Intelligent Malware Detection Using a Neural Network Ensemble Based on a Hybrid Search Mechanism

Stephen M. Akandwanaho
Richfield Graduate Institute of Technology, South Africa
https://orcid.org/0000-0002-1520-1847

Muni Kooblal
Richfield Graduate Institute of Technology, South Africa
https://orcid.org/0000-0001-5145-9071

Abstract
Malware threats have become increasingly dynamic and complex, and, accordingly, artificial intelligence techniques have become the focal point for cybersecurity, as they are viewed as being more suited to tackling modern malware incidents. Specifically, neural networks, with their strong generalisation performance capability, are able to address a wide range of cyber threats. This article outlines the development and testing of a neural network ensemble approach to malware detection, based on a hybrid search mechanism. In this mechanism, the optimising of individual networks is done by an adaptive memetic algorithm with tabu search, which is also used to improve hidden neurons and weights of neural networks. The adaptive memetic algorithm combines global and local search optimisation techniques in order to overcome premature convergence and obtain an optimal search outcome. The results from the testing prove that the proposed method is strongly adaptive and efficient in its detection of a range of malware threats, and that it generates better results than other existing methods.

Keywords
cybersecurity, malware, malware detection, artificial intelligence, neural networks, neural network ensemble, memetic algorithm, optimisation, hybrid search, tabu search

Acknowledgement
This article draws on the contents of a conference paper presented at THREAT 2019: Enhancing Africa's Cybersecurity Posture, 26-27 June 2019, Johannesburg, https://www.threatcon.co.za/2019.html

DOI: https://doi.org/10.23962/10539/28660

Recommended citation
Akandwanaho, S. M. , & Kooblal, M. (2019). Intelligent malware detection using a neural network ensemble based on a hybrid search mechanism. *The African Journal of Information and Communication (AJIC),* 24, 1-21.
https://doi.org/10.23962/10539/28660

1. Introduction

Malware attacks have increased recently, in Africa and globally, due to advances in technology and the growing number of miscellaneous internet of things (IoT) devices being connected to data networks (Xiao, Lin, Sun & Ma, 2019). The nature of malware attacks has also dramatically changed, as sophisticated attacks have become ubiquitous. The sophistication and complexity of malware have manifested in miscellaneous ways; the common way has been malware camouflage and obfuscation, where the attack comes in the form of a solution to a problem together with a demand for ransom money (Kalaimannan, John, DuBose & Pinto, 2017). This type of attack has continued to evolve and to produce more variants, so that the perpetrators can continue to profit from such pernicious attacks.

According to Symantec, malware has increased sharply since 2014 (Lee & Kwak, 2016) and continues to increase. Symantec also reports that the majority of the new malware programs are variants of the existing destructive malware, which is indicative of the evolution that is taking place in order for the programs to be more complex for the countermeasures and to avoid detection.

In order to mitigate malware, a number of countermeasures have been advanced in the literature (Jerlin & Marimuthu, 2018). However, existing techniques have not fared well due to the obfuscation tactics of malicious software and other advances in evading detection. Malware targets have also expanded to include mobile platforms, thereby posing another challenge to existing mitigation efforts.

Most existing research has reported on the efficiency of machine learning (ML) and artificial intelligence (AI) techniques in malware detection and mitigation (Chen, Su & Qiao, 2018). The techniques are useful in the classification of malware, such as Trojans, worms, among others, and in mapping suitable techniques to the malware type. In addition, the ML techniques, through feature extraction, increase the accuracy of detecting malware by reducing the search space, so as to home in on the specific malware (Khammas, 2018). This alleviates some of the current challenges, including the conventional detection methods being evaded by new variants of malware due to

search limitations. This can be attributed to the learning abilities of ML, as well as their capacity to mine data patterns, relationships and procedure analysis.

The modern malware is increasingly adaptive and dynamic in nature, which makes self learning techniques important. Specifically, the self learning techniques, such as neural networks, are able to self-organise and self-evolve, as well as classify and process data in parallel, and are hence able to detect mutated and other instrinsic forms of malware (Barriga & Yoo, 2017).

Agent-based methods have also become promising approaches for malware detection in both web and mobile applications (Kendrick, Criado, Hussain & Randles, 2018). This is due to the interactions between agents and their environments, which create more focused and accurate inputs, so as to generate robust intelligent solutions against malware. (Agents are any entities that can make decisions like a human being through their interactions with one another and the environment.) Moreover, the heterogeneity of agents and of their attributes enriches the capabilities of agent-based applications in combating different types of malware. Through these interactions and the inherent data collection and storage capabilities of agents, patterns can be inferred, which are useful for predictions.

Prediction has been extensively explored in the domain of malware detection, especially using machine learning techniques to predict the behaviour of malware and its hotspots or risk areas (Mahrin et al., 2018). The behavioural analysis of malware also includes classification, which is essential for investigating and prioritising threats. The recent malware attacks have become much more coordinated, for example, botnets which represent a string of devices that are interconnected to communicate and share information with one another, so as to launch large-scale and high-level attacks. Botnets use autonomous programs known as bots, which mimic human behaviour in interactions with users with a view to collecting information and using it to conduct various kinds of malicious attacks (Khoshhalpour & Shahriari, 2018). The level of complexity inherent in these kinds of attacks continues to pose a significant challenge to the traditional malware detection techniques, as well as to the common network environments.

The threat of malware has spread to mobile telephony platforms and proliferated exponentially on account of the openness, and popularity of use, of mobile platforms (Ren, Liu, Cheng, Guo & Chen, 2018). These mobile platforms have become carriers of very sensistive data, ranging from personal financial information to the private details of users' lives. Any data breach on these platforms due to malware attacks can have severe consequences.

The web and desktop platforms nowadays also carry a similar risk, due to a proliferation of desktop applications that enable users to process sensitive information, and also

due to mixed storage and processing of business and personal information. One of the biggest attacks in modern times was the WannaCry ransomware attack in 2017, which affected more than 150 countries in less than a week. More than 200,000 devices were attacked in a matter of days, through forcing the encryption of users' data until a ransom was paid (Wang et al., 2018). This is one of the many examples of malware that use advanced algorithms to conduct large-scale malicious attacks. The existential challenge posed by these techniques is their ability to evade detection and literally throw the traditional and less intelligent detection methods into confusion.

In this article, we advance a novel approach, with a view to outflanking the intelligent malware and providing a robust countermeasure to a wide variety of malware. The approach is based on a *neural network ensemble*, wherein an intelligent search optimisation process is conducted by memetic algorithm and by the k-means machine-learning clustering algorithm, in order to generate the optimal solution for complex malware detection. The approach that we advance applies to cyber systems throughout the world. Malware attacks in Africa are similar to attacks elsewhere, and thus we have designed the algorithm with the aim of helping to solve a global problem.

The rest of the article is organised as follows: section 2 provides a review of related work, section 3 provides the problem definition, section 4 provides the methodology, section 5 provides the discussion and presentation of experimental results, and section 6 concludes.

2. Related work

Neural-network-based malware detection

Due to the dynamic behaviour and increasing obfuscation tactics of malware, more intelligent solutions have been sought, and prominent among them are neural networks. Many factors make neural networks attractive for solving problems of this nature, but the overriding factor is that, due to their intrinsic training processes, they achieve accuracy and efficacy in solving very complex problems (Hassan & Hamada, 2017), thus making them suitable for malware detection.

Yan, Qi and Rao (2018) present an *ensemble* method for detecting malware based on a deep neural network. The approach uses a *convolutional* neural network and a memory technique to learn raw data and make inferences regarding the existence or nonexistence of malware. The inferences are based on patterns extrapolated from both the structure and code of the malicious file. (A convolutional recurrent neural network is a blend of the recurrent neural network and the convolutional neural network. Convolutional neural networks can be characterised as those that apply convolutions (a kind of mathematical operation) and that classify data regardless of the positioning.) This approach is similar to the *recurrent neural network ensemble* proposed by Rhode, Burnap and Jones (2018). The ensemble studies behavioural data

and makes inferences regarding the maliciousness of an executable file. This is done during the execution by collecting a small sample of behavioural data with a view to detecting and blocking malicious processes before they cause damage. In order to classify this behavioural data, a classifier is presented based on a convolutional recurrent neural network (Alsulami & Mancoridis, 2018), in order to classify families of malware and to extrapolate better patterns for improving detection acurracy. The method extracts features adaptively from MS Windows files to classify them.

In the same vein, Kabanga and Kim (2018) apply the convolutional neural network to the classification of malware *image*. Instead of using text and other forms of data as inputs, image vectors are used to train neural networks. The convolutional neural network is set up with three layers, in order to achieve the classification function.
In order to identify and classify complex patterns in data for malware detection, Le, Boydell, Namee and Scanlon (2018) present a classification method based on deep learning. The approach uses data-driven techniques to identify features for classification. Multiple deep-learning architectures are utilised, and each input is classified into a malware class in terms of various neural network layers, whereby vectors are generated for feature extraction and classification. This classification method contrasts with mechanisms that rely on expert domain knowledge. For example, Zarras, Webster and Eckert (2016) propose a malware classification strategy based on both recurrent and convolutional neural networks. System calls are obtained and sequenced using a sequential model to form a domain for feature extraction and classification. The same principle is applied by Martinelli, Marulli and Mercaldo (2017), wherein a dynamic analysis is conducted on system calls and a convolutional neural network is deployed to distinguish malicious data from benign data in an Android data sample. The recurrent neural networks, unlike convolutional neural networks, recall input samples and reuse them for classification of the current samples.

A dynamic malware detection technique based on deep learning is also presented by Yin, Zhou, Wang, Jin and Xu (2018), wherein malware execution and monitoring functions are separated and analysed independently. A log is produced for the monitoring processes where information is then extracted as the input for the neural network. The training enables the neural network to recognise and classify various types of malware. The approach of Selvaganapathy, Nivaashini and Natarajan (2018) uses a restricted Boltzmann machine (RBM) with a stacking technique to select features in a neural network and detect malicious patterns in uniform resource locators (URLs). A miscellany of classes are used for classification. A similar problem is solved by Le, Pham, Sahoo and Hoi (2017), with convolutional neural networks and the detection technique embedded in the URL so as to train the neural network in all aspects of the URL, including words and characters. An RBM is an artificial neural network that is stochastic (i.e., randomly determined) in nature, and a probability distribution can be generated from its inputs.

In order to detect and classify malware in unseen files, Rad, Nejad and Shahpasand (2018) apply a binary classifier to MS Windows files. The training of the neural network classifier is done with a view to giving it the ability to distinguish malicious files from benign ones.

Many of the aforementioned approaches provide novel solutions, but the malware landscape has dramatically changed in recent years. The changes are mostly epitomised in the shifting optima (Souri & Hosseini, 2018), i.e., shifts in the most favourable solution among a set of constantly changing feasible solutions. The shifting optima makes tracking difficult, and makes it overwhelmingly difficult for static and slow-evolving solution approaches to find optima. Thus, the more efficient malware countermeasures will be those that are highly adaptive and dynamic in their search and optimisation processes for malware detection.

Memetic algorithm solutions

As malware has become more disguised and sophisticated, search heuristics have come to be considered effective optimisation mechanisms, not only for detecting malware threats but also for finding optimal solutions. However, the challenge for single search heuristics is that they get stuck in local optima in the course of the search (Xu et al., 2017), which undercuts the quality of the search outcome. Thus, hybrid mechanisms, such as memetic algorithms, are preferred metaheuristics for conducting the optimisation search process, especially in non-stationary malware environments.

Okobah and Ojugo (2018) present a memetic model for malware intrusion detection. The approach uses classification rules, as well as a fitness function based on the evolutionary process, to generate a feasible solution. Mohammadi and Namadchian (2017) apply a memetic algorithm to optimise detection of irregular traffic. In order to classify malicious traffic, a memetic evolutionary classifier is used. The classifier functions in diverse and dynamic environments. This approach draws parallels with the hybrid mechanism proposed by Xue, Jia, Zhao and Pang (2018), where the feature selection is done by differential evolution, and neighbourhood improvements are conducted by the k-nearest algorithm, with a view to averting premature convergence. Shah, Ehsan, Ishaq, Ali and Farooq (2018) present a hybrid classifier to classify irregular activity. A genetic algorithm and a support vector machine are deployed for, respectively, feature selection and optimising of parameters to enhance accuracy. The training resources required, with respect to time, are significantly reduced through faster covergence, provided a robust and optimal feature selection process exists.

Dash (2017) applies a hybrid of particle swarm optimisation (PSO) and gravitational search to detect intrusions and malicious activity (Dash, 2017). In the same vein, Altaher and Barukab (2017) combine the PSO with the adaptive neural fuzzy inference system in order to distinguish malware-infected Android applications

from malware-free applications. Fuzzy rules are generated to guide the classification process, through intelligent optimisation of parameters, using PSO search based on the evolutionary process that Razak et al. (2018) apply to solve a similar problem.

In the work by Zhirou and Jing (2018), the malicious attack vector is presented as a weighted network, and a memetic algorithm is used to optimise the cost on each node with a view to minimising the cost of attacking a node. An optimal search yields a node combination with the lowest cost.

Although all the approaches outlined above represent state-of-the-art advances in the area, the malware problem landscape continually evolves, at a exponential speed, as a result of the interconnectedness of systems (WEF, 2018) in the fourth industrial revolution (4IR), which demands more intelligent and adaptive solutions provided by advanced algorithms to counter the increasingly complex attacks.

3. Problem definition

The malware detection problem we addressed with this research is a *combinatorial optimisation* problem, where there is a finite set of multiple feasible solutions and the aim is to optimise and generate the best solution (Schweidtmann & Mitsos, 2019). The neural network ensemble we developed represents a blend of various neural networks whose functions are synergised with a view to minimising error and achieving precision (Yan et al., 2018). Neural networks are trained to build a strong capability for solving specific problems. Once they are trained, neural networks generate invidual outcomes which are combined to form an ensemble solution outcome. Ensemble approaches aim to offset the drawbacks associated with individual networks, and they present robust solutions to complex problems.

Accordingly, the problem we seek to address with our proposed model is two-fold: (1) optimisation of a neural network ensemble, and (2) malware detection. The mathematical formulation of the problem is as follows:

Notation	Denotation
C_l	class label
C_m	class of malware
C_b	class of benign
B_i	behaviour
Δ	dataset

$$\Delta = (v_1, v_2, .., v_n) \quad (1)$$

where $v_1, v_2 v_1, v_2$ are feature vectors in the dataset. The behaviour of the data is then represented as $B_{v_n} B_{v_n}$ and the directed graph, which represents the data relationships and dependencies in the form of weights

in a neural network (Xiao et al., 2019), is denoted by $G_n G_n$, where nn represents the retrieved number of graphs from the dataset. The behaviour of the sample for a class of malware is defined as follows:

$$B_{S_\Delta} = B_{v_n} C_m \tag{2}$$

$$B_{S_\Delta} = B_{v_n} C_b \tag{3}$$

where $B_{S_\Delta} B_{S_\Delta}$ is the behaviour of the sample, $S,S,$ in the dataset.

Therefore, the behaviour of the dataset is defined as:

$$BS_{G_n} = B_{v_n} C_m + B_{v_n} C_b \tag{4}$$

$$BS_{G_n} = B_{v_n} (C_m + C_b) \tag{5}$$

$B_{v_n} = \{0,1\}\, B_{v_n} = \{0,1\}$ for malware or benign, so that if the sample contains 1 then data is malware and if it contains 0 then data is benign.

This helps to calculate the detection error from the fitness function (Sheng et al., 2017), as follows.

$$F = \beta_1 x\, t_e + \beta_2 x\, n_c + \beta_3 x\, d_t \tag{6}$$

$$t_e = \sum_{j=1}^{P} \sum_{i=1}^{N} (y_i - z_i)^2 \tag{7}$$

$$n_c = \frac{AC}{TC} \tag{8}$$

$$d_t = \frac{1 - B_{V_n} C_m}{B_{V_n} (C_m + C_b)} \tag{9}$$

where $t_e t_e$, $n_c\, n_c$ and $d_t d_t$ denote, respectively, the training error, complexity of the neural network, and detection error. The target output, $y_i y_i$ and current network output, $z_i z_i$ are used to compute the error generated from training the neural network, as shown in Equation (7). The parameters are defined by $\beta_1 \beta_1$, $\beta_1 \beta_1$ and $\beta_1 \beta_1$ while active connections with weights, total connections including those on hidden neurons, training patterns, and neurons, are represented, respectively, by $ACAC, TC$ TC, PP and NN. The problem therefore is to minimise the function FF in Equation (6), which ultimately reduces training and detection errors, which in turn increases the detection rate of malware behaviour patterns in the data sample.

The second part of the problem is the optimisation of a neural network ensemble based on connection weights. As mentioned before, the neural network ensemble represents an amalgam of disparate networks (Choi &

Lee, 2018), where each individual network generates an output pattern from training input data. The network connections are defined by weights, which indicate the strength of these connections and of the influence of neurons on each other (Ojha, Abraham & Snasel, 2017) through synapses. The influence of neurons is determined by the strength level of the connections between them. The synapses generate an output, as a product of weight and input. Let ww represent weight of the neural network connection, such that $w = w_1, w_2, \dots, w_N w = w_1, w_2, \dots, w_N$ and pp represents inputs. In order to adjust and map outputs and inputs to the neuron, a bias operator, bb is utilised. This is expressed as follows.

$$F = A.\sum_{i=1}^{N} w_i\, p_i + b \tag{10}$$

where all connections, $N, N,$ use the activation function, AA, which activates neurons based on the weight, thus creating a bias to the process with a view to achieving a nonlinear output pattern, as demonstrated by Abiodun et al. (2018). The problem therefore is to optimise the weights in Equation (10) for all connections using a learning optimisation algorithm, so as to create a convergence of high-quality networks to the ensemble.

4. Methodology

Memetic algorithm

Memetic algorithm is one of the methods we used in this study to tackle the optimisation problem defined in the previous section. Memetic algorithm is a blend of global and local search heuristics that combines exploration and exploitation search processes in order to generate high-quality solutions (Nguyen & Sudholt, 2018). The main differentiating factor between memetic algorithm and genetic algorithm is that memetic algorithm mimics the evolution of the cultural environment, rather than mimicking the natural evolution of genes. This capability enables memetic algorithm to lend itself to malware detection, since malware, seeking to avoid detection, is capricious in its attributes and its interactions with the environment (Acarali, Rajarajan, Komninos & Zarpelao, 2019).

The memetic algorithm exploitation function is executed by the local search technique, with a view to forestalling premature convergence of the search process, and thus yielding an optimal result. The local search scours the neighbourhood regions of the solution for a better solution, based on fitness values for mutated solutions. Due to this strength and other benefits, memetic algorithm has been extensively applied to solve various real-world complex problems (Chaimanee & Supithak, 2018) that have become increasingly difficult to deal with through single and non-dynamic optimisation techniques. As indicated by Gu et al. (2019), hybrid algorithms provide a good balance between diversification and intensification of the search, enhancing the quality of the search process as well as the outcome.

In the model we developed and tested, adaptive mutation and recombination operators are applied to the memetic algorithm, in order to create better *offspring* configurations, which are essentially hybrid solutions composed of two existing *parent* configurations. The features from both parent chromosomes (i.e., individual solutions in the sample) are extracted and combined using the recombination operator through individual interactions and cooperation. The mutation operator is then applied to build new features, with a view to forming more robust solutions (Bereta, 2019). The mutation operator helps in calibrating the diversification levels of the population sample, e.g., if there is low diversification, new features are injected by increasing mutation. Figure 1 illustrates this memetic algorithm procedure.

Figure 1: Algorithm 1 (memetic algorithm procedure)

```
begin
P = Intitialise Population
Evaluate fitness of individuals
repeat
      Select parents
      Apply Operators
      Apply Local Search
      Evaluate fitness
         if offpsring is better than the existing
individual
           Replace existing solution with offspring
      or else
            keep existing solution
      end if
      Adjust mutation operator
      Update population
until termination criteria reached
return optimal solution
end
```

The population is randomly constructed in Algorithm 1, and one of the essential features of memetic algorithm is application of the local search technique. The local search procedure implemented in this work is the tabu search metaheuristic, and one of the prominent hallmarks of tabu search is its use of the adaptive memory function to store solutions as the search progresses through various iterations. This is important because it makes the information readily available for decision-making at any point in the course of the search (Lucay, Galvez & Cisternas, 2019). The search can then be strategically directed to promising areas where optimal solutions are most likely to be found, based on search information collected by tabu search. Figure 2 illustrates this tabu search procedure.

Figure 2: Algorithm 2 (tabu search procedure)

```
begin
Choose a solution, ii in sample, PP
while stop criterion is not reached do
Perform improvement search
Create a subset s*s* of solutions in PP
Add visited solutions to tabu, t
Evaluate fitness of individuals
Choose best, jj in s*s*
if f(j) > f(i)f(j) > f(i) then
Replace ii with jj in PP
end if
Update tabu list
Output optimal solution
end
```

In the tabu search algorithm, solution cycling is prevented via continual improvements until there is attainment of local optimality. The tabu search procedure in Algorithm 2 depicts process steps for local exploitation of the search space with a view to forestalling premature convergence. The procedure exemplifies dynamic local search optimisation, where continual update of the tabu list is conducted. This lends itself to malware detection and search environments, especially in today's environments where malware evolves rapidly. The application of tabu search in optimisation is guided by *weightage* to determine the penalty severity of constraint violations (Dai, Cheng & Guo, 2018). Hard constraints, which are constraints that must be satisfied and applied, carry high weights, and thus large penalties in the case of violations, while soft constraints carry lower weights and smaller penalties.

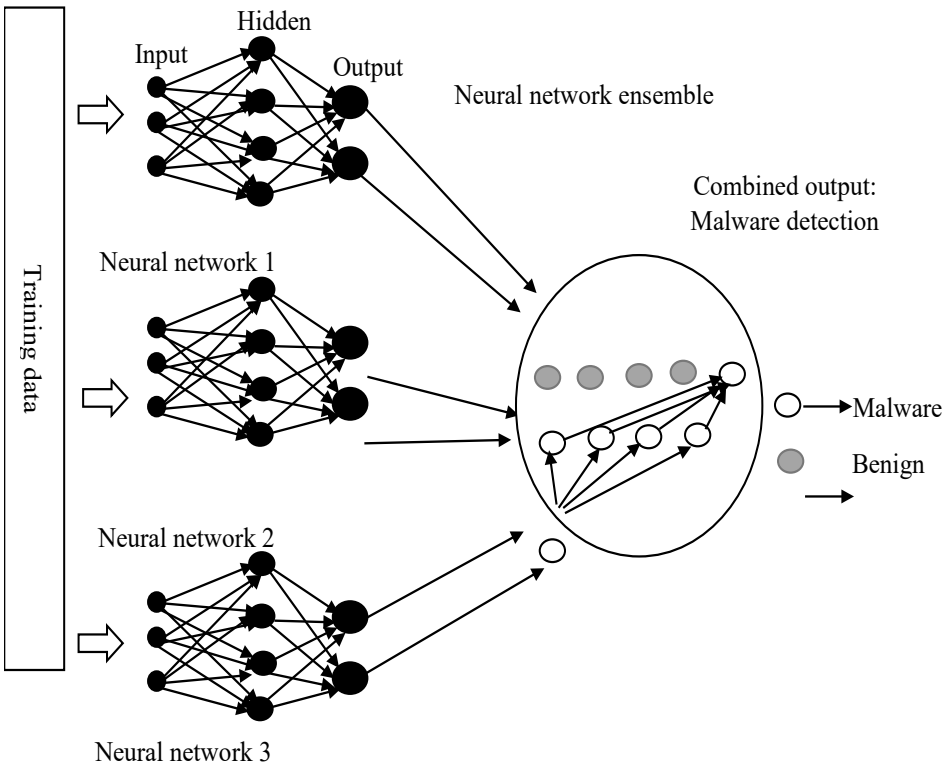
Neural network ensemble

A neural network is a data-based network that maps inputs to output patterns, or processes inputs to generate output through training processes. The neural network architecture mimics the natural functioning of the brain, in which billions of interconnected neurons transmit signals to each other to generate activity or action for the various functions of the body (Shapshak, 2018). In order to apply this natural phenomenon to computing, artificial neural networks are designed to simulate the neuron structure and processes of the brain, with a view to creating cognitively intelligent computer systems that can be deployed to solve complex problems (Jat, Dhaka & Limbo, 2018).

Neural networks go through training processes to enable them to learn and develop the required capability and mastery to solve various problems. The learning of the neural network is based on the way information travels through networks, as

propagated by neurons. The neurons are connected to one other and assigned weight values to indicate the importance and value of each connection. The neural network observes and learns the information flow, and influence, of the neurons. This can be achieved by *feed-forward*, where learning is performed in a forward sequential format from inputs to outputs, or by *back-propagation*, where the optimisation procedure is reversed, starting from the actual outputs and comparing them with the expected output, so as to adjust the connection weights with a view to decreasing the error. Figure 3 illustrates the functioning of a neural network ensemble.

Figure 3: Neural network ensemble, and its application to malware detection



A neural network ensemble represents a blend of individual neural networks that is aimed at combining various models to generate an environment with strong

generalised capacity and minimal error (Li et al., 2018). The mathematical formulation of the neural network that we developed and tested is as follows:

From Figure 3, let AA, BB and CC represent neural networks 1, 2 and 3, such that $A = (a_1, a_2, \ldots, a_n) A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_n)$ $B = (b_1, b_2, \ldots, b_n)$ and $C = (c_1, c_2, \ldots, c_n) . C = (c_1, c_2, \ldots, c_n)$.

Let ww denote the weights and DD be the network that represents the combined output of neural networks, A, BA, B and CC. The activation function is used to map inputs to output based on the weights of connections for inputs as well as biases between neurons (Eger, Youssef & Gurevych, 2018), such that $f = (A + B + C).w \rightarrow Df = (A + B + C).w \rightarrow D$. The neural network ensemble in Figure 3 is therefore presented as follows.

$$D = \delta \sum_{i=1}^{n} w * (A + B + C) + b \tag{11}$$

where $\delta\delta$, ww and bb are the activation function, the weight and bias vectors respectively. The bias factor helps to influence the outcome of the neural network, as well as its behaviour by determining the triggering value of the activation function, $\delta\delta$, hence acting as an anchor to the network.

This implies that with this additional parameter, the behaviour of the neural network can be adjusted with a view to achieve optimal learning and performance. In order to get an optimal neural network ensemble, the optimisation of the individual neural networks is vital (Ju, Bibaut & Van der Laan, 2018) and to this end, a memetic algorithm is deployed in this work to optimise the search process by exploitation and exploration of the search space so as to generate high quality trained neural networks that can compose a robust ensemble network.

5. Experiments

The experiments are conducted using the Intel Core i3-4005U @1.70GHz(4 CPUs), 8GB RAM, 64-bit Operating System. R-programming and MATLAB environments are used for the neural network ensemble implementations and analysis. The neuralnet Library in the R platform is utilised to train the neural networks. The environments are also used to perform memetic optimisation, where global search, as well as local search improvements, are done using genetic and tabu search algorithms respectively. A stacking approach is used to combine classifiers, so as to synthesise estimate outputs from various neural networks and achieve high levels of accuracy (Ma, Wang, Gao, Wang & Khalighi, 2018). A single outcome is then produced for the neural network ensemble. The optimisation threshold in the neural net is set at

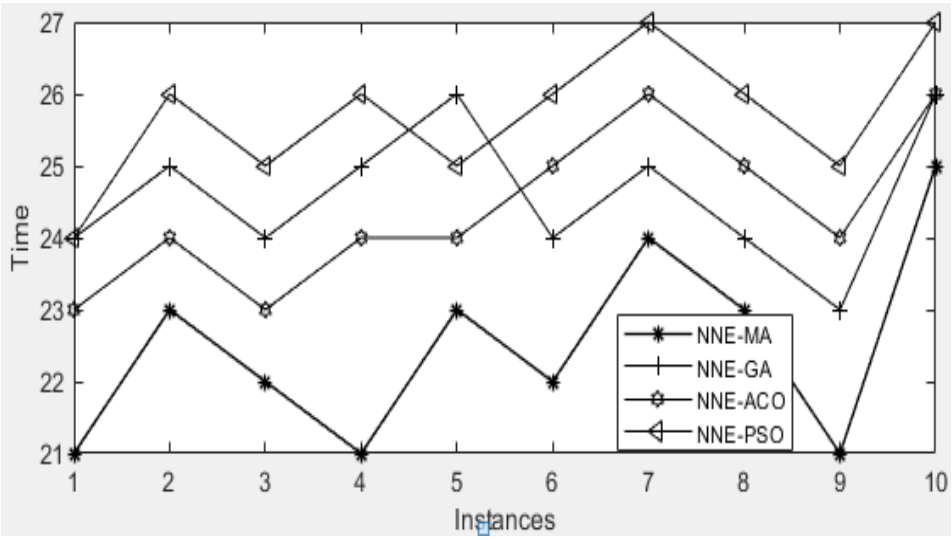
0.1, such that if the $error < 0.1error < 0.1$, the optimisation will automatically stop.

The datasets used are obtained from the Center for Machine Learning and Intelligent Systems (2016). The training datasets are multivariate, with both malicious and benign features labelled as -1 and +1 respectively. Features are extracted, and data is divided into *testing* and *training* sets. The algorithm is first trained on the training dataset before conducting actual tests, so as to be able to detect malware. The mathematical formulation for the training is as follows:

> Let $\sigma\sigma$ be the learning rate at $0.00010.0001$, which controls the rate at which weights update, for each training epoch.

Feature selection is conducted by memetic algorithm, which helps to avoid premature convergence and to reduce data dimensionality and computational resource use, in order to achieve faster convergence. The neural network ensemble using a memetic algorithm (NNE-MA) is compared with well-known optimisation techniques on a similar set of datasets. The techniques are genetic algorithm (GA) (Amjad et al., 2018), ant colony optimisation (ACO) (Xu et al., 2018), and particle swarm optimisation (PSO) (Liu, Li & Zhu, 2019). The neural network ensemble is then combined with each of these techniques for feature selection optimisation, which results in, respectively, the NNE-GA, NNE-ACO, and NNE-PSO convergence comparisons, as shown in Figure 4.

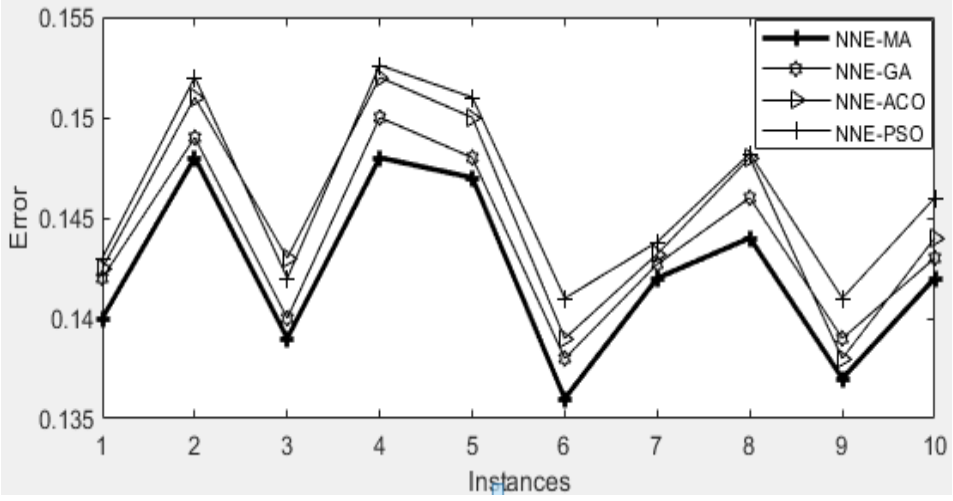
Figure 4: Convergence comparisons between NNE-MA and NNE with existing algorithms



The training error as defined in Equation (9) (presented earlier, in section 3) represents the difference between the current output and the desired output, as per the labels. The actual output is defined in Equation (11) (presented earlier, in section 4), based on weights and bias of neural networks. The error that is generated is measured by the difference between the expected and obtained outputs based on the specific threshold value.

The neural network ensemble using a hybrid search (NNE-MA) produces the least error across all epochs, as shown in Figure 5. This can be ascribed to the improvement in the search mechanism embedded in the memetic algorithm to improve the fitness of solutions, as well as to provide a balance between intensification and diversification of the search. The tabu search, as presented in section 4, is applied to the solutions in the sample, so as to search the neighbourhood of each solution for better fitness individuals.

Figure 5: Error comparisons between NNE-MA and NNE with existing algorithms



Once a better solution is obtained, the current individual is replaced with the new solution and this ultimately leads to more accuracy and better quality of the final output for the neural network ensemble, which is demonstrated in Figure 5.

The statistical output in Figure 6 demonstrates a low error mean for the proposed algorithm, compared to other methods. There is also less variability in data when NNE-MA is applied, as shown by comparisons in the standard deviation. N represents the dataset sample.

Figure 6: One-sample statistical analysis

	N	Mean	Std. Deviation	Std. Error Mean
NNE-MA	10	.14230	.004398	.001391
NNE-GA	10	.14797	.011865	.003752
NNE-ACO	10	.14507	.004927	.001558
NNE-PSO	10	.14606	.004585	.001450

The degrees of freedom, which subtract one from the valid sample and the mean ratio, are represented by df and t respectively, as shown in Figure 7. Based on the t distribution, the p-value in Sig. (2-tailed) indicates a level that is less than the 0.05 threshold value for determining the significance of the results (Shaffer, 2019). It can therefore be inferred that there exists a significant difference between experimental results generated by the algorithms in this work.

Figure 7: One-sample test analysis

	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
NNE-MA	102.312	9	.000	.142300	.13915	.14545
NNE-GA	39.436	9	.000	.147970	.13948	.15646
NNE-ACO	93.118	9	.000	.145070	.14155	.14859
NNE-PSO	100.737	9	.000	.146060	.14278	.14934

6. Conclusions

In this article, we have made the case for a neural network ensemble, based on a hybrid search mechanism, for malware detection. The approach combines global search and local search heuristics, through a memetic evolutionary search process. The tabu search algorithm is used as the local search technique, to improve the quality and fitness of solutions through scouring the neighbourhood of each solution for better individuals.

After training the model on malware datasets to learn both benign and malicious features, the proposed model is able to detect malicious software and achieve faster convergence when compared with existing techniques. In addition, a proper balance between diversification and intensification of the search is achieved, which enables the algorithm to achieve strong accuracy levels. The experimental results we have presented in this article thus indicate that combining several neural networks in an

ensemble generates strong performance, especially when a memetic algorithm is applied to develop solutions and produce optimal outcomes.

Future work will include creating more algorithmic synergies to improve the ability of the search technique to converge towards high-quality solutions, which is necessary in today's rapidly changing and increasingly risk-ridden cyberspace environments.

References

Abiodun, O., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4*(11), 1–41. https://doi.org/10.1016/j.heliyon.2018.e00938

Acarali, D., Rajarajan, M., Komninos, N., & Zarpelao, B. B. (2019). Modelling the spread of botnet malware in IoT-based wireless sensor networks. *Security and Communication Networks, 2019*, 1–13. https://doi.org/10.1155/2019/3745619

Alsulami, B., & Mancoridis, S. (2018). Behavioral malware classification using convolutional recurrent neural networks. In *13th International Conference on Malicious and Unwanted Software* (pp. 103–111). https://doi.org/10.1109/MALWARE.2018.8659358

Altaher, A., & Barukab, O. M. (2017). Intelligent hybrid approach for Android malware detection based on permissions and API calls. *International Journal of Advanced Computer Science and Applications, 8*(6), 60–67. https://doi.org/10.14569/IJACSA.2017.080608

Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z., Anjum, N., & Asgher, U. (2018). Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Mathematical Problems in Engineering, 2018*, 1–32. https://doi.org/10.1155/2018/9270802

Barriga, J. J., & Yoo, S. G. (2017). Malware detection and evasion with machine learning techniques: A survey. *International Journal of Applied Engineering Research, 12*(18), 7207–7214.

Bereta, M. (2019). Baldwin effect and Lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem. *Memetic Computing, 11*(1), 35–52. https://doi.org/10.1007/s12293-018-0256-7

Center for Machine Learning and Intelligent Systems (2016). Machine learning repository [Website]. University of California, Irvine. Retrieved from https://archive.ics.uci.edu/ml/datasets.php?format=&task=other&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table

Chaimanee, A., & Supithak, W. (2018). A memetic algorithm to minimize the total sum of earliness tardiness and sequence dependent setup costs for flow shop scheduling problems with job distinct due windows. *Songklanakarin Journal of Science and Technology, 40*(5), 1203–1218. doi:10.14456/sjst-psu.2018.148

Chen, H., Su, J., & Qiao, L. (2018). Malware collusion attack against SVM: Issues and countermeasures. *Journal of Applied Sciences, 8*(10), 1–20. https://doi.org/10.3390/app8101718

Choi, J. Y., & Lee, B. (2018). Combining LSTM network ensemble via adaptive weighting for improved time series forecasting. *Mathematical Problems in Engineering, 2018,* 1–8. https://doi.org/10.1155/2018/2470171

Dai, H., Cheng, W., & Guo, P. (2018). An improved tabu search for multi-skill resource-constrained project scheduling problems under step-deterioration. *Arabian Journal for Science and Engineering, 43*(6), 3279–3290. https://doi.org/10.1007/s13369-017-3047-4

Dash, T. (2017). A study on intrusion detection using neural networks trained with evolutionary algorithms. *Soft Computing, 21*(10), 2687–2700. https://doi.org/10.1007/s00500-015-1967-z

Eger, S., Youssef, P., & Gurevych, I. (2018). Is it time to swish? Comparing deep learning activation functions across NLP tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 4415–4424). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1472

Gu, Q., Li, X., & Jiang, S. (2019). Hybrid genetic grey wolf algorithm for large-scale global optimization. *Complexity, 2019*, 1–18. https://doi.org/10.1155/2019/2653512

Hassan, M., & Hamada, M. (2017). A neural networks approach for improving the accuracy of multi-criteria recommender systems. *Applied Sciences, 7*(9), 1–18. https://doi.org/10.3390/app7090868

Jat, D. S., Dhaka, P., & Limbo, A. (2018). Applications of statistical techniques and artificial neural networks: A review. *Journal of Statistics and Management Systems, 21*(4), 639–645. https://doi.org/10.1080/09720510.2018.1475073

Jerlin, A. M., & Marimuthu, K. (2018). A new malware detection system using machine learning techniques for API call sequences. *Journal of Applied Security Research, 13*(1), 45–62. https://doi.org/10.1080/19361610.2018.1387734

Ju, C., Bibaut, A., & Van der Laan, M. (2018). The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics, 45*(15), 2800–2818. https://doi.org/10.1080/02664763.2018.1441383

Kabanga, E. K., & Kim, C. H. (2018). Malware images classification using convolutional neural network. *Journal of Computer Science and Communications, 6*(1), 153–158. https://doi.org/10.4236/jcc.2018.61016

Kalaimannan, E., John, S. K., DuBose, T., & Pinto, A. (2017). Influences on ransomware's evolution and predictions for the future challenges. *Journal of Cyber Security Technology, 1*(1), 23–31. https://doi.org/10.1080/23742917.2016.1252191

Kendrick, P., Criado, N., Hussain, A., & Randles, M. (2018). A self-organising multi-agent system for decentralised forensic investigations. *Journal of Expert Systems with Applications, 102*, 12–26. https://doi.org/10.1016/j.eswa.2018.02.023

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

Khoshhalpour, E., & Shahriari, H. R. (2018). BotRevealer: Behavioral detection of botnets based on botnet life-cycle. *The ISC International Journal of Information Security, 10*(1), 55–61.

Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. (2017). URLNet: Learning a URL representation with deep learning for malicious URL detection. In *Proceedings of ACM Conference* (pp. 1–13). Washington, DC. Retrieved from https://arxiv.org/pdf/1802.03162.pdf

Le, Q., Boydell, O., Namee, B. M., & Scanlon, M. (2018). Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation, 26*, 118–126. https://doi.org/10.1016/j.diin.2018.04.024

Lee, T., & Kwak, J. (2016). Effective and reliable malware group classification for a massive malware environment. *International Journal of Distributed Sensor Networks, 2016*, 1–6. https://doi.org/10.1155/2016/4601847

Li, H., Wang, X., & Ding, S. (2018). Research and development of neural network ensembles: A survey. *Journal of Artificial Intelligence Review, 49*(4), 455–479. https://doi.org/10.1007/s10462-016-9535-1

Liu, Z., Li, H., & Zhu, P. (2019). Diversity enhanced particle swarm optimization algorithm and its application in vehicle lightweight design. *Journal of Mechanical Science and Technology, 33*(2), 695–709. https://doi.org/10.1007/s12206-019-0124-5

Lucay, F. A., Galvez, E. D., & Cisternas, L. A. (2019). Design of flotation circuits using tabu-search algorithms: Multispecies, equipment design, and profitability parameters. *Minerals, 9*(3), 1–22. https://doi.org/10.3390/min9030181

Ma, Z., Wang, P., Gao, Z., Wang, R., & Khalighi, K. (2018). Ensemble of machine learning algorithms using the stacked generalization approach to estimate the warfarin dose. *PLoS ONE, 13*(10), 1–12. https://doi.org/10.1371/journal.pone.0205872

Mahrin, M. N., Chuprat, S., Subrarao, A., Ariffin, A. F., Talib, M. Z., Darus, M. Z., & Aziz, F. A. (2018). Malware prediction algorithm. *Journal of Theoretical and Applied Information Technology, 96*(14), 4660–4676.

Martinelli, F., Marulli, F., & Mercaldo, F. (2017). Evaluating convolutional neural network for effective mobile malware detection. *Procedia Computer Science, 112*, 2372–2381. https://doi.org/10.1016/j.procs.2017.08.216

Mohammadi, S., & Namadchian, A. (2017). A new deep learning approach for anomaly base IDS using memetic classifier. *International Journal of Computers Communications & Control, 12*(5), 677–688. https://doi.org/10.15837/ijccc.2017.5.2972

Nguyen, P. T., & Sudholt, D. (2018). Memetic algorithms beat evolutionary algorithms on the class of hurdle problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 1071–1078). Kyoto: ACM. https://doi.org/10.1145/3205455.3205456

Ojha, V. K., Abraham, A., & Snasel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence, 60*, 97–116. https://doi.org/10.1016/j.engappai.2017.01.013

Okobah, I. P., & Ojugo, A. A. (2018). Evolutionary memetic models for malware intrusion detection: A comparative quest for computational solution and convergence. *International Journal of Computer Applications, 179*(39), 34–43. https://doi.org/10.5120/ijca2018916586

Rad, B. B., Nejad, K. H., & Shahpasand, M. (2018). Malware classification and detection using artificial neural network. *Journal of Engineering Science and Technology, 13*, 14–23.

Razak, M. F., Anuar, N. B., Othman, F., Firdaus, A., Afifi, F., & Salleh, R. (2018). Bio-inspired for features optimization and malware detection. *Arabian Journal for Science and Engineering, 43*(12), 6963–6979. https://doi.org/10.1007/s13369-017-2951-y

Ren, B., Liu, C., Cheng, B., Guo, J., & Chen, J. (2018). MobiSentry: Towards easy and effective detection of android malware on smartphones. *Mobile Information Systems, 2018*, 1–14. https://doi.org/10.1155/2018/4317501

Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *Computer Security, 77*, 578–594. https://doi.org/10.1016/j.cose.2018.05.010

Schweidtmann, A., & Mitsos, A. (2019). Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications, 180*(3), 925–948. https://doi.org/10.1007/s10957-018-1396-0

Selvaganapathy, S., Nivaashini, M., & Natarajan, H. (2018). Deep belief network based detection and categorization of malicious URLs. *Information Security Journal: A Global Perspective, 27*(3), 145–161. https://doi.org/10.1080/19393555.2018.1456577

Shaffer, L. K. (2019). Before $p < 0.05$ to beyond $p < 0.05$: Using history to contextualize p-values and significance testing. *The American Statistician, 73*(1), 82–90. https://doi.org/10.1080/00031305.2018.1537891

Shah, A. A., Ehsan, K. M., Ishaq, K., Ali, Z., & Farooq, M. S. (2018). An efficient hybrid classifier model for anomaly intrusion detection system. *International Journal of Computer Science and Network Security*, *18*(11), 127–136.

Shapshak, P. (2018). Artificial intelligence and brain. *Bioinformation, 14*(1), 38–41. https://doi.org/10.6026/97320630014038

Sheng, W., Shan, P., Mao, J., Zheng, Y., Chen, S., & Wang, Z. (2017). An adaptive memetic algorithm with rank-based mutation for artificial neural network architecture optimization. *IEEE Access, 5*, 18895–18907. https://doi.org/10.1109/ACCESS.2017.2752901

Souri, A., & Hosseini, R. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. *Journal of Human-centric Computing and Information Sciences, 8*(3), 1–22. https://doi.org/10.1186/s13673-018-0125-x

Wang, Z., Liu, C., Qiu, J., Tian, Z., Cui, X., & Su, S. (2018). Automatically traceback RDP-based targeted ransomware attacks. *Wireless Communications and Mobile Computing, 2018,* 1–13. https://doi.org/10.1155/2018/7943586

World Economic Forum (WEF). (2018). *The global risks report 2018.* Geneva. Retrieved from http://www3.weforum.org/docs/WEF_GRR18_Report.pdf

Xiao, F., Lin, Z., Sun, Y., & Ma, Y. (2019). Malware detection based on deep learning of behavior graphs. *Mathematical Problem in Engineering, 2019*, 1–10. https://doi.org/10.1155/2019/8195395

Xu, H., Pu, P., & Duan, F. (2018). Dynamic vehicle routing problems with enhanced ant colony optimization. *Discrete Dynamics in Nature and Society, 2018,* 1–13. https://doi.org/10.1155/2018/1295485

Xu, Y., Wu, C., Zheng, K., Wang, X., Niu, X., & Lu, T. (2017). Computing adaptive feature weights with PSO to improve Android malware detection. *Security and Communication Networks*, *2017*, 1–14. https://doi.org/10.1155/2017/3284080

Xue, Y., Jia, W., Zhao, X., & Pang, W. (2018). An evolutionary computation based feature selection method for intrusion detection. *Security and Communication Networks*, *2018,* 1–10. https://doi.org/10.1155/2018/2492956

Yan, J., Qi, Y., & Rao, Q. (2018). Detecting malware with an ensemble method based on deep neural network. *Security and Communication Networks, 2018,* 1–16. https://doi.org/10.1155/2018/7247095

Yin, W., Zhou, H., Wang, M., Jin, Z., & Xu, J. (2018). A dynamic malware detection mechanism based on deep learning. *International Journal of Computer Science and Network Security, 18*(7), 96–102.

Zarras, B. K., Webster, G. D., & Eckert, C. M. (2016). Deep learning for classification of malware system call sequences. In *Australasian conference on artificial intelligence* (pp. 137–149). Wellington, New Zealand: Springer. https://doi.org/10.1007/978-3-319-50127-7_11

Zhirou, Y., & Jing, L. (2018). A memetic algorithm for determining the nodal attacks with minimum cost on complex networks. *Physica A: Statistical Mechanics and its Applications, 503*, 1041–1053. https://doi.org/10.1016/j.physa.2018.08.132