutually exclusive although many of them exist in more than one class. Virus: It infects computers and other files by replicating itself. It cannot exist independently so it attaches with other files more precisely executable files and application and due to its replicating features, it spread across files and even computers through network. It cause system performance degradation and denial of service. (4). Other malware are Worms: These are malicious piece of code that exist independently. they have feature to replicate itself. They propagate through storage devices and emails, also consume network and computer resources which leads to system degradation in performance. As they can create multiple copies of themselves, antivirus scanners can identify these codes because of multiple existence. (5) Trojan Horse behaves like a useful program but it has harmful purpose. They do not replicate themselves but it transferred in a computer by internet interaction like downloading. It steals sensitive information, observe activity of users and can delete and alter or corrupt files on the system where it resides. (6) Rootkit takes control of the operating system such that it can hide itself or can make a safe environment for other malwares to hide in the system. Basically, it a masking technique to cheat antivirus so that they cannot find malwares in the system and consider them as normal applications. The Spyware are used to steal someone personal information or keep the watch on user's activities. It is installed without the knowledge of system owner and secretly collect the information and send it back to the creator. Even company with big names like google

also use spyware to collect the required information of their users. [7]

References

- AlTurjman, F., Salama, R.: An overview about the cyberattacks in grid and like systems. In: AlTurjman, F. (ed.) *Smart Grid in IoT-Enabled Spaces: The Road to Intelligence in Power*, pp. 233–247. CRC Press, Boca Raton (2020). <https://doi.org/10.1201/9781003055235-11>
- Al-Ani Mustafa Majid, Ahmed Jamal Alshaibi, Evgeny Kostyuchenko, Alexander Shelupanov, A review of artificial intelligence based malware detection using deep learning, *mMaterials Today: Proceedings*, Volume 80, Part 3, 2023, Pages 2678-2683, ISSN 2214-7853, <https://doi.org/10.1016/j.matpr.2021.07.012>. (<https://www.sciencedirect.com/science/article/pii/S221478532104858>)
- Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," in *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.
- Spafford, Eugene. "The internet worm incident." *ESEC'89* (1989): 446-468. [5] Li, Jun, and Shad Stafford. "Detecting smart, self-propagating Internet worms." *Communications and Network Security (CNS)*, 2014 IEEE Conference on. IEEE, 2014.
- Idika, Nwokedi, and Aditya P. Mathur. "A survey of malware detection techniques." *Purdue University* 48 (2007).
- Yin, Heng, et al. "Panorama: capturing system-wide information flow for malware detection and analysis." *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007.
- Rabia Tahir, "A Study on Malware and Malware Detection Techniques," *I.J. Education and Management Engineering*, 2018, 2, 20-30 Published Online March 2018 in MECS (<http://www.mecs-press.net>) DOI: 10.5815/ijeme.2018.02.03
- Sun, Baohua, et al. "Multi-modal Sentiment Analysis using Super Characters Method on Low-power CNN Accelerator Device." *arXiv preprint arXiv:2001.10179* (2020).
- [Y. Liu, X. Li and Z. Xiang, "The Effect of Chatbot-customer Interaction on Consumer Brand Advocacy: Exploring the Role of Chatbots," 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 2022, pp. 185-190, doi: 10.1109/ICEIEC54567.2022.9835050.