Published by
NIGERIAN SOCIETY OF PHYSICAL SCIENCES
Available online @ https://journal.nsps.org.ng/index.php/jnsps

J. Nig. Soc. Phys. Sci. 7 (2025) 2172

Journal of the Nigerian Society of Physical Sciences

An adaptive neuro-fuzzy inference system for multinomial malware classification

Amos Orenyi Bajeh*, Mary Olayinka Olaoye, Dare Muyiwa Mobolaji, Fatima Enehezei Usman-Hamza, Ikeola Suhurat Olatinwo, Peter Ogirima Sadiku, Abdulkadir Bolakale Sakariyah

Department of Computer Science, University of Ilorin, Ilorin, 240003, Nigeria

Abstract

Malware detection and classification are important requirements for information security because malware poses a great threat to computer users. As the growth of technology increases, malware is getting more sophisticated and thereby more difficult to detect. Machine learning techniques have been extensively used for malware detection and classification. However, most of them are binomial classifications that only detect the presence of malware but do not classify them into types. This study sets out to develop a multinomial malware classifier using an adaptive neuro-fuzzy inference system (ANFIS) and investigate the effectiveness of ANFIS in the classification. A first-order Sugeno ANFIS model was developed. It has five layers and uses two if-then rules. The ANFIS model was trained and tested with two prominent malware datasets from the Canada Institute of Cyber Security. The experimental results showed that the performance of the ANFIS model degrades as the size of the datasets increases, and the accuracy, precision, recall, and root mean square error is 94%, 0.88, 0.87, and 0.19 respectively.

DOI:10.46481/jnsps.2025.2172

Keywords: Malware, Adaptive neuro-fuzzy inference system, Artificial intelligence, Fuzzy logic, Artificial neural network

Article History :
Received: 02 June 2024
Received in revised form: 30 September 2024
Accepted for publication: 10 November 2024
Available online: 12 January 2025

Communicated by: Oluwatobi Akande

1. Introduction

Advancements in technology have led to the prolific use of computer systems and the Internet. According to the International Telecommunication Union (ITU) malware report, 51.2% of the population of the world (about 3.9 billion people) were using the Internet as at the end of 2018, and the number continues to increase every year [1]. This widespread use of computers has also brought about an increase in their malicious usage. Malware is a computer program designed for malicious purposes and thus poses threats to computer and information security. According to the 2020 malware report of Malwarebytes, more than 50 million malware events were detected in 2019 [2]. Malware programs are constantly increasing in number and sophistication. The Internet has made the transmission of malware easier than before, posing cyber security threats. Malware types include worms, trojan horses, backdoors, spyware, ransomware, bots, and rootkits.

The last decade has witnessed tremendous growth in using artificial intelligence for solving problems including cyber security threats. Several published studies on malware detection techniques have leveraged machine learning [3]. The number of research papers published in 2018 is 7,720, a 95% increase

*Corresponding author: Tel.: +234-809-227-2747.
Email address: bajehamos@unilorin.edu.ng (Amos Orenyi Bajeh)

from the number of publications in 2015, which in turn, is a 47% increase from the number of publications in 2010. This increase in the number of studies is the result of various factors including but not limited to the increasing number of publicly labeled feeds of malware, the increase in computational power, and the evolution of the machine learning field which achieved breakthrough success on a wide range of tasks [4]. Artificial neural network, a machine learning technique with the feature of processing information in parallel, is being extensively used for malware detection. For instance, *Damaševičius et al.* [5] proposed an ensemble-based classification using a neural network model to classify Windows portable executable (PE) malware, and the result of the study shows a performance accuracy of 98.8%. Even though the neural network is modeled after the human brain neurons, its output is not like humans, it cannot explain the decision. On the other hand, fuzzy systems can model data but cannot learn from it. The combination of a neural network and a fuzzy inference engine yields a hybrid system called artificial neuro-fuzzy inference system (ANFIS).

Recently, researchers have adopted several machine learning algorithms for the binomial classification of malware datasets into benign or malicious [5]. However, much research has not been conducted on the multinomial classification of malware. Malware should not only be detected but also categorized according to the type of malicious attack it was designed to perform [6]. This will facilitate applying precise and target solutions to the instance of malware attack. In this research, an Adaptive neuro-fuzzy inference system (ANFIS) will be developed for multinomial malware classification. This model will facilitate the classification of malware to their specific types.

The remaining part of this paper is organized as follows. Section 2 discusses malware. Section 3 presents the concept of the adaptive neuro-fuzzy inference system. Studies carried out on the detection and prediction of malware are reviewed in section 4, and section 5 presents the methodology used in the study including the study framework, data collection and preprocessing method, the model, and the performance evaluation measures. The results of the conducted experiments are presented and discussed in section 6, and the paper is concluded in section 7 with a summary of the paper and some future works.

2. Malware

Malicious software, known as malware, is a computer code designed to disrupt, damage, or gain unauthorized access to a computer system or network. Malware comes in many variants. Thus, there are numerous methods that they take to infect computer systems. Though varied in type and capabilities, malware usually has one or more of the following objectives:

1. Provide remote control for an attacker to use an infected machine.
2. Send spam from the infected machine to unsuspecting targets.
3. Investigate the infected user's local network.
4. Steal sensitive data.

The following sections discuss the prominent classes of malware that infect computer systems.

2.1. Ransomware

Ransomware is malicious software that restricts access to a computer or encrypts data until a ransom is paid in exchange for accessing the computer or device. An example is the decryptor that encrypts the computer files and demands a ransom through Bitcoin. Ransomware is one of the most dangerous malware rapidly spreading worldwide. The number of users affected by ransomware keeps growing along with an increase in malware modification. Recently, locker ransomware and crypto-ransomware have been extensively used for attacks. The locker ransomware locks the user out of the basic computer functions forcing the user to pay ransom to regain control. Crypto on the other hand encrypts files and other sensitive data threatening to destroy them unless a fee is paid.

2.2. Adware

Adware is a malicious program designed to display advertisements on a computer or mobile device directing the software to advertisement websites or collecting data about the user. It is usually bundled with other files and programs downloaded from the Internet. Changes are made to the browser homepage installing addon popups the user does not need. Besides causing users discomfort, the adware can slow down and subsequently crash the computer.

2.3. Scareware

Scareware is malicious software that tricks computer users into visiting malware-infested websites. It is also known as deception software, rogue scanner software, or fraud software. Scareware may come as pop-ups appearing as legitimate warnings from antivirus software companies claiming your computer's files have been infected. Thus, users are frightened into paying for software that will fix the problem. What they end up downloading, however, is fake antivirus software (malware) intended to steal the victim's data.

Fraudsters also use other tactics, such as sending out spam mail to distribute scareware. Once that email is opened, victims are fooled into buying worthless services. Falling for these scams and releasing your credit card information opens the door for future identity theft crimes.

2.4. Virus

Malicious software that replicates by copying itself to another program. A virus aims to infect the vulnerable system, gain administrator control, and steal sensitive data. Computer viruses spread through emails, installation of executables, downloading infected software, and through storage devices such as USB drives and external hard disks. As soon as a virus gets into the computer it replicates itself. There are different types of viruses. Boot sector virus takes control when booting a computer. Web scripting viruses exploit the code of web browsers and web pages. The residence virus installs itself into the computer and gets executed at any moment. The polymorphic virus changes its code each time the file containing it is executed.

2.5. Trojan

Malicious software that looks harmless for users to download. Trojans can give attackers back door control over the victim's computer. They can also capture keyboard strokes or steal sensitive information such as passwords and credit card pins. Usually, they get downloaded on a computer and disguised as legitimate programs through emails or free-to-download files and programs. The delivery method typically sees an attacker use social engineering to hide malicious code within legitimate software and gain access to the victim's system. Once downloaded, the trojan malicious code will execute the task for which it was designed, such as gaining backdoor access to corporate systems, spying on users' online activity, or stealing sensitive data. The presence of Trojan malware can be indicated by unusual activity such as unexpected and unsolicited changes in computer settings.

3. Machine learning, fuzzy logic, and ANFIS

This section presents the concept of machine learning (ML), fuzzy logic which underpins fuzzy inference systems, and adaptive neuro-fuzzy inference system (ANFIS).

3.1. Machine learning

ML is a sub-field of artificial intelligence that involves endowing computers to act automatically by learning patterns in data and using the knowledge to analyze new sets of related data in the domain of interest. This concept has been applied in self-driving cars, practical speech recognition, effective web agents, and a vastly improved understanding of the human genome. New areas such as telecommunication networks are adopting ML. There are three ML approaches: supervised learning, unsupervised learning, and reinforcement learning.

In the supervised learning approach, a pre-labeled dataset set is used to train an ML model. After the training process, a test dataset is used to validate the model to ensure that the model performs as expected. This technique is widely used today in medical diagnostics of diseases such as diabetes and cancer. Also in the stock market, machine learning is used to study trends and predict which stock to invest in based on the trend. Unsupervised learning is the type of ML in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. The goal of unsupervised learning is to find the underlying structure of the dataset, group that data according to similarities, and represent that dataset in a compressed format. Reinforcement Learning technique allows the system to determine the best way of taking an action or performing a task automatically. This is made possible by a reward system in which any action wrongly taken attracts a penalty which will enable the system to re-adjust. Also, any action taken correctly will be rewarded to enable the system to perform better. It is successfully applied only in areas where huge amounts of simulated data can be generated, like robotics and games.

One of the most popular ML approaches is artificial neural networks.

3.2. Artificial Neural Network (ANN)

A Neural Network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

3.3. Fuzzy Logic (FL)

FL is a method of decision-making that mimics human reasoning. Like human reasoning, the FL approach involves all intermediate possibilities between the digital values YES and NO or 1 and 0. The conventional logic block that a computer can understand takes crisp input and produces a definite output as Yes or No, True or False. The inventor of fuzzy logic, Lotfi Zadeh, observed that, unlike computers, the human decision-making dataset includes a range of possibilities between YES and NO, such as certainly yes, possibly yes, cannot say, possibly no, certainly no.

3.4. Adaptive Neuro-Fuzzy Inference System (ANFIS)

An adaptive neuro-fuzzy inference system or adaptive network-based fuzzy inference system (ANFIS) is a kind of artificial neural network based on a Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has the potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy IF-THEN rules that have the learning capability to approximate nonlinear functions. Hence, ANFIS is a universal estimator. For efficient and optimal use of the ANFIS model, the tuning parameters can be optimized by using optimization algorithms such as the genetic algorithm [7].

4. Related works

The exponential growth of malware has created a significant threat in our daily lives which heavily rely on computers running all kinds of software. Malware writers using innovative approaches create several variants of malicious software by using techniques such as packing and encrypting techniques. Malicious software classification and detection play an important role and a big challenge for cyber security research [8]. Studies have been carried out on the detection or classification of computer malware. This section presents some notable studies in this area.

Tanuwidjaja *et al*. [9] proposed a malware detection scheme based on modified deep abstraction and weighted feature selection. In this study, experiments were performed to find the optimum setting for both stack autoencoder network (SAE) and artificial neural network (ANN). The experiment shows that the best result is given by the combination of SAE with 2 hidden layers and ANN with 2 hidden layers. The experimental result shows a 95.490% detection rate, 2.007% false alarm rate, and 6 times faster compared to deep abstraction weighted features selected.

Most malware detection techniques include signature-based, specification-based, and static-based. These techniques have high false positives, low accuracy, and an inability to detect zero-day and polymorphic malware. Sodiya *et al*. [10] modeled an Adaptive Neuro-Fuzzy System for Malware Detection (ANFSMD) to address the problems of high false positives, low accuracy, and inability to detect zero-day attacks. ANFSMD utilizes the Application Programming Interface (API) calls and operation codes to study the behaviour of Portable Executable (PE) files. The PE files were disassembled into low-level codes and the identified features were grouped for efficient detection. Five features used for fuzzification were selected using a weighted average [10].

Nugraha [11] studied the existing malware classification algorithm's features, applied the existing algorithm, and evaluated the algorithm for malware classification. The algorithm used is random forest and gradient boost. This algorithm can detect malware with high accuracy up to 99.3%. The results showed an accuracy of 99.42% and 98.73% for the random forest, and gradient boosting respectively.

Rabadi & Teo [12] proposed an advanced window method for malware detection and classification. Machine learning algorithms such as SVM, random forest, and decision trees were used as classifiers. A lightweight API-based dynamic feature extraction technique was adopted. Experimental results show that the developed model could reach an accuracy of over 99.89% and outperform many state-of-the-art API-based malware detectors.

Damaševičius *et al*. [5] proposed ensemble model based on a neural network for malware classification. The first-stage classification was performed by a stacked ensemble of dense (fully connected) and convolutional neural networks (CNN). At the final stage, the classification was performed by a meta-learner. The experimental result shows an accuracy of 98.8%.

Khammas [13] proposed malware detection using sub-signatures and machine learning techniques. In this five machine learning classifiers were used (IBK, SVM, Decision Tree (J48), Naïve Bayes, and AdaBoostIM) to detect malware at the host level. General malware files that contain three different types of malware (Trojan, worm, and virus) were input into the developed model. The result shows an accuracy of 99.78% and a zero false-positive rate.

Azeez *et al*. [1] proposed Windows PE malware detection using ensemble learning. In this approach, an ensemble model of a deep neural network was developed. The base classification was done by a stacked ensemble of fully connected and one-dimensional convolutional neural networks (CNNs). A machine learning algorithm did the final classification. Five machine learning algorithms were used, naive Bayes, decision tree, random forest, gradient boosting, and AdaBoosting. The result shows an accuracy of 98.62%.

Lu *et al*. [14] proposed android malware detection based on a hybrid deep learning model. An android malware detection model based on a hybrid deep learning model with a deep belief network (DBN) and gated recurrent unit (GRU) was developed. Android malware was analyzed after extracting static features, and dynamic behavioral features with strong anti-obfuscation ability were also extracted. The extracted features were then input into the system for training and prediction. The result shows an accuracy of 96.58%.

Yousefi-Azar *et al*. [15] proposed a malware detection scheme. The proposed system analysis is based on a neural network. Three phases were implemented by a neural network with two hidden layers and an output layer. Tf-smashing was used for the feature extraction. The developed model was evaluated on both Android and Windows OS. The result shows an F1-score of 97.21% and 99.45% on Android dex files and Windows PE files respectively, in the applied datasets.

Liu *et al*. [16] proposed a graph-based feature generation approach in Android malware detection with machine learning techniques. This approach highlights two major distinguishing aspects: context-based feature selection and graph-based feature generation. An Android application was considered as a collection of reduced CFDs (interpret control flow graphs) and original features from these graphs were extracted. The results show an accuracy of 95.4% and a recall of 96.5%.

Shhadat *et al*. [17] proposed the use of machine learning techniques to advance the detection and classification of unknown malware. A more enhanced feature set was presented using the random forest to decrease the number of features. KNN, SVM, NB, RF, LR, and DT algorithms were applied on a benchmark dataset in the experiments. Results achieved accuracy improvements overall binary and multi-classifiers. The highest accuracy (98.2%) was achieved by decision trees for binary classification and 95.8% by random forest algorithms for multi-class classification. The lowest accuracy was achieved by naıve Bayes with an accuracy of 91% and 81.8% for binary classification and multi-class classification, respectively.

Alzaylaee *et al*. [18] proposed deep learning-based Android malware detection using real devices. Experiments were performed with over 30,000 applications (benign and malware) on real devices. Furthermore, experiments were conducted to compare the detection performance and code coverage of the stateful input generation method with the commonly used stateless approach using the deep learning system. The result shows that the developed system achieved up to 97.8% detection rate (with dynamic features only) and 99.6% detection rate (with dynamic and static features) respectively which outperforms traditional machine learning techniques.

Gao *et al*. [2] developed a malware classification for the cloud via semi-supervised transfer learning. A byte classifier based on a recurrent neural network (RNN) for its detection component was designed to detect malware. The accuracy of the byte classifier was only 94.72% after supervised learning.

An ASM classifier was proposed for the prediction component, and it achieves 99.69% accuracy. The transfer component invokes the prediction component to classify an unlabeled dataset, and it combines the predicted labels and byte features of the unlabeled dataset into a new training dataset. The new dataset was transferred to the byte classifier for training again. The result shows that semi-supervised transfer learning improved the accuracy of the detection component from 94.72% to 96.9%.

Aiterher *et al*. [19] proposed a neural fuzzy classifier based on Adaptive Neuro-Fuzzy Inference System (ANFIS) for malware detection. Firstly, the malware exe files were analyzed and the most important API calls were selected and used as training and testing datasets. using the training data set the ANFIS classifier learned how to detect the malware in the test dataset. Result shows, the performances of the Neuro fuzzy classifier were evaluated based on the performance of training and accuracy of classification, that the proposed Neuro fuzzy classifier can detect the malware exe files effectively.

Asrafi [8] classified eight malware according to their family. Four feature selection algorithms were provided to select the best feature for the multiclass classification problem. The top 100 features were selected and used for machine learning modeling. Five machine learning algorithms are compared to find the best models. The frequency distribution of features is found by ranking the features of the best model. It was concluded that the frequency distribution of every character of the API call sequence can be used to classify the malware family.

Some researchers have used static based approach, machine learning, controlled called graph to analyze malware. A static based approach was used to detect malware by applying ordered of weighted averaging and the method of a parameterized family of aggregate operators was used to select prominent features from malware [20]. Jamuna and Edwards [21] used Network-based techniques used to monitor the traffic produced by some categories of malware.

Vinod *et al*. [22] classified the technique that can be used for malware detection into anomaly-based detection and signature-based detection. Anomaly-based detection uses the knowledge of what is considered normal to determine if a file is malicious or not while signature-based detection searches for known patterns of data within an executable. Eskandari and Hashemi [20] and Schultz *et al*. [35] identified two major challenges that current signature-based techniques are faced with. Firstly, they can only detect malware whose behaviours are closed to the be/haviours of the known signature and any significance difference will make the malware to be unnoticed. Secondly, malware is generally unpredictable, and this frequently causes false alarms.

The following studies proposed hybrid approaches to malware detection.

In Ref. [23], Yoo *et al*. proposed an advanced hybrid approach using random forest and deep learning for malware. The developed hybrid model combines a random forest and a deep learning model using 12 hidden layers to determine malware and benign files, respectively. This model also includes certain proposed voting rules to make final decisions. In an experiment involving 6,395 a typical sample, the hybrid decision model achieved a higher detection rate (85.1% and standard deviation of 0.006) than that of the prior model (65.5%) without voting rules.

Andrade *et al*. [24] proposed a model based on LSTM neural networks to identify five different types of malware. The expert result shows a True positive rate (TPR) of 92.19% with the rootkit class which was the best result. The worst result was with the trojan class with 51.06%, the rootkit class has the best false positive rate (FPR) of 3.07% followed by the worm class with 3.93%. The trojan class got the worst result, 11.53%. Compressing the classes into two classes, clean wares, and malware, an accuracy of 90.63%, a TPR of 92.76%, followed by an FPR of 8.16% and FNR of 7.24% was achieved. In Ref. [25], Norouzi et al. used memory access patterns to distinguish malicious families. The feature selection process was performed and trained by ML Models. They took 50,000 features by selecting the highest information gain (IG) ranking. CFS in Weka was also performed to find the best 10,000 features. However, the accuracy was 0.845% with RF classifiers in their work.

Xiaofeng *et al*. [26] used API calls for the TF-IDF feature selection algorithm on 552 malicious and benign datasets and achieved 96.4% accuracy. Banin & Dyrkolbotn [6] investigated the use of low-level features of memory access patterns for the detection and classification malware into families and types. Six (6) machine learning methods were modeled for the classification. The study selected 1000 malware consist of 100 from each of 10 types and 10 families of malware from a dataset created under the initiative of Testimon research group (https://testimon.ccis.no/). The malware types used for the study include backdoor, pws, rogue, trojan, trojandownloader, trojandropper, trojanspy, virtool, virus, worm; while the malware families considered in the study include agent, hupigon, obfuscator, onlinegames, renos, small, vb, vbinject, vundo, and zlob. The study assessed the performance of the machine learning models in terms of accuracy. The best accuracy recorded for malware family classification is 0.688, and for malware type classification is 0.845 by the RF model.

Al-Andoli *et al*. [27] presented a hybride deep leaning model for malware detection. The model combined particle swarm optimization for optimization and backpropagation. A parallel computing architecture was used to execute the deep learning to maximize efficiency and scalability. Several datasets were used in the study. The experimental results showed that the proposed technique is effective at detecting malware

Arif *et al*. [28] proposed a risk-based fuzzy analytical hierarchy process-based multi-criteria decision-making mobile malware detection system for analyzing Andrios applications. This study focused on static analysis which analyses mobile applications using permission-based characteristics for malware detection. Danger analysis was used to make mobile users more conscious of the possibility that each permission request they approve might include a significant degree of danger. The experimental result showed an accuracy of 90.54%.

In Ref. [29], Djenna *et al*. developed dynamic deep learning-based models combined with heuristic approaches to detect and classify five malware families: adware, Radware,

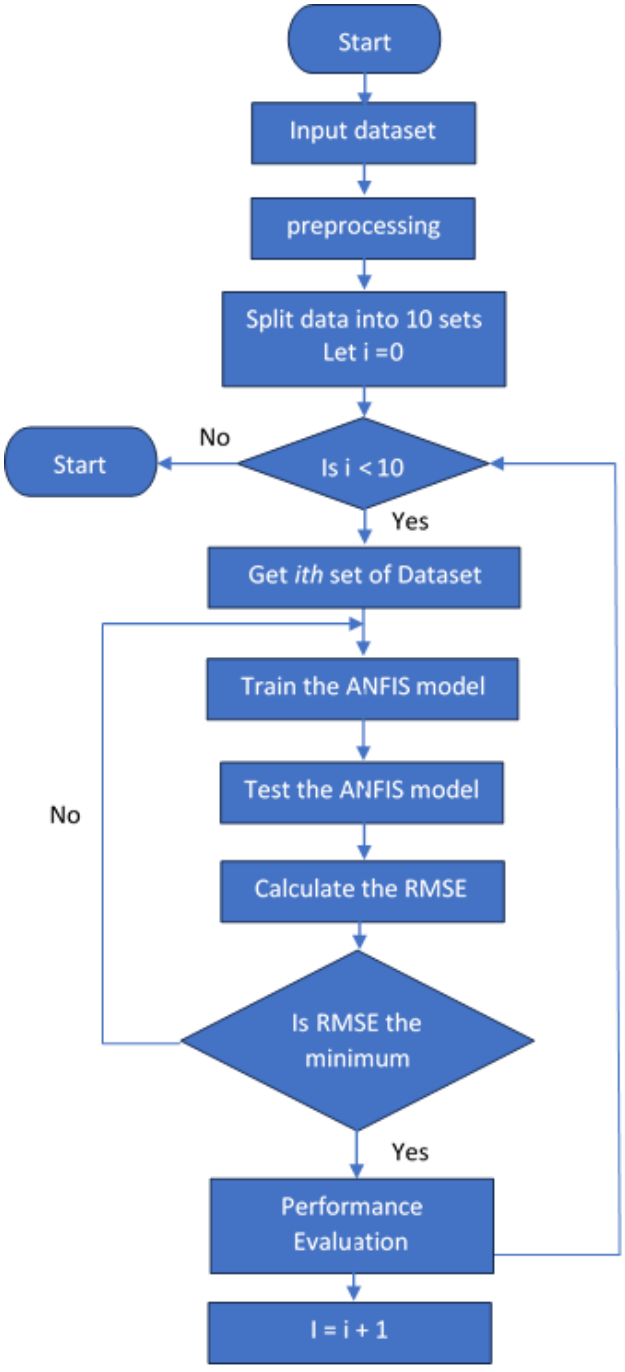

Figure 1. Model Development Process Flow.

rootkit, SMS malware, and ransomware. The CICAndMal2017 dataset by the Canadian Institute for Cybersecurity was used in the study. Deep learning and machine learning approaches were used in this study. CNN and DNN deep learning methods, and random forest and decision tree machine learning models were developed. The study showed that the combination of deep learning and heuristic approaches performs better than the use of static deep learning methods.

Masud *et al*. [30] researched on IG feature selection method and used a malware and benign dataset and achieve 95.7 percent accuracy.

The Intrusion Detection System architecture commonly used in commercial and research systems has several problems that limit their configurability, scalability, or efficiency [31]. Chavan *et al*. [31] applied two machine-learning paradigms, Artificial Neural Networks and Fuzzy Inference Systems to design an Intrusion Detection System. SNORT was used to perform real-time traffic analysis and packet logging on the IP network during the training phase of the system. Then a signature pattern database was constructed using protocol analysis and the Neuro-Fuzzy learning method.

5. Methodology

This section presents the study framework, the dataset and its preprocessing, the ANFIS architecture, and the performance measures used in the study.

5.1. Study framework

Figure 1 depicts the study strategy. The process starts by passing the dataset through a preprocessing stage. A preliminary study of ANFIS models showed that the performance of the model degrades as the size of the dataset increases; see the results in Figures 6 – 25 and Tables 2 - 21 where larger size of datasets are used compared to the results in Figure 26 and Table 22 for smaller size of dataset, the model performed better on te smaller dataset. Thus, the preprocessed dataset is divided into 10 subsets. Each subset is split into training and testing datasets using 70:30 ratio respectively. The training set served as input into the designed model for training. After that, the testing set is used to validate the developed model. The preprocessed training dataset is inputted into the ANFIS model for training. At every instance of training, the root means square error is calculated. Training stops when the mean square error is at a stable minimum value.

5.2. Data acquisition and preprocessing

Two prominent datasets, CICMalDroid-2020 and CCCS-CIC-AndMal-2020, from the Canada Institute for Cyber Security, are considered in this study. The CICMalDroid-2020 and the CCCS-CIC-AndMal-2020 datasets contain 11,598 and 195,624 instances. Ten malware families: Adware, Banking Malware, Mobile Malware, SMS malware and Benign, are considered in the CICMalDroid-2020. Nine malware families: Adware, Banking Malware, Riskware, Backdoor, File infector, Ransomware, Scareware, Zero-day malware and Benign, are considered in the CCCS-CIC-AndMal-2020. Some of the features of the datasets are described in Table 1. Data preprocessing is one of the important and prerequisite steps in data mining when the dataset consists of incomplete (missing), noisy(outliers), and inconsistent data, In this study, the Pearson correlation coefficient was used for the data preprocessing because it gives information about the magnitude of the association, or correlation, as well as the direction of the relationship.

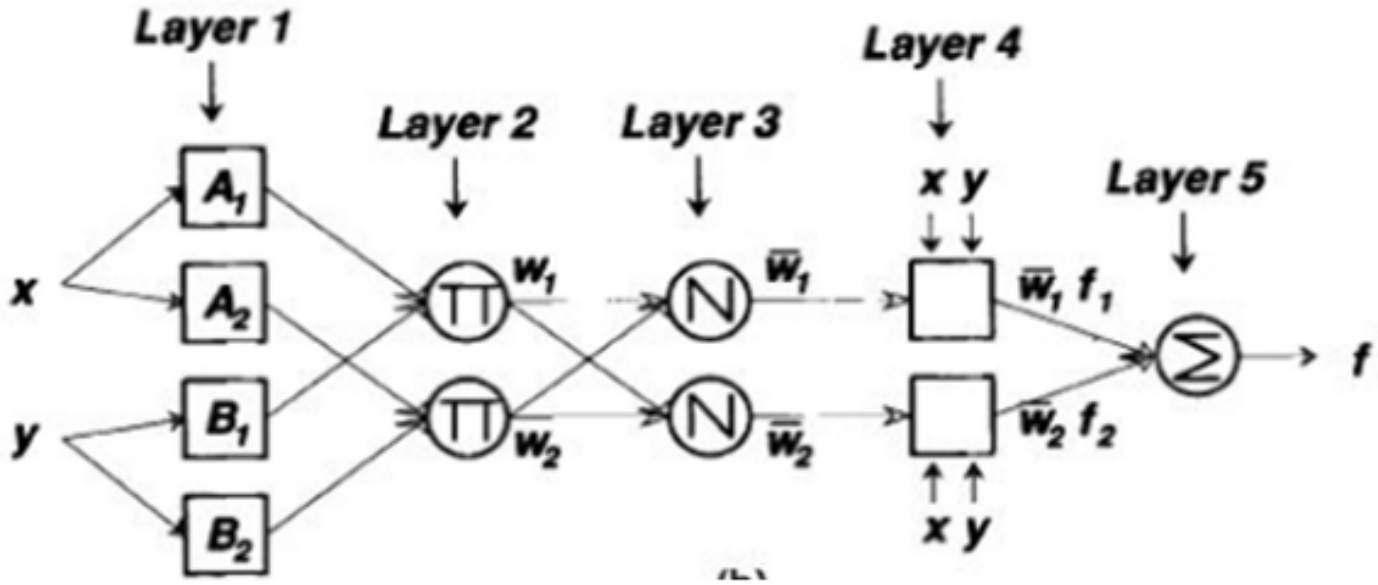

Figure 2. ANFIS architecture (Mitiku & Manshahia [32].

Table 1. Features of the Dataset.

Feature	Data type	Description
Destination Port	Numeric	Destination port number or traffic used by the software.
The total length of forwarding packets	Numeric	The total length of a forwarding packet from software or program
FWD Pack length maximum	Numeric	A maximum number of packets sent by software.
FWD Pack length minimum	Numeric	A minimum number of packets sent. By software.
Backward packet length STD	Numeric	Backward packet length Standard deviation

5.3. *Adaptive neuro-fuzzy inference system*

In this study, the first-order Sugeno model of adaptive neuro-fuzzy (ANFIS) is used [5]. The first-order Sugeno model ANFIS architecture uses two IF-THEN rules as follows:

Rule 1:

If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$

Rule 2:

If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$

In the rules, x and y denote the input, while A_i and B_i represent the fuzzy sets used in the model.

The model consists of 5 layers as shown in Figure 2. The function of each of the layers is described in the following sections.

5.3.1. *Fuzzification*

Fuzzification takes place in the first layer of ANFIS. The numerical inputs are converted into a membership value computed using the membership function subscript fuzzy sets defined in the model. A membership function $\mu_{\mathcal{A}}$ of a set A on a universe of discuss X is define as $\mu_{\mathcal{A}}: X \rightarrow [0, 1]$. Every node in the first layer acts as a membership function, and its output is referred to as membership value or degree of membership which falls between 0 and 1. The following equation applies to layer 1 [7]:

$$O_1, i = \mu Ai\,(x), \quad for\ i = 1, 2,\ or \tag{1}$$

$$O_1, i = \mu Bi - 2\,(y), \quad for\ i = 3, 4, \tag{2}$$

$$\mu_{A(x)} = \frac{1}{1 + \left|\frac{x - c_i}{a_i}\right|^{2b}}. \tag{3}$$

The model uses 3 Gaussian membership functions per input. Grid partition was used to generate the membership function. At the beginning of training, this method divides the data space into rectangular sub-spaces using axis-paralleled partition based on a predefined number of membership functions and their types in each dimension.

5.3.2. Firing strength of fuzzy rules

Every node in layer 2 multiplies the incoming signals and passes the product to the next layer. The output of each node is referred to as the firing strength of a rule [5]. Equation 4 states the computation of the firing strength of the fuzzy rules where,

$$i = wi = \mu Ai(x)\,\mu Bi(y) \tag{4}$$

$$i = 1, 2.$$

5.3.3. Normalize firing strength

Each node in layer 3 calculates the normalized firing strength of a rule which is the ratio of the *ith* rule firing strength to the sum of all rules' firing strengths [5].

$$O_3,\ i\ =\ \varpi\ =\ \frac{w_i}{w1 + w2}, i = 1, 2. \tag{5}$$

The outputs of this layer are called normalized firing strengthens.

5.3.4. Combining independent variable with dependent variable

Every node j in layer 4 takes the weighted normalized firing strength values and combines them with the original inputs from the training data set to calculate an output called weighted consequent value as depicted in equation (6) [5].

$$O_4, i = wifi = i(pix + qiy + ri)\,. \tag{6}$$

5.3.5. Prediction and final output

The final step is layer 5 which determines the sum of all incoming signals and applies them to a test dataset to output a predicted value. This step also coordinates the de-fuzzification process which converts the data back to meaningful crisp values as shown in equation (7) [5]. This step also coordinates the de-fuzzification process which converts the data back to meaningful crisp values as shown in equation (7).

$$O_i = \frac{\sum_i w_i f_i}{\sum_i w_i}. \tag{7}$$

In this study, MATLAB programming environment was used for analysis, because neuro-fuzzy was effectively implemented.

5.4. Performance evaluation

The proposed model was evaluated using root mean square error, recall, precision, and accuracy. They are computed using equations (8) – (11) respectively.

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_1 - y_2)^2}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}, \tag{9}$$

$$Precision = \frac{TP}{TP + FP}, \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}, \tag{11}$$

where TP is true positive, TN is true negative, FP is false positive and FN is false negative. They are measured from the confusion matrix obtained during the experiments.

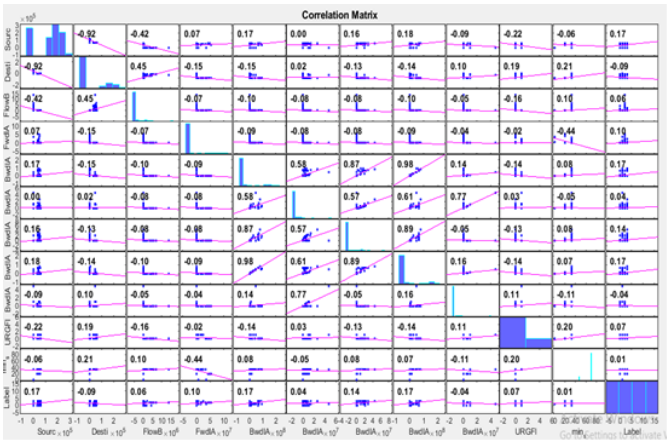

Figure 3. Features Correlation matrix.

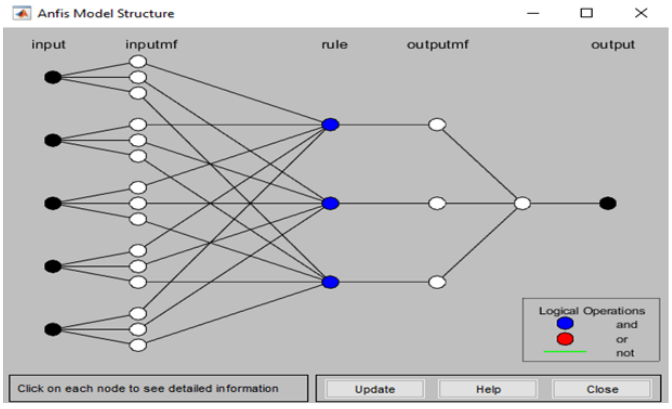

Figure 4. Generated ANFIS Structure.

6. Results and discussion

This section presents the results of the data preprocessing, the ANFIS model, and the performance analysis of the model.

6.1. Data preprocessing

The features with a high correlation with the label or outcome variable were selected as the important features for the ANFIS model. Figure 3 shows the correlation matrix of the dataset. The matrix clearly shows that some features have a negative correlation while others have a positive correlation. Features with negative correlation are not suitable representatives of the dataset. Only five features with the highest positive correlation values were used to create the ANFIS model while others were eliminated. The label attribute values were encoded as 0, 1, 2, 3 to represent benign, adware, scareware, and ransomware respectively.

6.2. Experimental results of the developed ANFIS model

Figure 4 depicts the structure of the developed ANFIS model. It has five inputs and one output. Every input variable uses a Gaussian membership function.

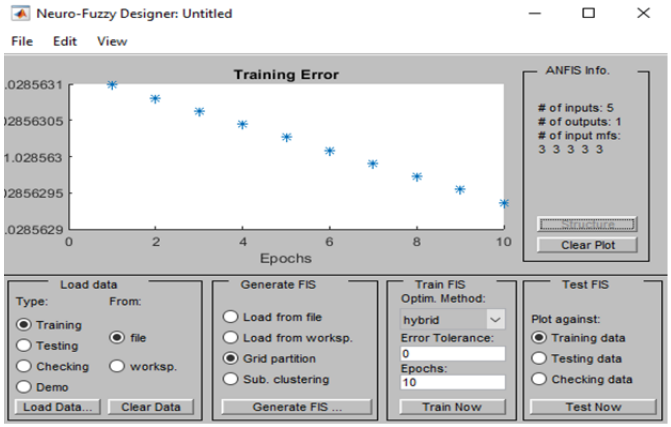

Figure 5. Training ANFIS.

Table 2. Subset_1 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.87	0.75	0.59	0.66
Banking Malware	0.88	0.67	0.68	0.67
SMS Malware	0.87	0.73	0.71	0.72
Riskware	0.83	0.59	0.55	0.57
Benign	0.87	0.60	0.81	0.68

6.3. *Result of ANFIS training*

The model was trained with 70% of the dataset for 10 epochs. Figure 5 depicts the training processing of the model using the ANFIS fuzzy logic toolbox in MATLAB. As shown in the figure, the 10 epoch was experimentally determined; several epochs were tried and it was observed that the model performs to its peak with 0 error tolerance at 10 epochs. Also, the hybrid method was selected as the optimization function. The hybrid method consists of backpropagation for the parameters associated with the input membership functions, and least squares estimation for the parameters of the output membership functions. The training process tunes the parameters of the membership functions of the FIS, thereby modeling the relationship between input and output data.

6.4. *Classification result*

The prediction results of the ANFIS model on the 10 subsets of each of the two datasets are presented in this section.

Each subset of the CICMalDroid-2020 dataset contain 348 instances of the 5 families of malware considered. Figures 6 – 15 presents the classification confusion matrix of the ANFIS model. The corresponding performance metrics for each subset are presented in Tables 2 – 11 respectively.

Similarly, each of the subsets of the CCCS-CIC-AndMal-2020 contains 2100 samples of the 10 malware families considered in this study. Figures 16 – 25 depict the classification confusion matrix while the corresponding performace metrics

Table 3. Subset_2 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.87	0.75	0.59	0.66
Banking Malware	0.87	0.65	0.65	0.65
SMS Malware	0.87	0.74	0.72	0.73
Riskware	0.83	0.59	0.55	0.57
Benign	0.88	0.60	0.82	0.69

Table 4. Subset_3 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.88	0.78	0.62	0.69
Banking Malware	0.88	0.66	0.65	0.65
SMS Malware	0.87	0.75	0.69	0.72
Riskware	0.83	0.61	0.59	0.60
Benign	0.88	0.60	0.84	0.70

Table 5. Subset_4 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.87	0.73	0.60	0.66
Banking Malware	0.89	0.70	0.69	0.70
SMS Malware	0.86	0.76	0.72	0.74
Riskware	0.83	0.59	0.54	0.56
Benign	0.89	0.57	0.81	0.67

Table 6. Subset_5 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.90	0.79	0.73	0.76
Banking Malware	0.88	0.71	0.59	0.64
SMS Malware	0.89	0.78	0.74	0.76
Riskware	0.89	0.68	0.76	0.72
Benign	0.92	0.74	0.88	0.81

Table 7. Subset_6 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.90	0.83	0.60	0.70
Banking Malware	0.89	0.70	0.71	0.70
MS Malware	0.86	0.73	0.72	0.73
Riskware	0.83	0.59	0.56	0.58
Benign	0.87	0.58	0.79	0.67

Table 8. Subset_7 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.88	0.75	0.64	0.69
Banking Malware	0.89	0.72	0.63	0.67
SMS Malware	0.84	0.76	0.76	0.76
Riskware	0.84	0.60	0.51	0.55
Benign	0.89	0.54	0.79	0.64

Table 9. Subset_8 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.86	0.70	0.58	0.63
Banking Malware	0.87	0.67	0.56	0.61
SMS Malware	0.82	0.73	0.68	0.70
Riskware	0.83	0.59	0.52	0.55
Benign	0.87	0.51	0.81	0.62

Table 10. Subset_9 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.86	0.72	0.53	0.61
Banking Malware	0.85	0.60	0.47	0.53
SMS Malware	0.79	0.65	0.64	0.65
Riskware	0.80	0.52	0.41	0.46
Benign	0.84	0.45	0.81	0.57

Table 11. Subset_10 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.86	0.73	0.51	0.60
Banking Malware	0.83	0.52	0.42	0.46
SMS Malware	0.75	0.64	0.64	0.64
Riskware	0.78	0.43	0.32	0.37
Benign	0.83	0.39	0.73	0.51

Table 12. Subset_1 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.62	0.69	0.66
Banking Malware	0.92	0.57	0.59	0.58
SMS Malware	0.92	0.67	0.59	0.62
Riskware	0.93	0.59	0.58	0.59
Backdoor	0.93	0.68	0.65	0.66
File Infector	0.92	0.64	0.53	0.58
Ransomware	0.92	0.59	0.76	0.66
Scareware	0.94	0.70	0.65	0.67
Zero-Day	0.93	0.65	0.70	0.67
Benign	0.92	0.59	0.55	0.57

Table 13. Subset_2 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.62	0.69	0.66
Banking Malware	0.92	0.57	0.59	0.58
SMS Malware	0.92	0.67	0.59	0.62
Riskware	0.93	0.59	0.58	0.59
Backdoor	0.93	0.68	0.65	0.66
File Infector	0.92	0.64	0.53	0.58
Ransomware	0.92	0.59	0.76	0.66
Scareware	0.94	0.70	0.65	0.67
Zero-Day	0.93	0.65	0.70	0.67
Benign	0.92	0.59	0.55	0.57

Table 14. Subset_3 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.62	0.69	0.66
Banking Malware	0.92	0.58	0.60	0.59
SMS Malware	0.92	0.67	0.59	0.62
Riskware	0.93	0.59	0.58	0.59
Backdoor	0.93	0.68	0.64	0.66
File Infector	0.93	0.64	0.53	0.58
Ransomware	0.92	0.59	0.76	0.66
Scareware	0.93	0.71	0.65	0.67
Zero-Day	0.93	0.65	0.70	0.67
Benign	0.92	0.60	0.55	0.57

Table 15. Subset_4 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.58	0.58	0.58
Banking Malware	0.93	0.67	0.64	0.66
SMS Malware	0.92	0.64	0.53	0.58
Riskware	0.92	0.58	0.74	0.65
Backdoor	0.93	0.69	0.64	0.66
File Infector	0.93	0.64	0.69	0.66
Ransomware	0.92	0.59	0.54	0.56
Scareware	0.92	0.58	0.58	0.58
Zero-Day	0.93	0.67	0.64	0.66
Benign	0.92	0.64	0.53	0.58

Table 16. Subset_5 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.93	0.64	0.70	0.67
Banking Malware	0.93	0.68	0.61	0.68
SMS Malware	0.92	0.68	0.59	0.63
Riskware	0.93	0.61	0.59	0.60
Backdoor	0.93	0.67	0.67	0.67
File Infector	0.92	0.63	0.52	0.57
Ransomware	0.93	0.61	0.76	0.68
Scareware	0.94	0.70	0.65	0.67
Zero-Day	0.93	0.64	0.70	0.67
Benign	0.92	0.58	0.56	0.57

Table 17. Subset_6 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.93	0.64	0.71	0.67
Banking Malware	0.93	0.61	0.61	0.61
SMS Malware	0.92	0.67	0.62	0.64
Riskware	0.92	0.56	0.56	0.56
Backdoor	0.93	0.67	0.65	0.66
File Infector	0.92	0.64	0.50	0.56
Ransomware	0.93	0.64	0.76	0.69
Scareware	0.93	0.65	0.63	0.64
Zero-Day	0.93	0.52	0.71	0.66
Benign	0.92	0.59	0.57	0.58

Table 18. Subset_7 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.93	0.65	0.70	0.68
Banking Malware	0.92	0.60	0.57	0.58
SMS Malware	0.92	0.67	0.60	0.63
Riskware	0.93	0.59	0.54	0.57
Backdoor	0.93	0.70	0.69	0.69
File Infector	0.93	0.66	0.55	0.60
Ransomware	0.93	0.63	0.76	0.69
S careware	0.93	0.66	0.65	0.65
Zero-Day	0.93	0.61	0.71	0.65
Benign	0.92	0.56	0.56	0.56

Table 19. Subset_8 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.93	0.67	0.69	0.68
Banking Malware	0.93	0.63	0.60	0.62
SMS Malware	0.93	0.69	0.62	0.65
Riskware	0.93	0.61	0.59	0.60
Backdoor	0.93	0.68	0.66	0.67
File Infector	0.91	0.62	0.52	0.57
Ransomware	0.93	0.63	0.74	0.68
Scareware	0.93	0.67	0.67	0.67
Zero-Day	0.93	0.61	0.72	0.66
Benign	0.91	0.54	0.55	0.55

Table 20. Subset_9 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.61	0.67	0.64
Banking Malware	0.93	0.61	0.61	0.61
SMS Malware	0.93	0.70	0.59	0.64
Riskware	0.93	0.60	0.54	0.57
Backdoor	0.93	0.66	0.64	0.65
File Infector	0.91	0.61	0.47	0.53
Ransomware	0.92	0.60	0.75	0.66
Scareware	0.94	0.66	0.63	0.65
Zero-Day	0.93	0.60	0.72	0.65
Benign	0.92	0.56	0.57	0.57

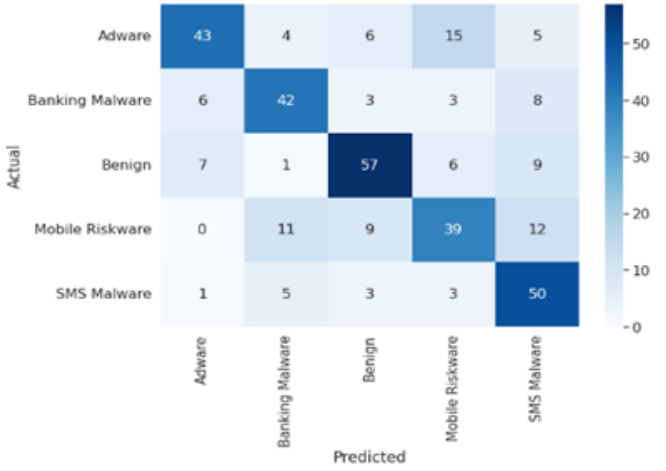

Figure 6. Subset_1 confusion matrix.

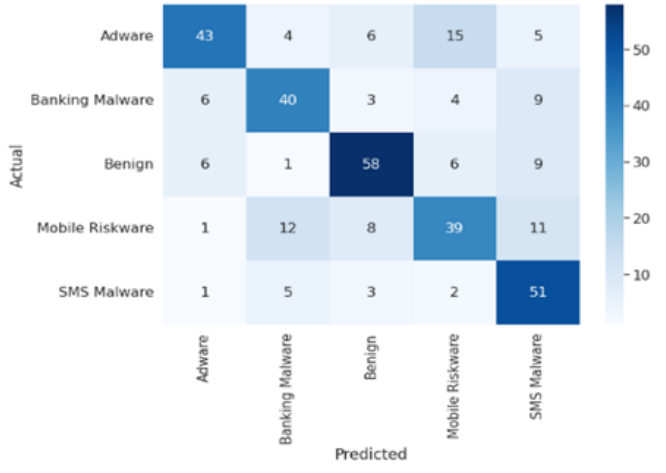

Figure 7. Subset_2 confusion matrix.

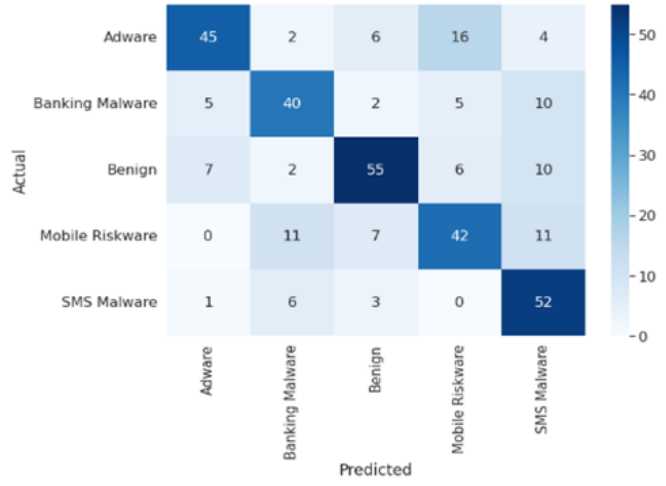

Figure 8. Subset_3 confusion matrix.

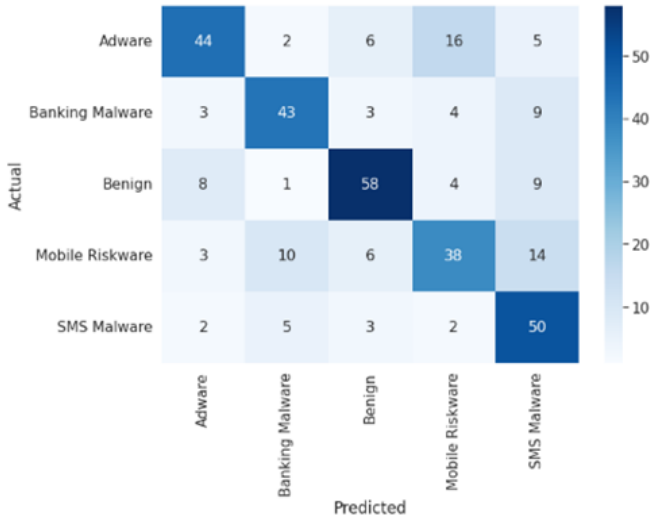

Figure 9. Subset_4 confusion matrix.

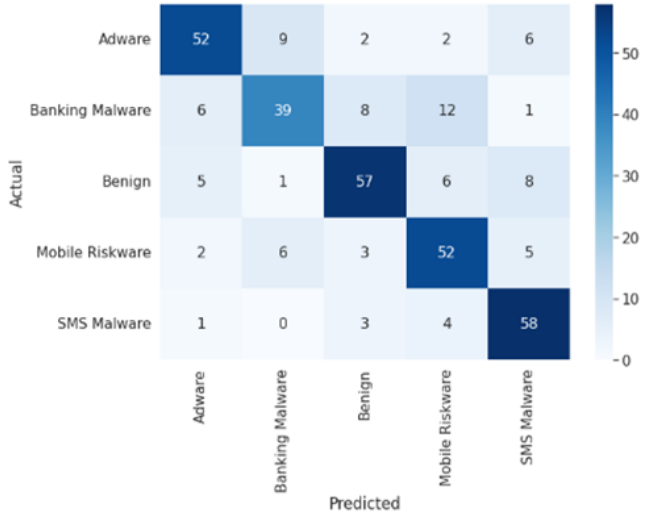

Figure 10. Subset_5 confusion matrix.

of the model on the 10 subsets are presented in Tables 12 – 21 respectively.

The splitting of each of the two datasets into 10 subsets is premised on the experimental observation that the performance of ANFIS models degrades as the size of the dataset increases. Figure 26 depicts the confusion matrix of the models' classification when a dataset of 60 instances is used for testing. Table 22 presents the corresponding performance metrics of the model on the smaller dataset. Comparing the results of this dataset size against the previous results of larger datasets (in Figures 6 – 25 and Tables 2 – 21) , this performance of ANFIS model is better on the smaller size dataset; where the performance of the model on smaller dataset yielded a maximum accuracy of 98%, the larger size dataset from CICMalDroid-2020 and CCCS-CIC-AndMal-2020 yielded a maximum accuracy of 92% and 94% respectively. The same pattern of result is seen in terms of precision and recall performance metrics: the

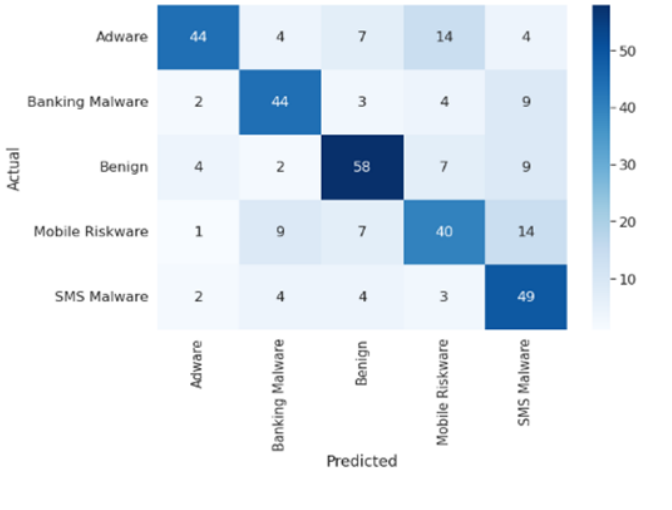

Figure 11. Subset_6 confusion matrix.

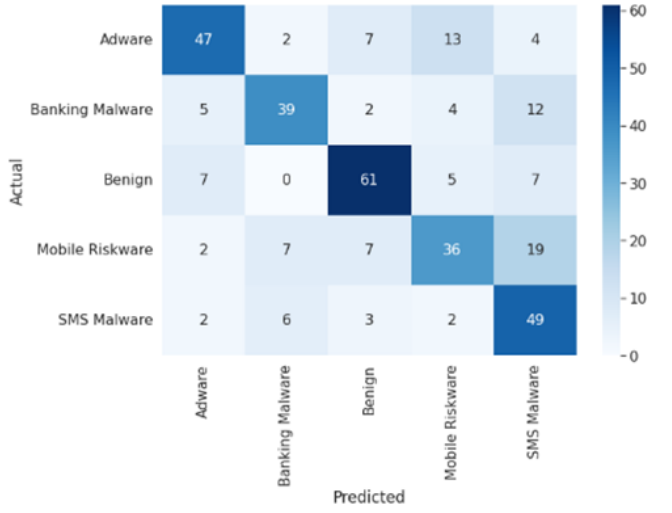

Figure 12. Subset_7 confusion matrix.

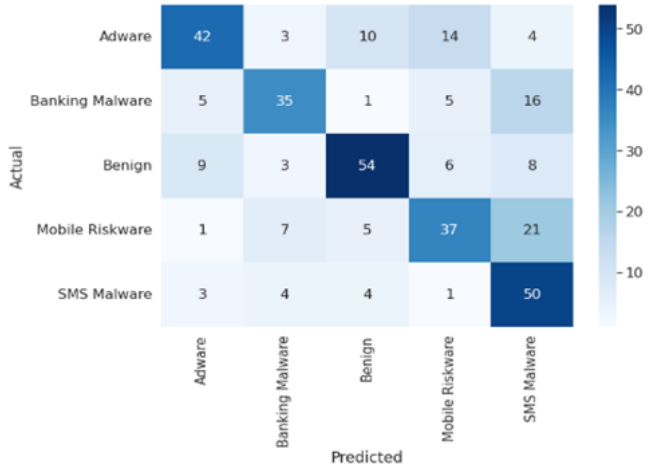

Figure 13. Subset_8 confusion matrix.

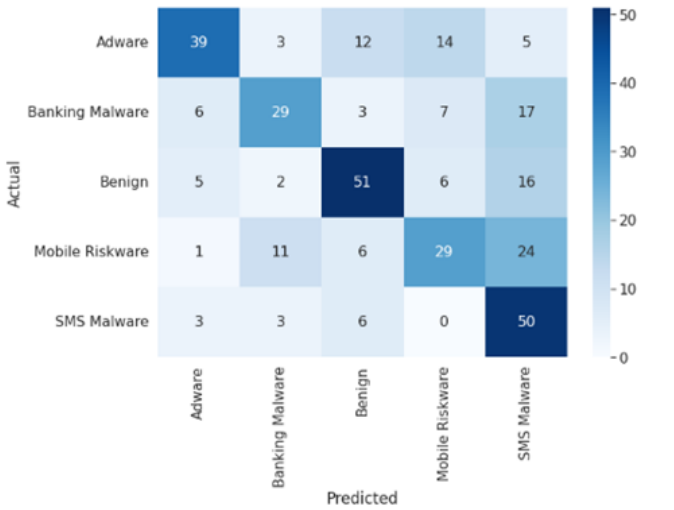

Figure 14. Subset_9 confusion matrix.

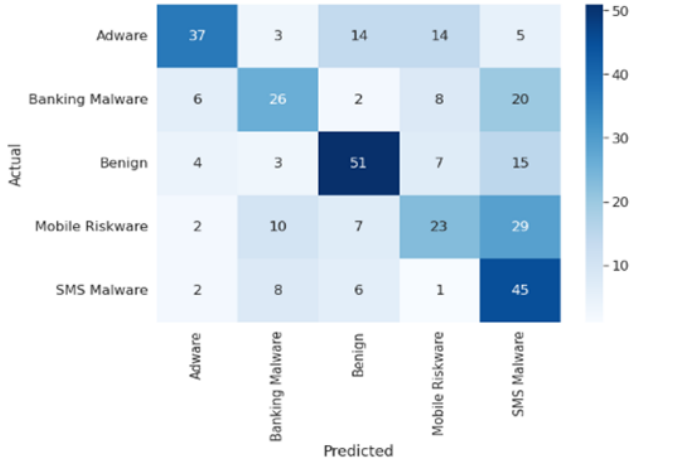

Figure 15. Subset_10 confusion matrix.

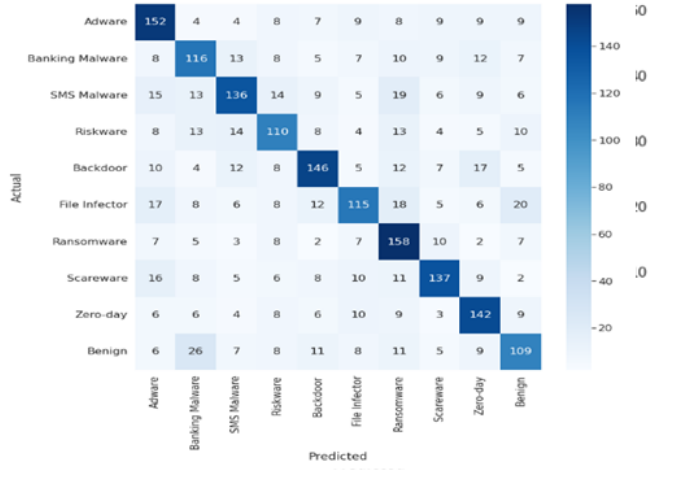

Figure 16. Subset_1 confusion matrix.

smaller dataset yielded a maximum precision and recall of 99%, CICMalDroid-2020 and CCCS-CIC-AndMal-2020 yielded a

Table 21. Subset_10 performance metrics.

	Accuracy	Precision	Recall	F1-Score
Adware	0.92	0.62	0.68	0.65
Banking Malware	0.91	0.54	0.54	0.54
SMS Malware	0.93	0.66	0.56	0.61
Riskware	0.92	0.57	0.53	0.55
Backdoor	0.93	0.67	0.62	0.65
File Infector	0.92	0.61	0.51	0.55
Ransomware	0.92	0.57	0.73	0.64
Scareware	0.92	0.64	0.63	0.63
Zero-Day	0.92	0.68	0.69	0.64
Benign	0.91	0.53	0.52	0.54

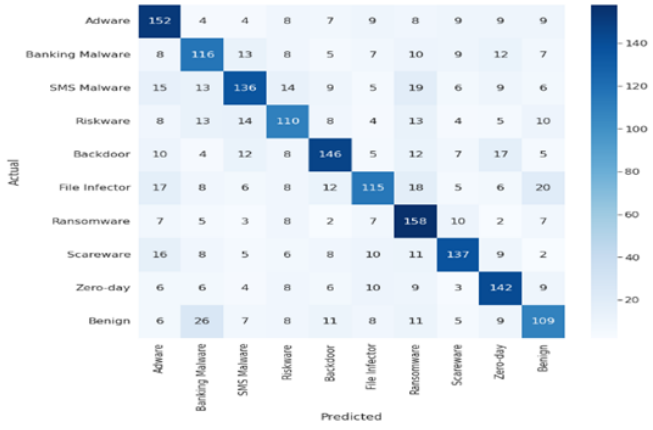

Figure 17. Subset_2 confusion matrix.

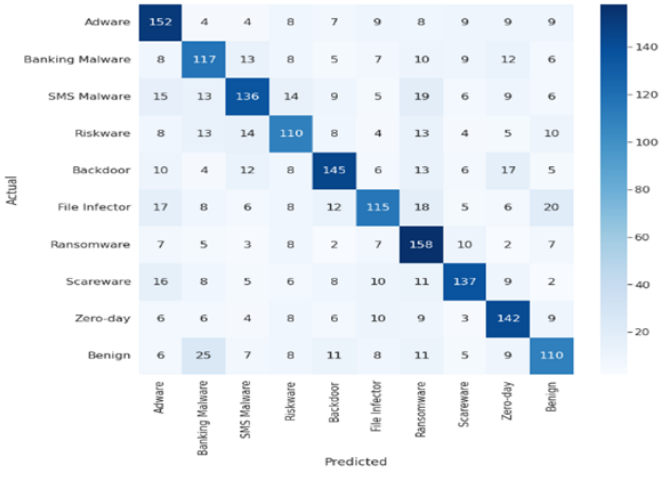

Figure 18. Subset_3 confusion matrix..

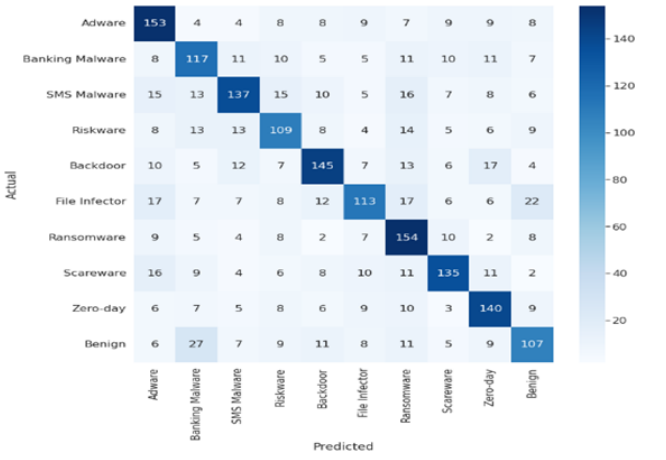

Figure 19. Subset_4 confusion matrix..

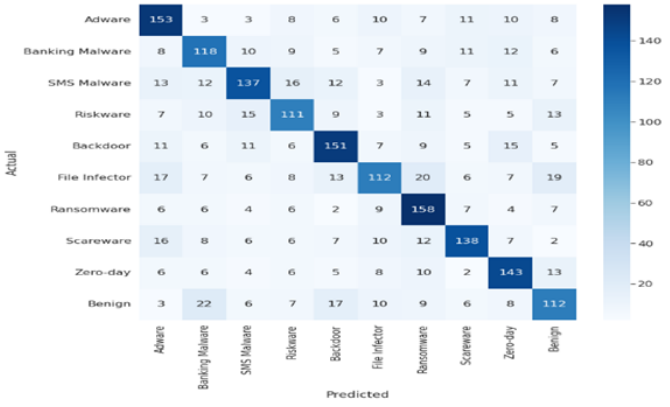

Figure 20. Subset_5 confusion matrix..

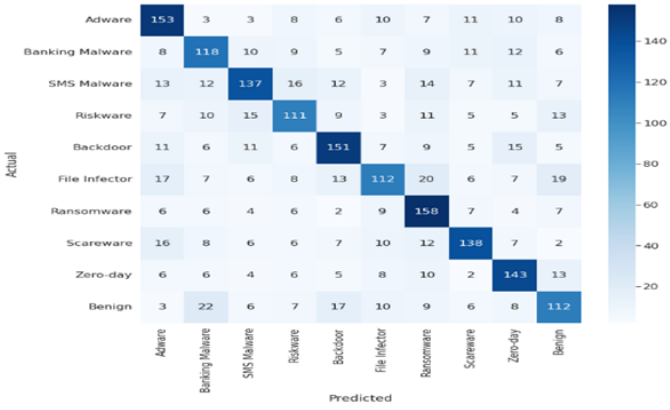

Figure 21. Subset_6 confusion matrix..

maximum precision of 83% and 71% respectively, and maximum recall of 88% and 76% respectively.

Thus, ANFIS model show good performance in the classification of malware families, its performace reduces as the size of the dataset used for testing increases.

6.5. Comparative analysis

This section presents a comparative analysis of the results of this study with some other studies reported in the literature. Studies that use ANFIS and ML models are compared. The ML

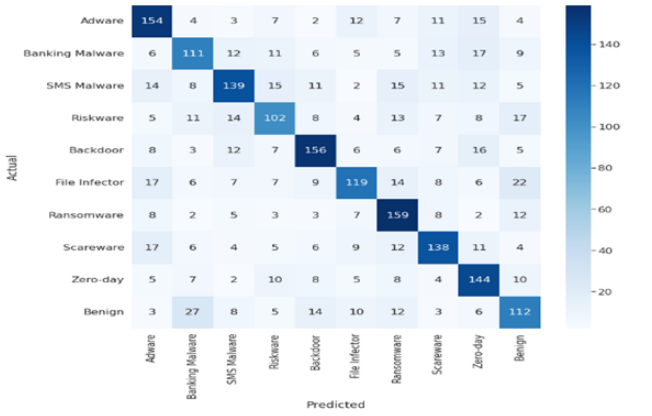

Figure 22. Subset_7 confusion matrix..

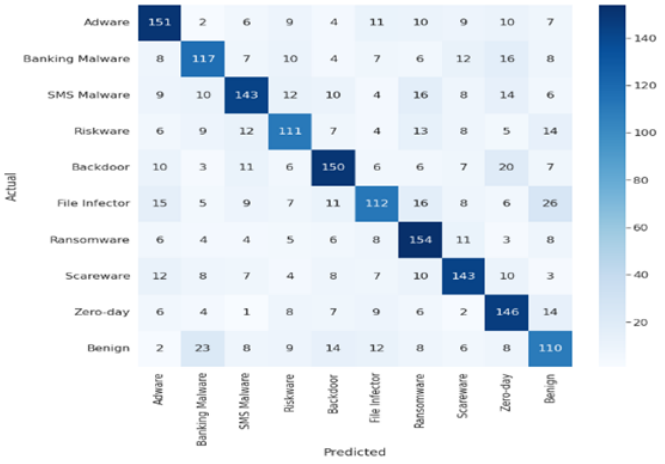

Figure 23. Subset_8 confusion matrix..

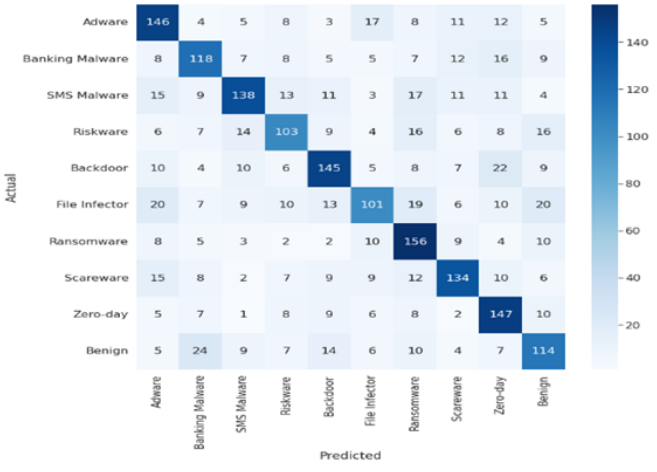

Figure 24. Subset_9 confusion matrix..

models are a select few in the literature, and they are to show how the ANFIS model performs vis-à-vis ML models [33]. Table 23 presents the performance comparison of accuracies irrespective of the malware family and datasets used in the studies.

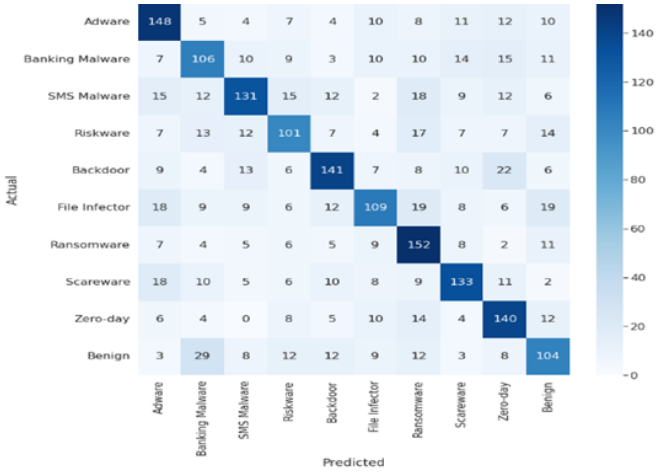

Figure 25. Subset_10 confusion matrix..

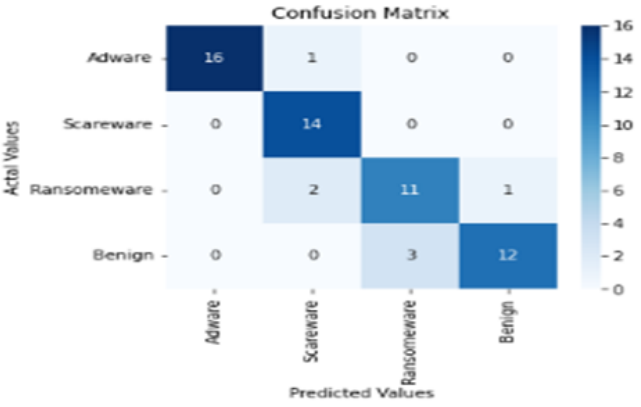

Figure 26. Confusion Matrix of the ANFIS prediction.

Table 22. Model's Prediction.

Metrics	Accuracy	Precision	Recall
Adware	0.98	0.94	0.99
Scareware	0.95	0.99	0.82
Ransomware	0.90	0.79	0.79
Benign	0.93	0.80	0.86

The results show that although the ANFIS model yields good results, it performs better with small datasets.

Table 23. Comparative analysis.

Study	ANFIS	RF	SVM	NB	DT	KNN	XGBoost	MLP	Gradient Boosting	Ensemble models
Damaševičius *et al.* [5]										98.9
Shhadat *et al.* [17]		95.8	88.6	81.8	92	88				
Sodiya *et al.* [10]	97.96	84.78	92.87							
Nugraha [11]		99.42							98.73	
Rabadi & Teo [12]		98.43	99.43		99.16		99.89			
This study	98^{+} 94*									

$^{+}$small dataset *large dataset

7. Conclusion

This study presented the application of an adaptive neuro-fuzzy inference system for malware detection and multinomial classification. Takagi' Sugano ANFIS model was implemented using MATLAB programming environment. The model uses five inputs and one output. 3 Gaussian membership function was used for each input. The hybrid optimization method was selected for training the ANFIS model on the Fuzzy Logic Toolbox of MATLAB [34]. The multinomial malware datasets used for training the model was collected from the Canada Institute of Cyber Security. With an RMSE of 0.19, the model showed a promising result of 98% accuracy in predicting the Adware malware family; 95% accuracy in predicting the Scareware malware family; 90% accuracy in predicting the Ransomware malware family; and 93% accuracy in predicting benign cases. However, due to the preliminary observation that the performance of the ANFIS model reduces as the size of the dataset increases, the two datasets from the Canadian institute were split into 10 subsets, and each was divided into training and testing sets using a 70:30 ratio. Although the accuracy marginally reduced, the precision and recall results were reduced in large-size datasets.

The limitation experienced in this study is the lack of a multinomial malware dataset for training the model. When this study was carried out, the available multinomial malware dataset was from the Canadian Institute of Cyber Security.

Further studies will consider other forms of the architectures of the neuro-fuzzy approach besides the ANFIS model considered in this study. Examples of such architecture include but are not limited to generalized approximate reasoning intelligent control (GARIC), fuzzy adaptive control network (Falcon), and neuro-fuzzy controller (NEFCON). Also, future works could consider more malware families especially those not considered in this study.

Data Availability

The two datasets used in this study are available at the Canadian Institute for Cybersecurity official websites:

1. CICMalDriod-2020: https://www.unb.ca/cic/datasets/maldroid-2020.html.
2. CCCS-CIC-AndMal-2020: https://www.unb.ca/cic/datasets/andmal2020.html.

References

[1] N. A. Azeez, O.E. Odufuwa, S. Misra, J. Oluranti & R. Damaševičius, "Windows PE malware detection using ensemble learning", Informatics **8** (2021) 10 https://doi.org/10.3390/informatics8010010.

[2] X. Gao, C. Hu, C. Shan, B. Liu, Z. Niu & H. Xie, "Malware classification for the cloud via semi-supervised transfer learning", Journal of Information Security and Applications **55** (2020) 102661. https://doi.org/10.1016/j.jisa.2020.102661.

[3] H. Faruk, H. Shahriar, M. Valero, F. B. Lamia, S. Shahria, K. Abdullah, W. Michael, C. Alfredo, L. Dan, R. Akond & W. Fan, "Malware detection and prevention using Artificial Intelligence techniques", IEEE International Conference on Big Data, 2021, Orlando, FL, USA, 2021, pp. 5369–5377. https://doi.org/10.1109/BigData52589.2021.9671434.

[4] D. Gibert, C. Mate & J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges", Journal of Network and Computer Applications **153** (2020) 102526. https://doi.org/10.1016/j.jnca.2019.102526.

[5] R. Damaševičius, A. Venčkauskas, J. Toldinas & S. Grigaliūnas,"Ensemble-based classification using neural networks and machine learning models for Windows PE malware detection", Electronics **10** (2021) 1. https://doi.org/10.3390/electronics10040485.

[6] S. Banin & G. O. Dyrkolbotn, "Multinomial malware classification vis low-level features", Digital Investigation **26** (2018) S107. https://doi.org/10.1016/j.diin.2018.04.019.

[7] B. Zhang & Y. C. Shin, "A data-driven approach of Takagi-Sugeno fuzzy control of unknown nonlinear systems", Applied Sciences **11** (2021) 1. https://doi.org/10.3390/app11010062.

[8] N. Asrafi, *Performance of malware classification on machine learning using feature selection*, Master of Science Thesis, Computer Science, Kennesaw State Universit, Kennesaw, 2020. https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?params=/context/cs_etd/article/1036/&path_info=Nusrat_Thesis_Final.pdf.

[9] H. C. Tanuwidjaja & K. Kim, "Enhancing malware detection by modified deep abstraction and weighted feature selection", Symposium on Cryptography and Information Security, Kochi, Japan, 2020, pp. 1–8. https://caislab.kaist.ac.kr/publication/paper_files/2020/scis2020_HR.pdf.

[10] A. S. Sodiya, O. J. Falana, S. A. Onashoga & B. S. Badmus, "Adaptive neuro-fuzzy system for malware detection", Journal of Computer Science **21** 2014 150044. https://ajol.info/index.php/jcsia/article/view/150044.

[11] U. Nugraha, "Malware classification using machine learning algorithm", Turkish Journal of Computer and Mathematics Education **12** (2021) 1834. https://turcomat.org/index.php/turkbilmat/article/view/3274.

[12] D. Rabadi & Sin G. Teo, "Advanced windows methods on malware detection and classification", Computer Security Applications Conference, Austin, USA, 2020, pp. 54–68. https://doi.org/10.1145/3427228.3427242.

[13] B. Khammas, "Malware detection using sub-signatures and machine learning technique", Journal of Information Security Research **9** (2018) 96. https://doi.org/10.6025/jisr/2018/9/3/96-106.

[14] T. Lu, Y. Du, L. Ouyang, Q. Chen & X. Wang,"Android malware detection based on a hybrid deep learning model", Security and Communication Networks **2020** (2020) n/a. https://doi.org/10.1155/2020/8863617.

[15] M. Yousefi-Azar, L. G. C. Hamey, V. Varadharajan & S. Chen, "Malytics: A malware detection scheme", IEEE Access, 6 (2018) 49418. https://doi.org/10.1109/ACCESS.2018.2864871.

[16] X. Liu, Q. Lei & K. Liu, "A graph-based feature generation approach in android", Engineering, **2020** (2020) n/a. https://doi.org/10.1155/2020/3842094.

[17] I. Shhadat, B. Bataineh, A. Hayajneh & Z. A. Al-Sharif, "The use of machine learning techniques to advance the detection and classification of unknown malware", Procedia Computer Science, **170** (2019) 917. https://doi.org/10.1016/j.procs.2020.03.110.

[18] M. K. Alzaylaee, S. Y. Yerima & S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices", Computers and Security **89** (2020) 101663. https://doi.org/10.1016/j.cose.2019.101663.

[19] A. Aiterher, A. Almomani & S. Ramadase, "Application of adaptive neuro-fuzzy inference system for information security", Journal of Computer Science **8** (2012) 983. https://thescipub.com/pdf/jcssp.2012.983.986.pdf.

[20] E. Eskandari & S. Hashemi, "A graph mining approach for detecting unknown malware", Journal of Visual Language and Computing **23** (2012) 154. http://dx.doi.org/10.1016/j.jvlc.2012.02.002.

[21] Jamuna A & S.E Vinoth Ewards, "Survey of traffic classification using ML", International Journal of Advanced Research in Computer Science **4** (2017) 65. https://www.ijarcs.info/index.php/Ijarcs/article/view/1598.

[22] P. Vinod, R. Jaipur, V. Laxmi & M. Gaur, "Survey on malware detection methods", Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security, Kanpur, 2009, pp. 74–79. https://dl.acm.org/doi/10.1109/IRI51335.2021.00033.

[23] S. Yoo, S. Kim, S. Kim & B. B. Kang,"AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification", Information Sciences **546** (2021) 420. https://doi.org/10.1016/j.ins.2020.08.082.

[24] E. D. O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin & F. C. Bernardini, "A model based on LSTM neural networks to identify five different types of malware", Procedia Computer Science **159** (2019) 182. https://doi.org/10.1016/j.procs.2019.09.173.

[25] M. Norouzi, A. Souri & M. Samad Zamini, "A data mining classification approach for behavioral malware detection", Journal of Computer Networks and Communications **2016** (2016) n/a. https://doi.org/10.1155/2016/8069672.

[26] L. Xiaofeng, Z. Xiao, J. Fangshuo, Y. Shengwei & S. Jing, "ASSCA: API based sequence and statistics features combined malware detection architecture", Procedia Computer Science **129** (2018) 248. https://doi.org/10.1016/j.procs.2018.03.072.

[27] M. N. Al-Andoli, S. C. Tan, K. S. Sim, C. P. Lim & P. Y. Goh, "Parallel deep learning with a hybrid BP-PSO framework for feature extraction and malware classification", Applied Soft Computing 131 (2022) 109756. https://doi.org/10.1016/j.asoc.2022.109756.

[28] J. M. Arif, M. F. Ab Razak, S. R. T. Mat, S. Awang, N. S. N. Ismail & A. Firdaus, "Android mobile malware detection using fuzzy AHP", Journal of Information Security and Applications **61** 102929. https://doi.org/10.1016/j.jisa.2021.102929.

[29] A. Djenna, A. Bouridane, S. Rubab, I. M. Marou, "Artificial Intelligence-based malware detection, analysis, and mitigation", Symmetry **15** (2023) 677. https://doi.org/10.3390/sym15030677.

[30] M. M. Masud, L. Khan & B. Thuraisingham, "A hybrid model to detect malicious executables", IEEE International Conference on Communications", Glasgow Scotland, 2007, pp. 1443–1448. https://ieeexplore.ieee.org/xpl/conhome/4288670/proceeding.

[31] S. Chavan, K. Shah, N. K. Dave, S. Mukherjee, A. Abraham & S. Sanyal, "Adaptive neuro-fuzzy intrusion detection systems", International Conference on Information Technology: Coding and Computing, Las Vegas, Navada, 2004, pp. 70-74. http://dx.doi.org/10.1109/ITCC.2004.1286428.

[32] T. Mitiku & M. S. Manshahia, "Neuro fuzzy inference approach: A Survey", International Journal of Scientific Research in Science, Engineering and Technology **4** (2018) 505. https://ijsrset.com/home/issue/view/article.php?id=IJSRSET184831.

[33] Y. Fang, Y. Liu, C. Huang & L. Liu, "Fastembed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm", PLoS ONE **15** (2020) 1. https://doi.org/10.1371/journal.pone.0228439.

[34] S. G. Tzafestas, "Fuzzy logic and neural network handbook", Journal of Intelligent & Robotic Systems **28** (2000) 293. http://dx.doi.org/10.1023/A:1008175521315.

[35] M. G. Schultz, E. Eskin & F. Zadok, "Data mining methods for detection of new malicious executables", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2001, pp. 38–49. https://doi.org/10.1109/SECPRI.2001.924286.