www.turkishstudies.net/turkishstudies

Turkish Studies

eISSN: 1308-2140

INTERNATIONAL BALKAN UNIVERSITY
Sponsored by IBU

Research Article / Araştırma Makalesi

Training Multi-Layer Deep Neural Networks Using Hybrid Algorithms to Detect Network Intrusions

Çok Katmanlı Derin Sinir Ağlarını Hibrit Algoritmalarla Eğiterek Ağlara İzinsiz Girişi Tespit Etmek

Ahmad A A Alfarra*

Abstract: One of the challenges facing security analysts and administrators is to enable intrusion detection in network systems, preventing intruders from entering the network. Network intrusion detection is usually signature-based and plays an effective role in detecting only known attacks. However, there are concerns about unknown attacks and the difficulty arises here. Which requires creating a model that can detect network attacks by discovering anomalies in the network. Therefore, in this article, an approach to detecting attacks is presented by monitoring the behavior of data and training it to distinguish between network traffic, which is normal traffic, and traffic that is classified as an attack on the network. The known database was used NSL-KDD, a dataset that monitors network traffic including various types of network attacks and detects normal traffic, it also contains numerous patterns and features. It includes 41 features for each traffic and is divided into five classifications. The four types are classified as types of network attacks and the fifth as normal traffic. One of the difficulties and problems is the existence of a wide variety of features. This requires finding a way to extract features that can have a significant impact on the results to achieve greater accuracy in classifying the type of attack. For this reason, principal component analysis PCA technology has been used to reduce feature sizes. Six models were created which are hybrid algorithms to optimize the performance of a multi-layer neural network by training their weights to classify the attack type in the network. A two hidden layer deep neural network was designed, and their weights were trained by designing six hybrid models. These models are hybrid algorithms based on four optimization algorithms which are Genetic Algorithm, Artificial bee colony algorithm, firefly algorithm, and Jaya algorithm. Several metrics were used to evaluate the performance of the proposed models. This paper contributed to finding an effective network intrusion detection methodology and developing a deep neural network by building effective optimization algorithms. Where the proposed algorithms performed well with accuracy ranging from 77.177% to 85.077%. The proposed algorithms also achieved a low MSE loss rate close to zero with an MSE value of 9.99868e-05. Also, precision values range from 82 to 97%, and most of the proposed algorithms achieved high rates when calculating the sensitivity and specificity. The proposed models have contributed to finding a high-performance methodology for detecting intrusion on the network and contributed to proving that the hybrid algorithms used are effective and help to the optimization can be used in other studies.

* Yüksek lisans öğrencisi, İstanbul Aydın Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü
Master student, Istanbul Aydın University, Faculty of Engineering, Department of Computer Engineering
ORCİD 0000-0002-1255-3337
ahmedalfarra@stu.aydin.edu.tr

Cite as/ Atıf: Alfarra, A. A. A. (2021). Training multi-layer deep neural networks using hybrid algorithms to detect network intrusions. *Turkish Studies*, *16*(5), 1433-1462. https://dx.doi.org/10.7827/TurkishStudies.51103
Received/Geliş: 27 April/Nisan 2021
Accepted/Kabul: 25 October/Ekim 2021
Published/Yayın: 30 October/Ekim 2021
Checked by plagiarism software
CC BY-NC 4.0

Structured Abstract: Background and objective of the study: Network attack is a constant danger and threat to Internet users, as cyber criminals continue to pursue networks and launch sophisticated attacks from time to time, which poses a threat to user's data. Therefore, this study aims to find an effective methodology capable of detecting attacks on the network to protect the users from hackers. The use of a deep neural network is an example of this type of approach used to analyze data behavior and act upon the neural network's understanding of that behavior. Therefore, the study aims to improve the performance of the deep neural network by removing backpropagation and replacing it with building hybrid algorithms that improve the solution instead of backpropagation. Most of the previous research that cited in this paper is based on a database called NSL KDD which that contains a variety of data that simulates normal traffic and also simulates attack on the network. Most of the researches focuses on two methodologies to develop a model capable and effective in detecting intrusion on the network. Either finding an appropriate mechanism to choose the appropriate features and reducing the dimensions or paying attention to developing the structure of the neural network and improving its performance.

Methodology

For an accurate and effective approach to network intrusion detection, a deep neural network has been constructed consisting of an input layer, two hidden layers and an output layer. The number of neurons in each layer is arranged in sequence as follows 15,9,4,1. In this paper, attention has been paid to choosing the features on which the network is trained by using principal component analysis technology to reduce the dimensions of the features. Choosing the perfect and correct features of a multi-layered neural network is a painstaking and necessary matter to ensure the effective performance and accuracy of a neural network. The Principal Component Analysis technology used to reduce the number of features by combining them using the k means clustering algorithm while preserving data contrast and ensuring its impact on network performance. The neural network is initialized using feedforward propagation. After that, an additional contribution is made to this paper by focusing on developing the structure of the neural network and improving its performance through the work of six models. These models were designed with the goal of training a neural network rather than using the usual back propagation method. These six hybrid models were designed by four algorithms: Jaya algorithm, Bee algorithm, Genetic algorithm, and Firefly algorithm.

Experimental results

Several metrics were relied on to test the performance and accuracy of the six proposed models, which is by evaluating the performance of the model as a whole and comparing the real results with the expected results and finding the error or loss rate by using the mean square error MSE. The second method is to know the number of correct and false predictions for each sample in the database. Then calculate the accuracy for this model based on that. In addition to calculate the confusion matrix, precision, detection rate DR, false alarm rate FAR, specificity, sensitivity, and sensitivity area under the curve AUC. While training the models, the MSE loss rate for the proposed JF algorithm was 9.93325e-05 and 9.999899e-05 for the GAJF algorithm. As we can see, the loss rate is very small and approaches zero, and the more MSE value approaches zero, the better the performance. In other hybrid algorithms, the MSE for GAF was 0.11773, 0.14276 for AFJ algorithm, 0.15147 for AF algorithm, and 0.17897 for proposed GF algorithm. After the samples were tested on test dataset, the JF and GAJF algorithm also achieved a low loss rate as the MSE loss rate for the JF algorithm was 9.99868e-05 and 9.99854e-05 for the GAJF algorithm. As we can see, the two algorithms also performed well. The MSE for GAF was 0.11843, 0.16025 for AF, 0.1971 for AFJ, and 0.21574 for GF. Note that hybrid models when containing Jaya algorithm always achieve high performance then firefly algorithm then ABC algorithm then genetic algorithm. To calculate the accuracy and to know the number of correct and wrong predictions for each sample in the database, the value of a cutoff point was found, and then all numbers greater than this value were changed to one and the smaller ones to zero. In this way, we can compare the true values with the expected values and find out the number of correct and false predictions and based on that we can calculate the accuracy. Most of the proposed hybrid algorithms achieved good accuracy, as the accuracy of AFJ algorithm was 85.077, 81,529 for GF algorithm, 80.951 for GAF, 81,338 for AF, and 77.177 for JF, while the accuracy of GAJF algorithm was 69.095. Also, precision values range from 82 to 97%, and most of the proposed algorithms achieved high rates when calculating the sensitivity and specificity.

Conclusion

Based on four optimization algorithms, six algorithms were designed with the aim of optimizing the weights values in a multi-layered neural network designed with the aim of classification and detection of the type of attacks present in the NSL KDD database. Also, because features selection plays an important role in the classification accuracy, a principal component analysis technique was used which aims to reduce the number of features by combining them to obtain better performance. The proposed six hybrid algorithms performed well. When comparing the results of the neural network with the results of the proposed models for the development of the neural network, we see that the proposed models were able to improve the neural network and obtained better results. and thus, these models contributed to finding a better methodology that can be adopted in other research In addition, the six hybrid algorithms that were used to develop the neural network have proven effective and can also be used in other research as optimization algorithms. Thus, by looking at the results, we see that the proposed methodology achieved good results in general compared to other studies. In research and future work, I highly recommend working to find a suitable and correct cut-off point using the same algorithms because as we can see in the results of the JF algorithm for example, the performance of the algorithm was very high and the MSE loss rate was close to zero, while its accuracy rate was 77.177% due to Failure to obtain a suitable cut-off point. On the contrary, other models achieved good results due to reaching an appropriate cutoff point.

Keywords: Deep learning, Intrusion detection system, NSL-KDD Data set, Metaheuristic algorithms and Classification.

Öz: Güvenlik analistlerinin ve yöneticilerinin karşılaştığı zorluklardan biri, ağ sistemlerinde izinsiz giriş tespitini sağlayarak izinsiz girişlerin ağa girmesini engellemektir. Ağa izinsiz giriş tespiti genellikle imza tabanlıdır veya İngilizcede "signature-based" olarak adlandırılan. Burada terim, bilinen saldırıların ağ trafiğiyle karşılaştırılmasıyla, saldırıların yeniden meydana gelmesi durumunda tespit edilebilecek, bilinen saldırıların bir veritabanına dayalı saldırı tespiti anlamına gelir. Bu yöntem yalnızca bilinen saldırıları tespit etmede etkilidir. Ancak, bilinmeyen saldırılarla ilgili endişeler var ve zorluk burada ortaya çıkıyor. Bu, ağdaki anormallikleri keşfederek ağ saldırılarını algılayabilen bir model oluşturmayı gerektirir. Bu nedenle, bu makalede, verilerin davranışını izleyerek ve normal trafik olan ağ trafiği ile ağa saldırı olarak sınıflandırılan trafiği ayırt etmek için eğiterek saldırıları algılamaya yönelik bir yaklaşım sunulmaktadır. Bilinen veritabanı, çeşitli ağ saldırıları da dahil olmak üzere ağ trafiğini izleyen ve aynı zamanda normal trafiği algılayan bir veri kümesi olan NSL-KDD'yi kullanıldı, ayrıca çok sayıda desen ve özellik içeriyor. Her trafik için 41 özellik içerir ve beş sınıfa ayrılır. Dört tür, ağ saldırı türleri ve beşinci normal trafik olarak sınıflandırılır. Zorluklardan ve sorunlardan biri, çok çeşitli özelliklerin varlığıdır. Bu, saldırı türünü sınıflandırmada daha fazla doğruluk elde etmek için sonuçlar üzerinde önemli bir etkisi olabilecek özellikleri çıkarmanın bir yolunu bulmayı gerektirir. Bu nedenle, öznitelik boyutlarını küçültmek için temel bileşen analizi teknolojisi kullanılmıştır. Ağdaki saldırı türünü sınıflandırmak için ağırlıklarını eğiterek çok katmanlı bir sinir ağının performansını optimize etmek için hibrit algoritmalar olan altı model oluşturuldu. İki gizli katmanlı derin sinir ağı tasarlanmış ve altı hibrit model tasarlanarak ağırlıkları eğitilmiştir. Bu modeller Genetik Algoritma, Yapay arı kolonisi algoritması, ateşböceği algoritması ve Jaya algoritması olmak üzere dört optimizasyon algoritmasına dayalı hibrit algoritmalardır. Önerilen modellerin performansını değerlendirmek için çeşitli metrikler kullanıldı. Bu makale, etkili bir ağ saldırı tespit metodolojisinin bulunmasına ve etkili optimizasyon algoritmaları oluşturarak derin bir sinir ağının geliştirilmesine katkıda bulunmuştur. Önerilen algoritmaların %77,177 ile %85,077 arasında değişen doğrulukla iyi performans gösterdiği yerlerde. Önerilen algoritmalar ayrıca 9.99868e-05 MSE değeri ile sıfıra yakın düşük bir MSE kayıp oranı elde etti. Ayrıca kesinlik değerleri %82 ile %97 arasında değişmektedir ve önerilen algoritmaların çoğu, duyarlılık ve özgüllük hesaplanırken yüksek oranlar elde etmiştir. Önerilen modeller, ağdaki izinsiz girişleri tespit etmek için yüksek performanslı bir metodoloji bulunmasına katkıda bulunmuş ve ayrıca kullanılan hibrit algoritmaların etkili olduğunu ve optimizasyona yardımcı olmanın diğer çalışmalarda kullanılabileceğini kanıtlamaya katkıda bulunmuştur.

Anahtar Kelimeler: Derin öğrenme, Saldırı tespit sistemi, NSL-KDD Veri seti, Meta sezgisel algoritmalar ve Sınıflandırma.

Introduction

In an age when everything and all devices are connected to the internet, it is natural to protect data and information on the network from attacks on network systems. These attacks are rapidly developing, requiring the establishment of effective network intrusion detection systems IDS. Here comes the role of cybersecurity to provide systems that protect the network and data from unauthorized access and detect malware and network attacks. IDS's was first created by Anderson in 1980 (Anderson, 1980), and was later developed by Denning in 1987 (Denning, 1987). Later many improvements were made. Network penetration detection systems work by analyzing network traffic and then determining whether that traffic is normal traffic or whether the network is under attack. IDS is divided into two parts; Detecting the presence of attacks on the network by matching the current traffic with previously known data that has been collected in a huge database that contains thousands of types of attacks, thus detecting any attack that occurs on the network. This method can be good for detecting known attacks, but it fails in the face of new attacks. The most effective way is to analyze the network's traffic and train it to understand its behavior and then, based on that behavior, determine whether the network is under attack or not. However, its performance may be very high in terms of a false positive rate (Niyaz et al., 2016:1-2). Reducing the false positive rate is among the challenges and goals in developing a system to detect unknown attacks, as the difficulties are in tracking the actual network traffic and identifying appropriate features from the dataset as the scenarios evolve and change, in addition to taking a lot of time to design an effective model. Therefore, in this research and as a contribution to developing a solution that reduces these errors, six models were created that use the deep neural network method for classification. The performance of the neural network is trained and improved by designing six hybrid optimization algorithms in order to achieve the best performance of the neural network. In addition to the use of principal component analysis technology to ensure minimum dimensions and optimum features. In previous researches, several solutions have been proposed to reduce these problems, which are reviewed in the related work section.

Related work

This section provides an overview of previous literature and research that has improved IDS performance, improved extraction of appropriate features from the database, and solutions to deep neural network problems we may encounter. All relevant previous studies discussed in this section use the popular NSL-KDD database.

Niyaz and others used self-taught learning STL, which is a deep learning approach. This method consists of two steps to classify data: Unsupervised Feature Learning UFL, then classification on labeled data. One of the methods used in UFL is the Sparse autoencoder, which is used to teach features. Weights learn with the auto-encoder back propagation algorithm. It uses the Sigmoid function as an activation function. And it uses the loss function and keep going until the features are taught. This property is then applied to SoftMax regression to classify network traffic into five types, four attack types, and a normal traffic type. This system helps IDS detect network attacks. The database used is NSL-KDD (Niyaz et al., 2017:2-4).

In other article, the correlation feature selection subset CFS to select features that improve prediction performance was used, and this method reduces dimensions. CFS ranks features based on the heuristic evaluation function based on correlations. With this function subsets of attribute vectors are evaluated. Then unnecessary features have low correlation with the class and therefore should be ignored by the algorithm. Thus, eliminating redundant features to better describe the specific problem with minimal performance loss. Thus, this method uses the "rank search" method to rank all features, from this list, csf is calculated for the best subset by looking at the predictive power of each feature together with the degree of redundancy. There are four types of attacks in the NSL KDD database, including U2R and R2L, which are few, and this could lead to bias against other attacks. So, to solve

this problem, it did a random sampling to get rid of the bias training problem, and then the classification was done with the CART algorithm. The model has been compared with other studies and performed well. This is done by reducing the features and then reducing the size to one third the size of the original data (Ingre et al., 2017:4-6).

In another article, Mohammadzadeh et al proposed a model for intrusion detection by building a model that combines two optimization algorithms, which are whale optimization algorithm WOA and flower pollination algorithm (FPA), which he named HWOAFPA. These two algorithms solve the problem of optimizing feature selection. After initial solutions are created by WOA by trapping prey and tracking bubbles, the FPA algorithm optimizes the solution and selects the appropriate feature by using a global and local search process in an opposite space using WOA solutions. The proposed model shows a high accuracy of 99.07 (Mohmmadzadeh et al., 2020:6-17).

Choudharya et al. used a deep neural network to detect network intrusion on the Internet of Things. Where 20 hidden layers were used and only eight features out of 41 features were extracted from the KDD NSL dataset. These features were used as inputs to the deep neural network. In the output, the behavior of network traffic is detected, whether it is an attack or normal traffic. The entire data set was used and divided as follows: 70% for training data, 30% for validation, and 30% for testing. In addition to the KDD NSL dataset, this proposed model also tested on another database, the UNSW dataset, and extracted only eight features. The proposed model in both data sets achieved an accuracy rate above 90%. The accuracy for the KDD NSL data set was 91.5% (Choudharya et al., 2020:1562-1568).

In another article, Ucar et al. used the stacking technique to detect attacks on the network. This approach stacks a number of classification algorithms and combines them to produce a single classifier. The NSLKDD data set was used. The algorithms used are Decision Tree (DT), K Nearest Neighbor (KNN), Neural Network and Logistic Regression (LR) which are considered as base learner models and Support Vector Machine (SVM) which are considered as meta learner. Through the results, it is shown that stacking helps to improve the performance of intrusion detection systems, as the proposed DT-LR-ANN + SVM model achieved an accuracy of 84.32% (Ucar et al., 2021:1331-1336).

In another research paper, Bachar et al. presented a network intrusion detection model based on the Support Vector Machine (SVM) algorithm, which classifies data into two parts, either attack or no attack. The SVM algorithm selects samples in space to increase the width of the gap between the two groups. New samples are classified based on the prediction to determine whether they belong to the first or second group based on their position on either side of the gap. Since the data cannot be separated linearly, the kernel trick was used to obtain an ideal data separation, where two kernels were used, namely SVM-Polynomial. Whereas SVM-Polynomial is not only concerned with the specific features of the input samples but also looks at the combinations of these elements to determine the similarity between them allowing the learning of non-linear models. The SVM-Gaussian kernel is also used, and its value is calculated by calculating the distance from one of the points. The proposed model achieved a high performance, as the accuracy of the SVM-Gaussian kernel was 93%, while the accuracy of the SVM-Polynomial kernel was 94% (Bachar et al., 2020:641-643).

In one of the previous research, Ghanem et al Proposed a way to solve the problem of overfitting and the problem of reaching local minimums in the neural network using a hybrid algorithm based on the bee algorithm and the dragonfly algorithm. They named it the HAD algorithm. This hybrid algorithm optimizes the weights values in the deep neural network instead of using the back propagation method to update the weights and thus train the network (Ghanem et al., 2020:130456). The optimization algorithms used to reduce the loss function, then when reaching a point where optimization cannot be achieved, new random values are imposed and improved by the

proposed hybrid optimization algorithm. The proposed algorithm was compared with ten other algorithms, and the proposed algorithm performed better in detecting network attacks (Ghanem et al., 2020:130456- 130458).

In another study, Alzahrani et al. designed a system based on machine learning to develop software-defined networks with the aim of detecting network intrusions. They used tree-based machine learning techniques such as decision tree, random forest and XGBoost algorithm. It can be used to classification tasks, and it uses the depth-first search approach. They also used the NSL-KDD database. Min-Max normalization technique was applied to prevent gradation. Five features that have the most impact on algorithm learning performance were selected out of 41 features in the database. The proposed architecture consists of three layers which are the infrastructure layer and include devices such as routers and switches and software that interact with devices such as OpenFlow switches. The software-defined networks SDN control layer is responsible for managing traffic and its flow on the network and the application layer is responsible for managing the network. Network traffic is then analyzed and separated from the SDN console for the purpose of detecting attacks and network intrusions to be blocked by specific rules in the firewall or routers. By using tree-based algorithms, network traffic is classified as attack or natural traffic. When comparing XGBoost with the other two tree-based methods, namely random forest, and decision tree, we find that the XGBoost algorithm achieved a higher result, with an accuracy of 95.55%, while the accuracy of the tree algorithm was 94.6% and the random forest algorithm was 94.5% (Alzahrani et al., 2021:6-11).

In another paper, Rawat et al. Propose a model that integrates uncontrolled feature selection method with Deep neural network DNN. Initially, Principal component analysis uses a technique called PCA to add and combine all similar features and reduce dimensions. This method integrates them by finding the mean normalization for all properties, then computing the covariance matrix, determining the value of k, and then clustering. In this way, the dimensions in the properties are reduced. Then 15 features were extracted from the 41 features then will added to the Deep Neural Network DNN, which is an advanced model of the feed forward neural network and contains 5 hidden layers where the number of neurons and layers are selected after the hyperparameters are studied. Binary cross entropy is used to calculate the error function. The output layer consists of five outputs where the attack type is categorized from the five attack types found in the NSL KDD database. As new intrusions were detected in the network, the model performed well against the others with an accuracy of 0.793. The deep neural network model provided better fit and accuracy in the test set. Rawat also conducted additional analysis using Software defined network properties. Using only six features, the accuracy of the deep learning model in the test set was 0.759 (Rawat et al., 2020:3-5).

In another paper, Istiaque et al. built a model by using the KDD'99 dataset for the purpose of network intrusion detection and testing the model with a different dataset, NSL-KDD where this dataset is a development of KDD'99. He chose 15 out of 42 features and for classification he used the Random Forest and MLP algorithm. The model achieved an accuracy of 70.808% when using 15 features and 97.233% when using 22 features (Istiaque et al., 2021:149-150).

In another paper, Tang et al. propose a model based on the use of stacked autoencoder (SAE) and attention mechanism and deep neural network (DNN) to detect network intrusion. After using min-max normalization technology to scale the data, and after using the one-hot encoding technique, which converts the values to zero or one, The features that contain zeros are then removed, and the data processing stage is completed. The attention mechanism is then added to SAE, where SAE stands for Stacked Autoencoder which is a symmetrical structure created by Autoencoder which uses an encoding and decoding process to reduce feature dimensionality and is classified as unsupervised learning. It contains three layers, an input layer and an output layer which are equal in size and the middle layer should be smaller than the input layer. After training the Autoencoder the middle layer

is used to train the second Autoencoder and with the process iterating and stacking the Autoencoder we achieve greedy hierarchical learning. This process is leveraged to train the weights of the deep neural network rather than randomly generating them. Although the data has been learned, it may contain redundant features, so it is added to the attention mechanism, which contains a value vector. This process is called Stacked Attention Autoencoder (SAAE). The attention mechanism aims to eliminate unnecessary features to improve network performance. Where the value in the vector is set to 0 when the feature is unnecessary and does not contribute to the prediction. After that, the intrusion is detected on the network by using the deep neural network, which consists of an input layer, an output layer, and two hidden layers. The SAAE-DNN model achieved an accuracy of 87.74%, which is excellent (Tang et al., 2020:2-10).

In another paper, it was proposed to use deep learning to detect zero-day attacks using the Autoencoder system, a neural network that tries to make its output equal to its input to learn features and reduce dimensions. Next, a single class SVM model for classification was created to compare its performance with the proposed Autoencoder. The goal is to maintain the lowest acceptable level of the false positive rate and an acceptably high rate of the true positive rate. The results show that the performance of automatic encryption devices enables zero-day attacks detection with high accuracy. The results showed detection accuracy of 89-99% for the NSL-KDD dataset and 75-98% for the CICIDS2017 dataset in zero-day attacks (Hindy et al., 2020:5-9).

In another article, Khreisat et al. built an intrusion detection model consisting of two classifiers, a C5.0 decision tree classifier and a one-class support vector machine. Where the NSL-KDD database was used. They built the model to be able to detect and predict attacks from the day a falcon. In C5.0 decision tree classifier attacks are detected based on knowledge of previous attacks. Whereas in a one-class support vector machine it is detected using anomaly-based IDS. The proposed model achieved a good accuracy of 83.24% (Khraisatet et al., 2020:4-7).

In another article, Hosseini et al built a two-stage model for network attack detection. The first stage is to discover and select the appropriate features by means of a wrapper method, which is a method that contains the features of the SVM and the genetic algorithm. The second stage is to detect the intrusion process on the network by using the artificial neural network. Whereas a hybrid gravitational search and PSO algorithm were used to improve the performance of the proposed algorithm. The data set used is the NSL-KDD dataset and only 4 out of 42 features were selected. The proposed model has been referred to as MGA-SVMHGS-PSO-ANN. The proposed model achieved good performance with an accuracy of 95.2% (Hosseini et al., 2020:8-24).

In another article a deep learning approach was applied to detect network intrusion in an SDN environment. By building a deep neural network (DNN) model for an intrusion detection system and training the model. It uses six of the 41 features of the NSL-KDD dataset which can be obtained in an SDN environment. The software-defined network environment (SDN) separates the control plan in the network from the data plane, which enables control of the network through the control plan or the so-called controller, which gives commands to the switches and routers in the network via the OpenFlow protocol. A model for the detection of flow-based network attacks using a deep neural network (DNN) in an SDN environment has been proposed. Parameters have been chosen which includes an input layer, 3 hidden layers, and an output layer. The number of the input neurons are 6, the output neurons are 2, and in the hidden layers there are 12,6,3 neurons, respectively. The Controller or SDN monitors all OpenFlow switches. The Controller sends the ofp_flow_stats_request request to all OpenFlow switches after a specified period for the network statistic request. Then OpenFlow switches respond to the Controller with ofp_flow_stats_reply and give it all the stats. After that, all these network statistics are sent to the deep neural network for analysis and detection of any network penetration. The model performed well with an accuracy of 75.75% (Tang et al., 2016:2-3).

Data set

The dataset for IDS is usually created manually or made available to researchers. The DARPA dataset is an example of data available to researchers. Also, KDD 99 data from DARPA 98 which contains 41 features plus another classification feature. The dataset used in this study is NSL-KDD, which is derived from the KDD 99 dataset, and contains 41 other features and classification features. The NSL-KDD dataset is a modified version of the KDD99 dataset because it contains unnecessary records, which adds to the complexity (Dhanabal & Shantharajah, 2015). The algorithms proposed in this research were trained on KDDTrain data, which includes 22 attack types, and performance was tested on KDDTest data containing 17 additional attack types. These attacks are divided into four different types of training and testing as shown in Table 1.

Table 1: Type of Attacks in NSL KDD Dataset

DOS	Probe	R2L	U2R
apache2	ipsweep	Spy	bufferoverflow
Back	Mscan	Warezclient	loadmodule
Land	Nmap	ftp_write	Perl
Mailbomb	portsweep	Guesspasswd	Ps
Neptune	Saint	Httptunnel	Rootkit
Pod	Satan, xsnoop	Imap	Snmpguess
processtable	multihop		sqlattack
smurf	named		worm
teardrop	phf, xlock		xterm
udpstorm	sendmail		
	snmpgetattack		
	warezmaster		

As shown in Table 1, the four categories of attacks are Denial of Service (DoS), which is an attempt to disrupt system or network services. Probe, this type of attack detects potential vulnerabilities of the target network that will be used in subsequent attacks. The third type of attack is called Remote to Local (R2L) and can be accessed as a user and remotely control the network for unauthorized activities. Finally, a User to Root (U2R) attack where attackers compromise vulnerabilities to gain access to the system. In addition to normal network traffic, all these types were tracked and compiled in the NSL-KDD database (Dhanabal & Shantharajah, 2015:449). Table 2 show the number of samples in each category, where the KDDTrain set contains 125,973 training samples and 22544 samples in the KDDTest test set. KDDTest test set includes 17 additional unknown attacks to test.

Table 2: Number of patterns for each type of attack

KDDTrain		KDDTest	
Type of attack	No.of patterns	Type of attack	No.of patterns
Normal	67343	Normal	9711
DOS	45927	DOS	7458
Probe	11656	Probe	2421
R2L	995	R2L	2754
U2R	52	U2R	200
Total	125973	Total	22544

Methodology

By reviewing the previous literature, we note that the research methodology focuses either on contributing to finding an appropriate mechanism through which to select the best features and reducing their dimensions in order to train the data or contributing to the development and improvement of the neural network architecture. Therefore, in this paper both methods are used with the same NSL-KDD database used in previous researches. Choosing the right and appropriate features affects the performance and accuracy of a deep neural network. Therefore, her choice is essential. The database contains 41 features, so initially the features are tagged and dimensioned using principal component analysis. Through this technique, the dimensions of these features were reduced to 15 features by merging the features that are close and interconnected with each other using the k mean clustering algorithm while maintaining the data contrast at 80%. These features are used as inputs in the Deep Neural Network input layer which consists of two layers, an input layer, an output layer, and two hidden layers. The first hidden layer contains 9 neurons, the second contains 4 neurons, and the output layer contains one neuron, as shown in Figure 1.

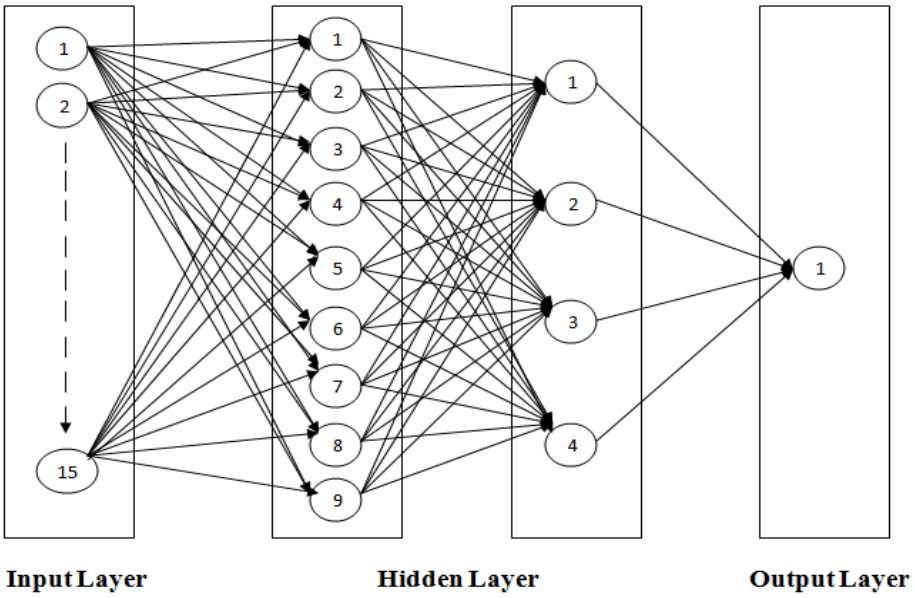


Figure 1: Proposed deep neural network

The deep neural network is then initialized, and the activation function is calculated by feed forward propagation as shown in equations (1), (2) and (3) below. To calculate the accuracy and effectiveness of the model, the mean square error MSE is used to calculate the loss and error ratio as shown in equation (4).

$$zi = \sum_{i=1}^{n} wij\ xi + \beta \quad (1)$$

$$fi(x) = \frac{1}{1+e^{-zi}} \quad (2)$$

$$yi = fi(\sum_{i=1}^{n} wij\ xi + \beta) \quad (3)$$

Where n represents the number of inputs, w represents the weights where we have 175w and their values range from zero to one, b represent the bias and the x inputs. To calculate the accuracy and effectiveness of the model, the mean square error MSE is used to calculate the loss and error ratio. The goal of using the MSE function is to find out the loss value, classification error value, or prediction after training a deep neural network, so the closer this value is to zero, the better. Thus,

the objective function as shown in equation (4) is the lowest value that can be obtained from the MSE function.

$$(Min)MSE \quad Y = \frac{1}{n} \sum_{i=1}^{n} (yi - yi')^2 \qquad (4)$$

Where yi represents the actual value, yi' represents the values obtained and predicted, and n represents the total number of samples.

Usually, the backpropagation algorithm is used to modify the values of weights and bias in order to improve the performance of the neural network. This is done by finding the difference between the real and expected values, then followed by partial derivation, and this process is repeated until we get the best solution. After each iteration we will update the weights by calculating gradients and since they are composite functions it will derive by using chain rule, after that we update the weights and bias values by using equations 5 and 6.

$$Wnew = Wold - \alpha \frac{dY}{dW} \qquad (5)$$

$$\beta new = \beta old - \alpha \frac{dY}{d\beta} \qquad (6)$$

Where W represents the weights, β represents the bias, α represents the learning rate and Y represents the loss function in equation 4.

Usually, deep neural network training is done by updating the weights values through the use of backpropagation method as shown in equation 5 and 6, but in this paper another method was used to update the weights. This method is an additional contribution and differs from previous researches where no optimizer will be added and no iteration will be used to modify the values of weights and biases based on backpropagation, but rather hybrid algorithms will be built and used instead with aim of optimizing the neural network and obtaining better results. Where 6 models were designed based on 4 optimization algorithms to optimize the objective function referred to in Equation (4). These algorithms are Jaya algorithm, ABC algorithm, genetic algorithm, and firefly algorithm.

Genetic algorithm

A genetic algorithm is an optimization algorithm that is a simulation of biological behavior in reproduction and survival based on Darwinism. It is considered a method of research and used to find optimal and difficult solutions. Beginning and after determining the objective function to be reached, then random elements are generated and considered as living organisms ready to reproduce. After that, the evaluation process, and the selection of the best elements for their selection for the mating process takes place. Because the elements with good qualities generate other elements that give good solutions. Then the mating process takes place, which consists of the crossover phase, which exchanges genes, and the mutation phase, which slightly changes some of these genes. This mating results in a new generation that is evaluated again, and the best elements are chosen from it. And so on until we get the best possible solution (Reeves, 2010). Figure 2 shows the pseudo-code of how the algorithm works.

Pseudo-code for a Genetic algorithm

```
(1)  Set parameters
(2)  Choose encode method
(3)  Generate the initial population
(4)  while i < MaxIteration and Bestfitness
     < MaxFitness do
(5)      Fitness calculation
(6)      Selection
(7)      Crossover
(8)      Mutation
(9)  end while
(10) Decode the individual withmaximum
     fitness
(11) return the best solution
```

Figure 2: Pseudo-code for a Genetic algorithm

Artificial Bee Colony Algorithm

ABC algorithm is another optimization algorithm that mimics the behavior of bees in foraging for food. It is used to find and optimize an optimal solution for many applications (Karaboga & Basturk , 2007). Common control parameters such as population size, number of iterations, and number of bees are defined for each phase. After determining the objective function to be solved, which is considered as a food source that the bees want to reach. The solution will be divided into three phases called the employed bee phase, onlooker bee phase and scout bee phase. Employed bees generate random initial values as food sources. Then it chooses a partner to acquire a new position that will serve as a source of food. Then it compares the old source and the new source to select the best one by applying a greedy selection. This is done by means of equation (7).

$$Xnew = x + \emptyset(x - xp) \quad (7)$$

The value of x indicates the current position, and xp indicates the position of the partner. ∅ is a random value ranging from -1 to 1.

Then, in the second phase, the onlooker bees select the best solution by calculating the probability among the proposed solutions as shown in equation (8), and then the optimal solution is the best probability. The best probability is determined by equation (9).

$$\text{probability i} = \frac{MSEi}{\sum MSEi} \quad (8)$$

$$r < \text{probability i} \quad (9)$$

Where r is a random number between zero and one. After that, it will select a neighboring resource and compare the two sources to choose the best one by applying greedy selection as in the first phase. In the third phase, the scout bees search for a new source to evaluate and compare it with the sources that have been found, with the aim of expanding the search circle and replacing and replacing bad sources or solutions. The algorithm ends with several iterations that are determined in

advance until we reach the best solution that can be reached. Figure 3 shows the pseudo-code of how the algorithm works.

Pseudo-code for a ABC algorithm

(1) generate the initial population xi (i = 1,2,...,SN)
(2) Evaluate the fitness of population
(3) Set Iteration to 1
(4) **Repeat**
(5) **For** each employed bee {
Produce new solution vi by using **(2)**
Calculate its fitness value fit (vi)
Apply greedy selection process }
(6) Clulate the probability values Pi for the solution (xi) by **(3)**
(7) **For** each onlooker bee {
Select a solution xi depending on Pi
Produce new solution vj
Calculate its fitness value fit(vj)
Apply greedy selection process }
(8) **If** there is an abandoned solution for the scout,
then replace it with a new solution which will be randomly produced by **(4)**
(9) Memorize the best solution so far
(10) Iteration = Iteration +1
(11) **Until** Iteration = Max Iteration No.

Figure 3: Pseudo-code for an ABC algorithm

Firefly Algorithm

The firefly algorithm is a swarm-based algorithm and is considered an optimization algorithm. Its classified according to metaheuristic algorithms. This algorithm simulates fireflies in the way they communicate. The fireflies produce a flash to attract their companions from other fireflies as well as to warn predators. Depending on the intensity of the light, the fireflies are attracted towards each other, and naturally the intensity of the light decreases with increasing distance. This means that the fireflies will be attracted to other, brighter fireflies (Yang & He, 2013). Based on this principle, this behavior can be used and simulated to improve our objective function in equation (4). This principle is applied by creating random weight values called fireflies and then applying equation (4) to calculate the error values for these random values. And since our aim is miniaturization, the brighter the less valuable. Where larger error values are pulled towards lower values. Thus, these errors are minimized by using equation (10), which simulates the behavior of fireflies.

$$Xnew = xi + \beta oe^{-\gamma rij^2}\,(xj - xi) + \alpha o\delta\left(rand - \frac{1}{2}\right)\,scal \qquad (10)$$

Xnew indicates the new position to be acquired, xi the current position of the firefly, and xj indicates where the fireflies will be attracted and is the brightest. As our goal is miniaturization, this means the brighter is less valuable. As shown in equation (11), the brightness of the light decreases along the distance according to the inverse square law.

$$I = Io\; e^{-\gamma r} \qquad (11)$$

γ Indicates the light absorption coefficient and the light intensity at a distance r can be calculated as in equation (12). Where r is the distance between fireflies.

$$rij = \sqrt{(xi - xj)^2 + (yi - yj)^2} \qquad (12)$$

Where *I0* expresses the intensity of light and similarly the brightness is β as in equation (13).

$$\beta\; = \beta\, o\; e^{-\gamma r} \qquad (13)$$

β0 denotes the attractiveness of xj at r = 0, according to the source it is preferred that β0= 1. The position of the firefly's update was based on the distance between the fireflies, and the was set to 0.01.

α describes the random motion step and is calculated as follows.

$$\alpha = \alpha o\; \delta^t\;;\;\; 0 < \delta < 1 \qquad (14)$$

αo refers for randomness strength and its value between zero and one. δ indicates randomness reduction, and its value is between 0.95 and 0.97.

After applying equation (10), we get new weights and values. These new values are applied in our objective function in Equation (4) and calculate the MSE error value. Then the greedy approach is applied to compare the new and old error values. If the new error value is less than the previous value, the previous weights will be replaced with the new ones, and thus the algorithm has succeeded in reducing the error value. This process is repeated until you achieve the best possible improvement. Figure 4 shows the pseudo-code of how the algorithm works.

Pseudo-code for a Firefly algorithm

```
(1) Initialisation max iteration ,α, βo,γ
(2) Generate initial population
(3) Define the Objective function f(x),
(4) Determine Intensity (I) at (x) of each
    individual determined by f(xi)
(5) While (t < IterationMax)
            For i=1 to n
                For j=1 to n
                  if (Ij>Ii)
                     Move firefly i towards
                     j in K dimension
                  end if
                Evaluate new solution and
                update light intensity
                end for j
            end for i
          Rank the fireflies and find the
          current best
     end while
(6) Post process results and visualization
End procedure
```

Figure 4: Pseudo-code for a Firefly algorithm

Jaya Algorithm

Most algorithms that rely on evolutionary intelligence and swarm are algorithms that require common control parameters such as the number of generation or population size. In addition, it needs other control parameters of the algorithm itself. For example, the genetic algorithm, in addition to identifying common parameters such as determining the number of iterations or population size, etc., required other special parameters of the genetic algorithm, such as crossover and mutation, and the same with other algorithms. As the appropriate setting of the special parameters is very important and affects the performance of the algorithm, it increases the computational effort and reaches local optimal solution instead of reaching the global optimal solution. Unlike other algorithms, Jaya algorithm is an algorithm without parameters of the algorithm. It is considered as metaheuristic algorithm, able to obtain high quality solution and used to solve optimization problems (Rao, 2016). After generating the elements or weights that range from zero to one, we will find the best and worst solution in the set. Then we will calculate a new value for each element in the group based on Equation 15.

$$Xnew = Xi,k + r1(xbest - |Xi,k|) - r2(xworst - |Xi,k|) \qquad (15)$$

r1 and r2 are random numbers ranging from zero to one. Xi, k represent the values of all the weights in the equation (4), which we aim to improve. As it appears from the equation, the first part indicates that the solution tends to approach the best solution and the second part indicates that the solution tends to approach the worst solution. After we have the new value, it is compared with the

previous value by using greedy selection. And because the objective function to be improved aims to reduce the error value as shown in Equation (4), if the new value is less than the previous value, it will be replaced. Thus, improvement is achieved by creating a new value for each element in the group and repeating the improvement until we reach the best possible solution. Figure 5 shows the pseudo-code of how the algorithm works.

Pseudo-code for a JAYA algorithm

(1)**For** k:=1 to n do (i.e., Set population size)
(2) **For** i :=1 to m do (i.e., Initialize population (Xi,j))
(3) **Evaluate** Xbest and Xworst
(4) **Set** l = 1 (Initialize iteration no.)
(5) **While** != iteration
(6) **Set** r1 & r2 = a random number from [0,1]
(7) **Update** Xnew =
$Xnew = Xi,k + r1(xbest - |Xi,k|) - r2(xworst - |Xi,k|)$
(8) **End**
(9) **If** solution Xnew better than Xi,k
(10) **Set** Xi,k = Xnew
(11) **Else**
(12) **Set** Xi,k = Xi,k
(13) **End**
(14) **Set** l = l + 1
(15) **Update** xbest and xworst
(16) **End While** maximum iterations or termination criterion satisfied
(17) End

Figure 5: Pseudo-code for a Jaya algorithm

Hybrid algorithms architecture design

In this paper, 6 hybrid models were designed based on the four explained algorithms. These hybrid algorithms aim to improve the performance and accuracy of the deep neural network indicated in Figure 1 and reduce the value of the error function. As we see when configuring all the mentioned four algorithms, the first step is to generate initial random elements, and then the optimization process takes place. This randomness may not guarantee us a good start, which may increase the computational effort or reduce the improvement process after that. But when using a hybrid algorithm that depends on more than one algorithm, the outputs of the first algorithm that are improved are the initial elements of the second algorithm, which increases the effectiveness of the optimization.

In the first model, the hybrid algorithm was named GAF algorithm, which denotes the first letter of the genetic algorithm, artificial bee colony algorithm, and the firefly algorithm. After the deep neural network indicated in Figure 1 has been initialized, which contains a sum of weights equal to 175. Five random solutions were generated for these weights and each solution contains 175 w. Where the letter w denotes the word weight. The value of each weight in the neural network ranges

from zero to one. Then the activation function is calculated and the objective function MSE referred to in equation 3 and 4 is calculated for each solution. After the objective function MSE is calculated for the five random solutions, the best 3 solutions are chosen to complete the process of mating and exchange of genes by implementing the genetic algorithm. The goal is to reduce the error value as much as possible, so the best solution is the smallest value. After choosing the lowest 3 values, their weights are used for mating, where the genes are exchanged and mutations occur between the first and second elements, the second and third elements, and the first and third elements. A randomize mutations and a uniform crossover was used to exchange genes. These 3 elements will generate 3 more new elements for which the objective MSE function is calculated, to become 6 elements. The best 3 elements are selected again to continue the optimize process. In addition, the best 4 elements are selected in order to use their weights for the second stage, which is the artificial bee colony algorithm. The iterations at this point were set to 18 times.

In an artificial bee colony algorithm, only the first and second phases were used, namely employed bee phase used and the onlooker bee phase. After arriving at the 4 best possible solutions from the genetic algorithm that contain weights. In the first phase of the ABC algorithm, the bees search for new locations for food sources and evaluate them by exchanging the values of the weights with neighboring solutions and applying the equation 7. Then, after obtaining the new weights, they are evaluated by calculating the objective function and comparing it to the previous values. A greedy selection was used for the evaluating. In the second phase of the bee algorithm the probabilities are calculated for the obtained MSE values. The value of r was set to 0.23. If the probability value is less than r, the values of the weights are exchanged with neighboring solutions, as in the first phase. We also calculate the MSE values, and the evaluation is done by using greedy selection as well. The same process is repeated 8 times to reach the best possible improvement. After that, the last weights that were optimized by the ABC algorithm are used and used for the firefly algorithm.

In the firefly algorithm, after adjusting the parameters, the fireflies with large error values are attracted to the fireflies with smaller values, in order to complete the process of miniaturization and improvement of the solution. This is done by using equation 10 to obtain new values for the weights. After calculating the MSE values, they are evaluated and compared to the previous values by using greedy selection. This optimization process is repeated 8 times. After that, the best solution from the four solutions is chosen as the best solution that the algorithm can reach. Figure 6 shows the hybrid GAF algorithm. In this paper, other models are designed in order to find out which models can reach the best possible solution.

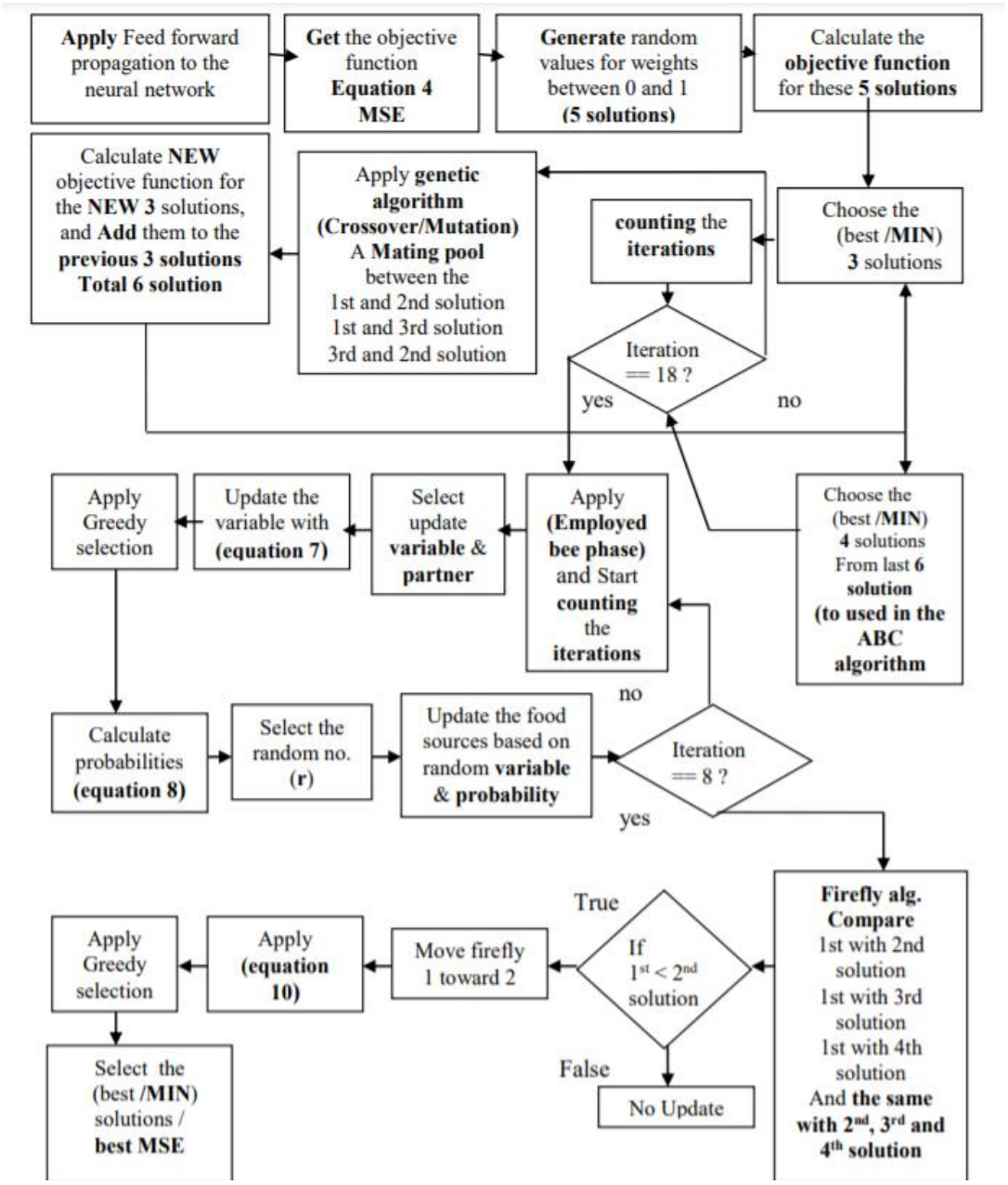


Figure 6: Hybrid GAF algorithm

In the second model, a hybrid algorithm was designed consisting of only two algorithms, namely the genetic algorithm and the firefly algorithm, and it called the GF algorithm. It works with the same idea indicated in the first model. After making the neural network, the genetic algorithm was applied with the same parameters as in the first model, and then the firefly algorithm was applied with the same parameters as well. Figure 7 shows the hybrid GF algorithm.

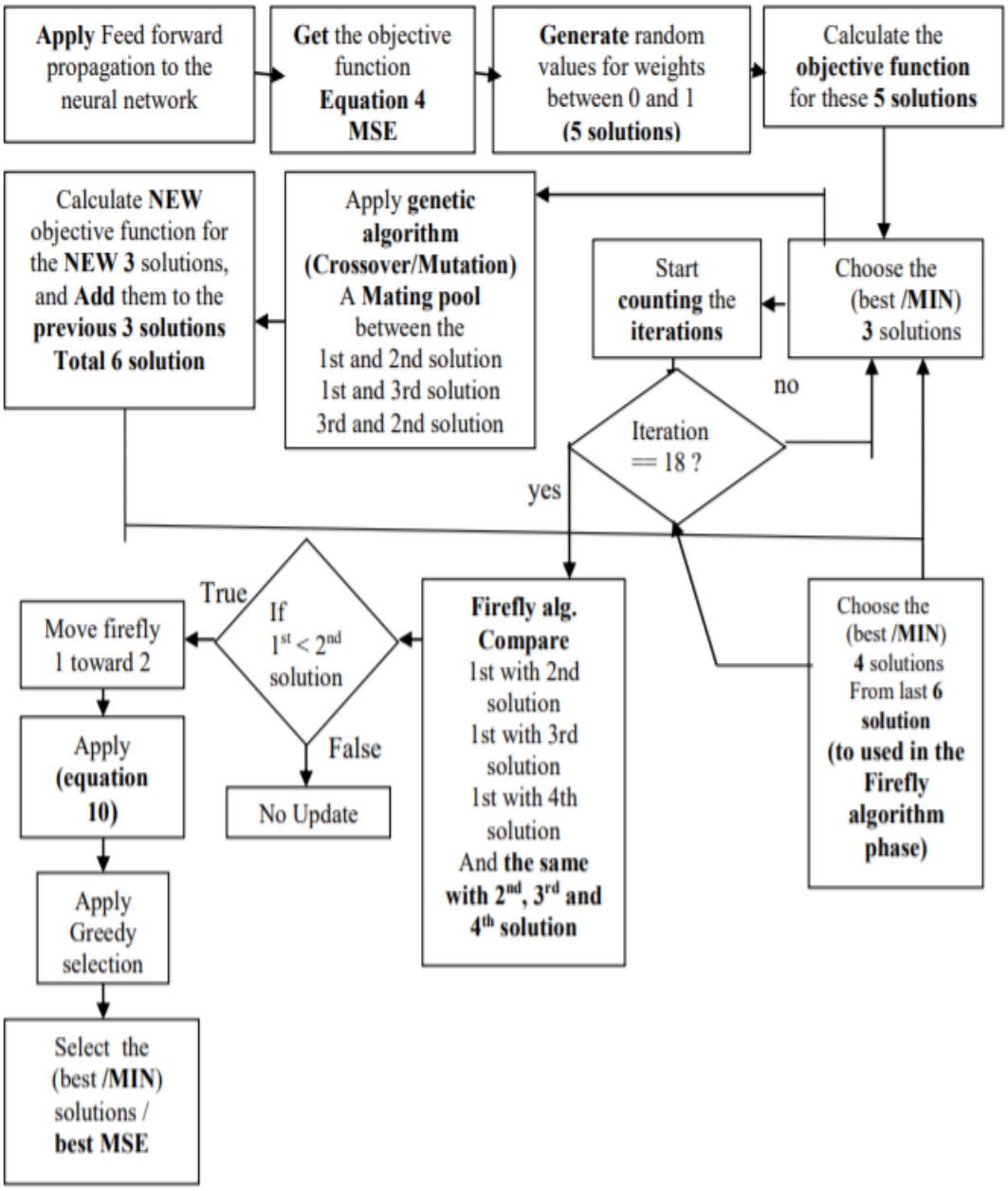


Figure 7: Hybrid GF algorithm

In the third model, the performance of the deep neural network is improved by using a hybrid algorithm consisting of the ABC algorithm and the firefly algorithm, and it called the AF algorithm, referring to the first letter of each algorithm. The solution starts with assuming 4 random solutions for weights after which they are optimized by using the hybrid algorithm. After that, and after the process of iteration and improvement, only one solution is chosen, which is the best solution from the four solutions. Figure 8 shows the hybrid AF algorithm.

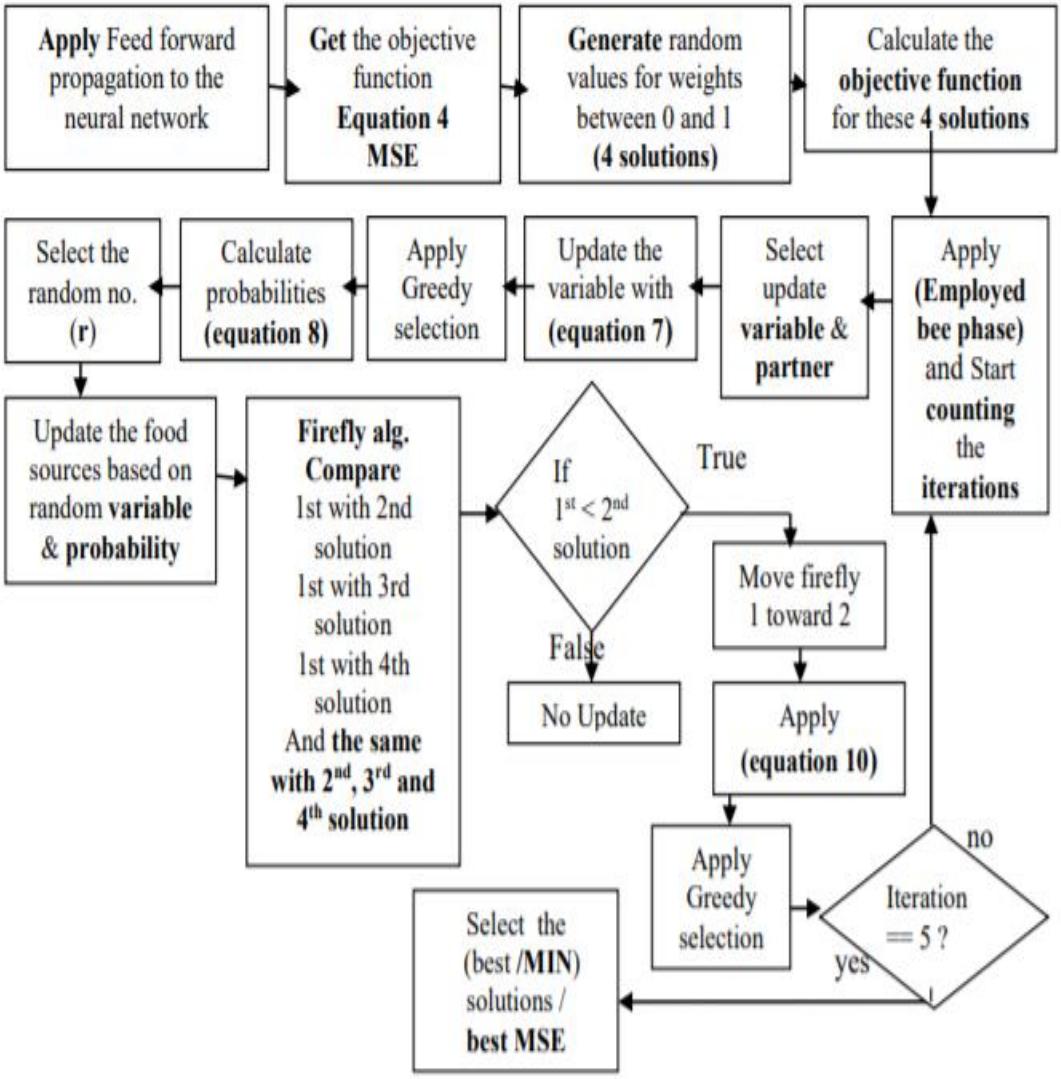


Figure 8: Hybrid AF algorithm

In the fourth model, a hybrid algorithm was designed based on 3 algorithms which are bee algorithm, firefly algorithm and Jaya algorithm. It is called AFJ algorithm, referring to the first letter of each algorithm. The first and second phases of the ABC algorithm was used and imposing 4 random solutions at the beginning for the weight's values. In addition, and unlike other models, firefly algorithm was combined with the ABC algorithm and treated as one algorithm. Meaning that in a single iteration, the output of the bee algorithm is the input of the firefly's algorithm, and it is repeated five times. Then the Jaya algorithm referred to in equation 15 is applied with the optimization process iterated 10 times. Figure 9 shows the hybrid AFJ algorithm.

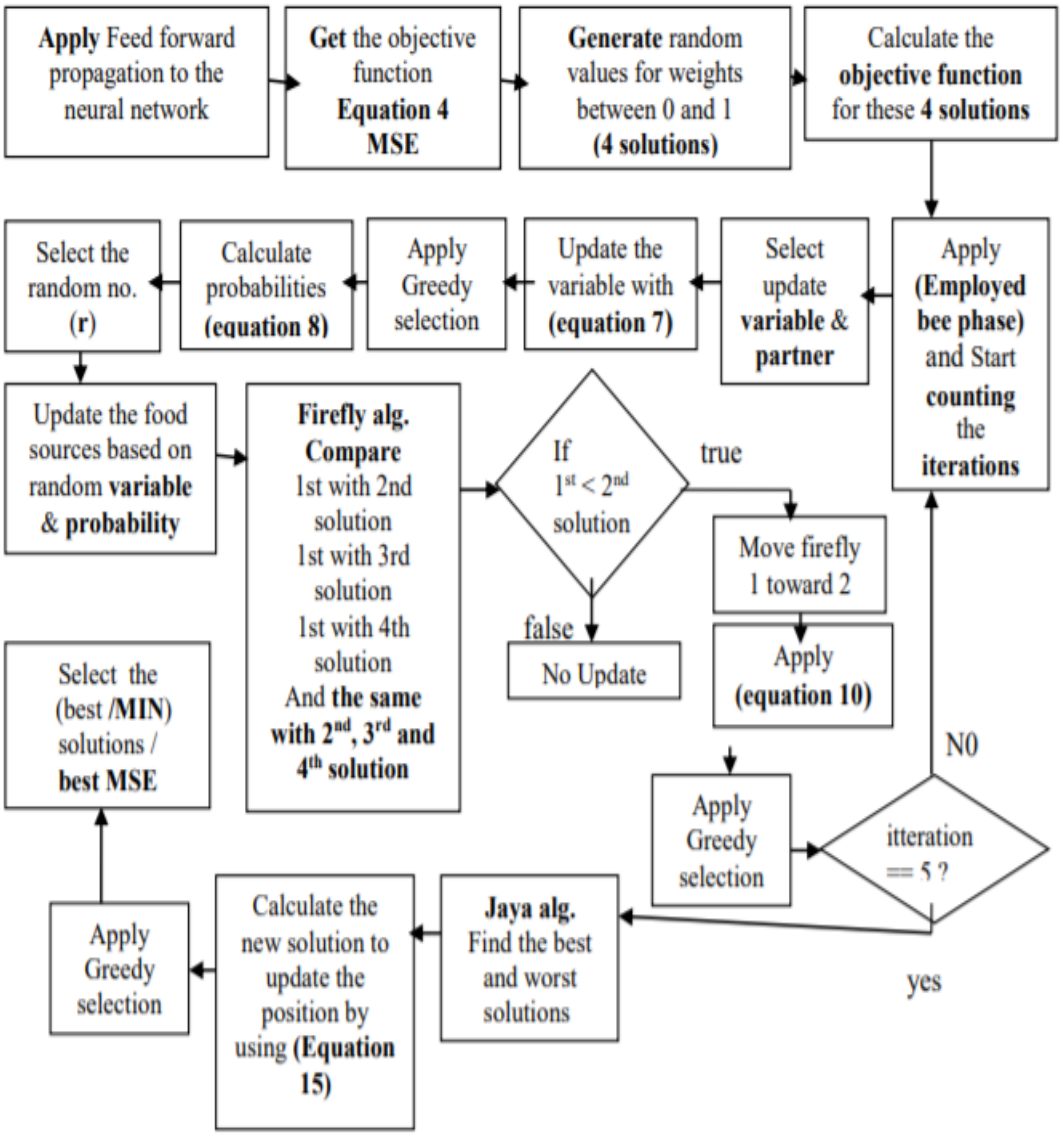


Figure 9: Hybrid AFJ algorithm

In the fifth model, a hybrid algorithm is designed based on firefly algorithm and Jaya algorithm. It is called the JF algorithm, referring to the first letter of each algorithm. Initially, the parameters were initialized, and 4 random solutions were imposed for the values of weights between zero and one. Then equation 4 is applied to calculate the objective function for these four solutions. To optimize the weights, the Jaya algorithm was used in the beginning. It selects the best value and the worst value from these four solutions that were found. Then, using equation 15, new and improved weights can be obtained, and we test them with greedy selection. This process is done ten times until we reach the best possible improvement. The final optimized solutions obtained from Jaya algorithm are used as parameters in Firefly algorithm. In the Firefly algorithm, each solution is compared with the other three solutions, so if it needs optimize, meaning that if its value is higher than the other values, it is optimized by applying equation 10. Because the goal is to reach the lowest possible value. Then greedy selection is applied to test the new values obtained. Finally, the best solution is chosen from the four optimized solutions. Figure 10 shows the hybrid JF algorithm.

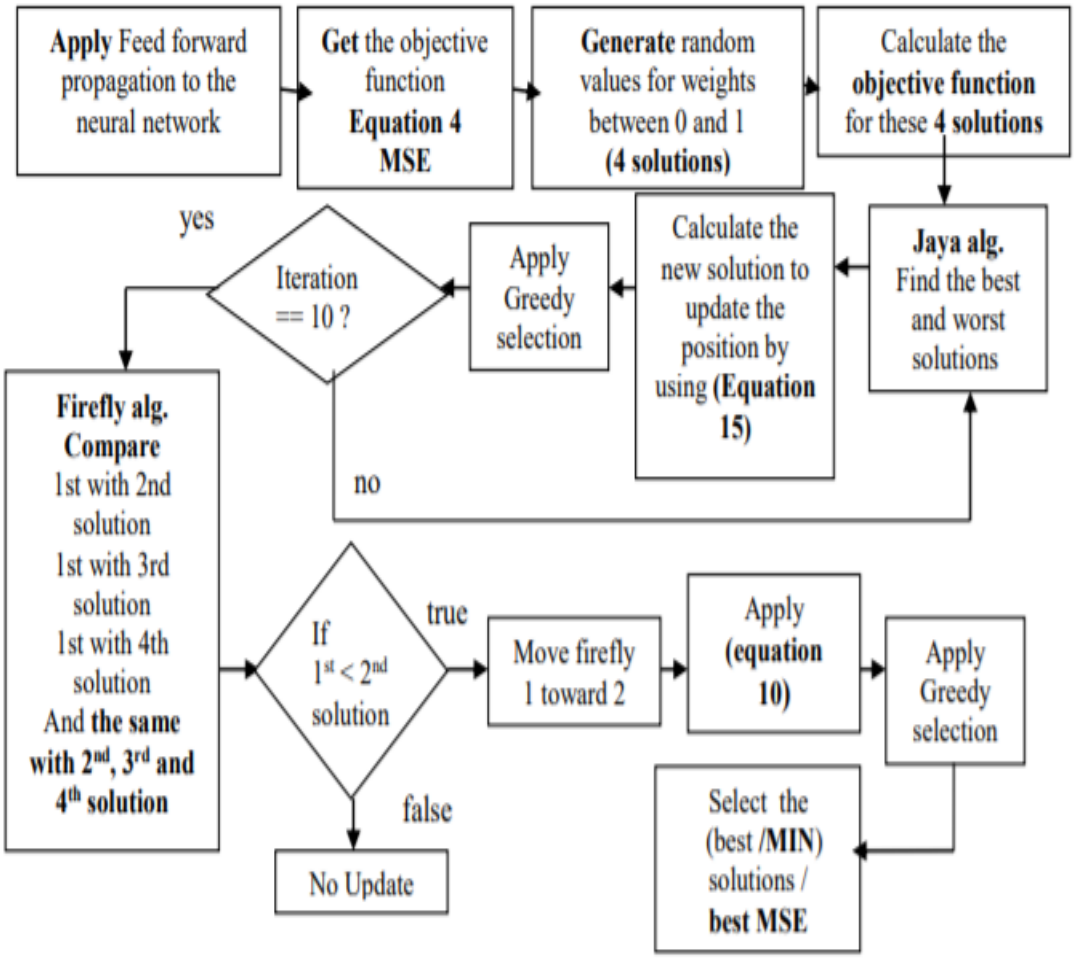


Figure 10: Hybrid JF algorithm

In the sixth model, a hybrid algorithm is designed based on genetic algorithm, ABC algorithm, Jaya algorithm and firefly algorithm. It is called the GAJF algorithm, referring to the first letter of each algorithm. Initially, the parameters were initialized, and 5 random solutions were imposed for the values of weights between zero and one. Then equation 4 is applied to calculate the objective function for these five solutions. To optimize the weights values, a genetic algorithm was used in the beginning, as it selects the best 3 values from these five solutions that were found. After that, the crossover and mutation process take place between the first and second solutions, the third and second solutions, and the first and third solutions. Thus, three new solutions resulting from the genetic algorithm are added to the previous three solutions. The best three solutions are selected, and the same process is repeated 18 times in order to reach the best possible improvement. From the last six solutions obtained the four best solutions are used as parameters of the ABC algorithm. In the second stage, the first and second stage of the ABC algorithm are used. Respectively, the Jaya algorithm and the firefly algorithm are used in one iteration and iterated 18 times in order to reach the best possible optimize. Figure 11 shows the hybrid GAJF algorithm.

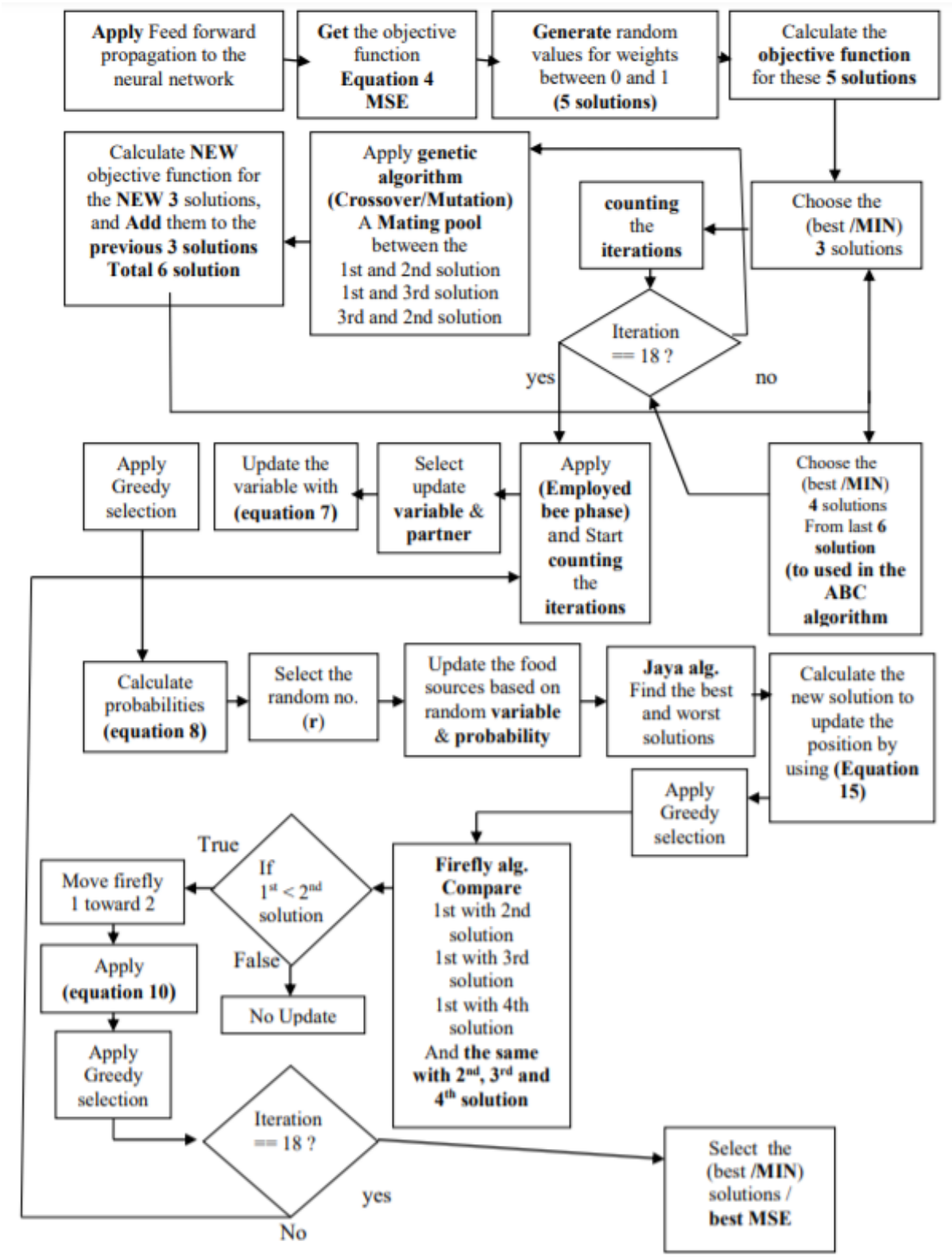


Figure 11: Hybrid GAJF algorithm

Results

The proposed hybrid algorithms are designed to train deep neural networks for network intrusion detection. In this section the performance and effectiveness of these algorithms are evaluated based on the NSL-KDD test dataset. To evaluate the samples a laptop was used running on a Windows10 64bit operating system with a Core i7 processor and 16GB RAM.

Performance metrics that were used to evaluate the proposed algorithms

To evaluate the performance of the proposed algorithms, several metrics were used, where MSE values were used, and accuracy was calculated for each model. We also calculated the confusion matrix, precision, detection rate DR, false alarm rate FAR, specificity, sensitivity, and sensitivity area under the curve AUC. The following is an explanation of the calculation mechanism for each of them.

After the deep neural network weights are optimized and trained by the proposed hybrid algorithms. These weights were used in addition to the new entries from the NSL-KDD test data set to implement the loss function or MSE objective function referred to in equation 4. The closer the MSE value is to zero, the better the performance. The proposed algorithms were able to reduce the error value as shown in Table 3. While training the models, the MSE loss rate for the proposed JF algorithm was 9.93325e-05 and 9.999899e-05 for the GAJF algorithm. As we can see, the loss rate is very small and approaches zero, and the more MSE value approaches zero, the better the performance. In other hybrid algorithms, the MSE value for GAF was 0.11773, 0.14276 for AFJ algorithm, 0.15147 for AF algorithm, and 0.17897 for proposed GF algorithm. After the samples were tested on Test dataset, the JF and GAJF algorithm also achieved a low loss rate as the MSE loss rate for the JF algorithm was 9.99868e-05 and 9.99854e-05 for the GAJF algorithm. As we can see, the two algorithms also performed well. And MSE for GAF was 0.11843, 0.16025 for AF, 0.1971 for AFJ, and 0.21574 for GF. Note that hybrid models when incorporated with Jaya algorithm achieve high performance.

Table 3: Results of MSE values for hybrid algorithms for train and test dataset

Name of the algorithm	MSEresult for Train dataset	MSE result for Test dataset
GAF	0.11773	0.11843
AFJ	0.14276	0.1971
AF	0.15147	0.16025
GF	0.17897	0.21574
JF	9.93325e-05	9.99868e-05
GAJF	9.999899e-05	9.99854e-05

This type of evaluation evaluates the overall performance of the algorithm and shows us the amount of loss or error rate between the true values and the values that the algorithm predicted. As we can see from Table 3, the amount of error is small because it is close to zero. Meaning that the algorithms have succeeded in optimization. But if we want to know the number of predictions that have been classified correctly or incorrectly, we will use another type of evaluation. So, the confusion matrix was used. Before doing this, we need to choose the cutoff point and determine its value. Then all the values that the algorithm predicts if they are above the cutoff point are converted to 1 and anything below is equal to 0. Then it is compared with the real values which classify intrusion on the network as 1 and classify the normal network traffic as 0. The cut-off point was chosen by averaging all the values that the algorithm had predicted and then multiplying them with incremental values and testing them as shown in equation 16. Then, by trial and error, we arrive at the best cutoff point.

$$\text{Cutoff point} = \sum(\frac{yi}{n}) * \text{random} \quad (16)$$

Where yi represents the predicted values and n represents the sum of samples in the database. Through trial and error, we change the random value until get the best cutoff point. FAR, DR, and ACC are calculated based on the number of true positives TP, false positives FP, true negatives TN, and false negatives FN. Table 4 shows these values for the hybrid algorithm. And formulas (17-22) explain how to calculate it mathematically.

$$\text{ACC} = \frac{TP+TN}{TP+TN+FP+FN} \quad (17)$$

$$\text{FAR} = \frac{FP}{TN+FP} \quad (18)$$

$$\text{DR} = \frac{TP}{TP+FN} \quad (19)$$

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (20)$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (21)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (22)$$

Table 4: TP, TN, FP and FN computation for the six hybrid algorithms

Name of the algorithm	TP	TN	FP	FN
GAF	12132	6117	2570	1724
AFJ	13300	5879	2808	556
AF	12037	6299	2388	1819
GF	12021	6358	2329	1835
JF	10324	7074	1613	3532
GAJF	7036	8532	155	6820

After comparing the real values with the predicted values, we can calculate ACC accuracy, DR detection rate, FAR, sensitivity and specificity as shown in Table 5. Most of the proposed hybrid algorithms achieved good accuracy, as the accuracy of AFJ algorithm was 85.077%, 81,529% for GF algorithm, 80.951% for GAF algorithm, 81,338% for AF algorithm, and 77.177% for JF algorithm, while the accuracy of GAJF algorithm was 69.095%. SE and SP in Table 5 refer to the sensitivity that measures the true positive rate and the specificity that measures the true negative rate. In order to obtain good results, the sensitivity and specificity should be as high as possible. Meaning that if the specificity of the test is 95%, then the rate of false negatives is 5%, and if the sensitivity is 98%, then false positives are 2%. And False alarm rate FAR indicates the false positive rate. It is the ratio between the number of attacks detected by the system to the number of actual attacks present in the dataset**.** Accuracy is a tool for calculating the ratio of correct expectations to all positive expectations. As shown in Table 5, the GAJF algorithm achieved the highest percentage of 97%, which is an excellent percentage, while the same algorithm achieved a lower accuracy of 69%, and this indicates that using one metric to calculate the accuracy of the model does not It is enough.

Table 5: Results of performance measurements of 6 hybrid algorithms

Name of the algorithm	ACC %	Precision%	DR%	FAR%	SP. %	SE. %
GAF	80.951	82.52	87.56	29.58	70.42	87.56
AFJ	85.077	82.57	95.99	32.32	67.68	95.99
AF	81.338	83.45	86.87	27.49	72.51	86.87
GF	81.529	83.77	86.76	26.81	73.19	86.76
JF	77.177	86.49	74.51	18.57	81.43	74.51
GAJF	69.095	97.84	50.78	1.784	98.22	50.78

We also used another metric to evaluate the performance of the proposed models, which is area under the curve. The area under the curve (AUC) measures the ability of the proposed models to classify. The higher the AUC value, the better the model performance. Where if the AUC value is equal to 0.5, then this means that the model is bad and unable to be classified, and the AUC value of 0.7 to 0.8 is considered good, and from 0.8 to 0.9 is excellent, and more than 0.9 is very excellent in classification. Table 6 refer to AUC values and figure 12 refer to the best AUC obtained. The AUC for GAF was 0.7899, 0.8183 for AFJ algorithm, 0.7969 for AF algorithm, 0.8068 for GF algorithm, 0.7797 for JF algorithm, and 0.7450 for GAJF algorithm. Compared with Figure 13, the AUC of the normal deep learning algorithm that uses backpropagation has achieved a value of 0.7867.

Table 6: Area under the curve (AUC)

Name of the algorithm	AUC
GAF	0.7899
AFJ	0.8183
AF	0.7969
GF	0.8068
JF	0.7797
GAJF	0.7450

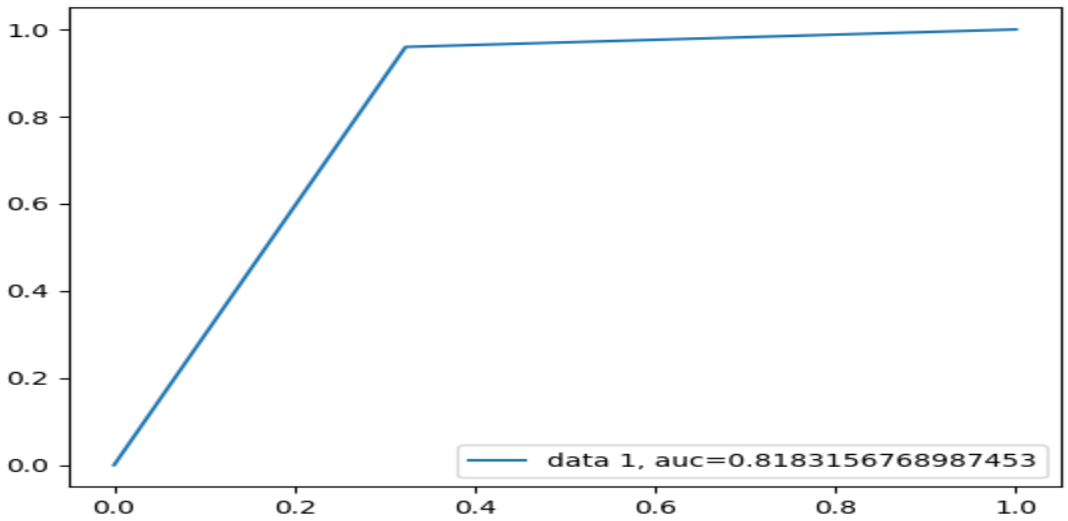


Figure 12: Area under the curve AUC for AFJ algorithm

Comparing the proposed models with the deep neural network

As mentioned earlier, the main idea of the proposed models is to improve the performance of the deep neural network by replacing backpropagation with one of the six hybrid algorithms designed with the goal of giving better results than the deep neural network. So, in this part we will compare the performance of the proposed models and the usual deep neural network. We used the same neural network architecture proposed in Figure 1 but with backpropagation for learning rather than using the proposed algorithms as shown in equation 5 and 6**.**

Table 7: TP, TN, FP and FN computation for DNN

Name of the algorithm	TP	TN	FP	FN
DNN + backpropagation	11046	6743	1944	2810

Table 7 shows the number of TP true positives, FP false positives, TN true negatives and FN false negatives for deep neural network. Then we apply equations (17-22) to evaluate the performance of the algorithm as shown in Table 8.

Table 8: Results of performance measurements of DNN

Name of the algorithm	ACC %	Precision%	DR%	FAR%	SP. %	SE. %
DNN+ back propagation	78.91	85.03	79.72	22.38	77.62	79.72

When comparing table 5 with table 8, we see that the deep neural network achieved 78.91% accuracy, while GAF algorithm, AFJ algorithm, AF algorithm and GF algorithm achieved higher accuracy. In addition, when comparing the values of Precision, we find that the JF algorithm and the GAJF algorithm outperformed the neural network, but the neural network algorithm achieved Precision higher compared to the other four algorithms. Likewise, when compared to other metrics as shown in table 8, we find that some of the proposed algorithms have outperformed the neural network. This means that the proposed models can be certified as a developer of neural networks for better results.

Table 9: Area under the curve (AUC)

Name of the algorithm	AUC
DNN+ back propagation	0.7867

Also, when calculating the area under the curve as shown in figure 13, the AUC value is 0.7867, which is a good percentage, but when we compare the neural network model with the proposed models, we see that the GAF algorithm, AFJ algorithm, AF algorithm and GF algorithm have outperformed the neural network as shown in table 6 and table 9. This means that the proposed models gave better results than the deep neural network, and the proposed algorithms proved effective in developing the deep neural network to get better results.

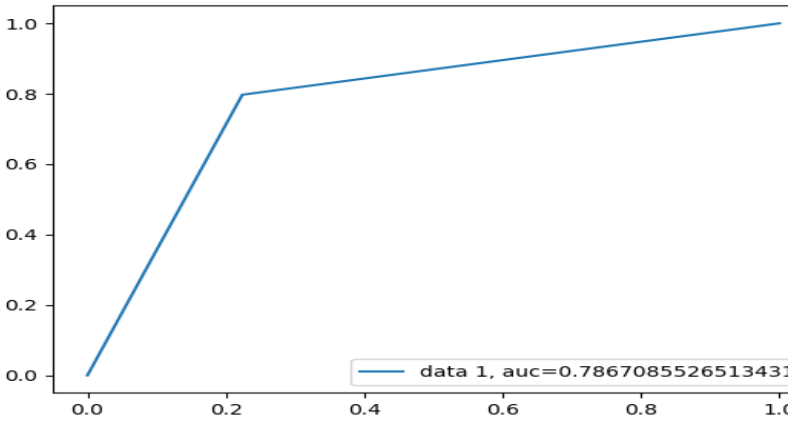


Figure 13: Area under the curve AUC for DNN

Comparing the results of the proposed models with previous studies

The following is a table to compare the algorithms used in the previous researches and the algorithms proposed in this paper. Table 10 shows the metrics used to evaluate these algorithms.

Table 10: Comparing the proposed models with previous studies

Algorithm name:	ACC%	Precision%	SP. %	SE. %	FAR
GAF	80.951	82.52	70.42	87.56	29.58
AFJ	85.077	82.57	67.68	95.99	33.32
AF	81.338	83.45	72.51	86.87	27.49
GF	81.529	83.77	73.19	86.76	26.81
JF	77.177	86.49	81.43	74.51	18.57
GAJF	69.095	97.84	98.22	50.78	1.784
XGBOOST	95.55	92	-	98	-
SVM	94	95.92	96.15	94	3.85
DNN with 6 SDN features	75.9	-	-	-	10.8
SAAE-DNN	87.47	86.47	67.89	-	-
HAD	91.7	92	89	94	-
DNN in SDN environment	75.75	-	-	-	-
CART	83.66	-	-	85.7	-
STL – 5 class classification	79.10	96.56	-	95.95	-
Auto-eencoder + SVM model	92	-	-	-	-
Deep learning in IoT	96.3	-	-	-	-
MGA-SVM-HGS-PSO-ANN	95.2	97.5	-	96.3	-
1-class SVM+C5.O DT classifier	83.24	-	-	-	-
HWOAFPA	99.07	-	-	-	-
Stacking model	84.32	-	69.51	87.60	-
Random forest+MLP	70.81	-	-	87.68	30.48

As we can see in table 10, the proposed algorithms in the first 6 lines are compared with the rest of the algorithms used in previous studies. We see that the comparison is made by using several metrics to evaluate the performance of algorithms such as Accuracy, Precision, FAR, Specificity, and Sensitivity. Some of these metrics were used in previous studies and some were not as we can see in table 10. When comparing the proposed algorithms with other algorithms based on accuracy, we note that some of the proposed algorithms outperformed some of the algorithms such as DNN with 6 SDN features, DNN in SDN environment, CART algorithm, STL-5 class classification, 1-class SVM+C5.O DT classifier, Stacking model and Random forest+MLP, while some other

algorithms outperformed the proposed algorithms and achieved higher accuracy. We note that the overall performance of the algorithms was very good and achieved high accuracy. When evaluating the algorithms on the Precision metric, we see that our proposed GAJF algorithm has achieved the highest percentage compared to other algorithms. As well as for the measures of Specificity and Sensitivity, our proposed algorithm achieved a high percentage as well. Through our comparison of the deep neural network with the proposed models that aim to develop the neural network, and by comparing them with previous studies, we note that the proposed models have contributed to finding a high-performance methodology for detecting intrusion on the network and contributed to proving that the hybrid algorithms used are effective and help to the optimization can be used in other studies.

The result of achieving the additional contribution in this paper

This paper aims to find an effective methodology that can detect the attack on the network, and in order to do this it is proposed to contribute to the development of the deep neural network by building 6 models based on hybrid algorithms. Then the six models are compared and the best one is selected by comparing it with the normal neural network and previous research. When comparing the results of the neural network as shown in table 8 and 9 with the results of the proposed models for the development of the neural network, as we see in table 5 and 6, we see that the proposed models were able to improve the neural network and obtained better results, and thus these models contributed to finding a better methodology that can be adopted in other research In addition, the six hybrid algorithms that were used to develop the neural network have proven effective and can also be used in other research as optimization algorithms. Thus, by looking at the results, we see that the proposed methodology achieved good results in general compared to other studies.

Discussion

By watching the results, we notice that the hybrid algorithms that contain the Jaya algorithm achieve high performance in the optimization process. On the contrary, when the genetic algorithm is used, which is slow to improve and may reach a solution that cannot be improved afterwards. When calculating the accuracy of the model, a correct and appropriate cutoff point must be chosen so that we can know the number of correct and false predictions. In this paper we were able to reach a good cutoff point for most of the proposed hybrid algorithms. But its performance should have been better than that, for example, looking at the GAJF algorithm, the performance of the algorithm was high, and the loss rate was very little and approaching zero, as the MSE value reached 9.99854e-05 and its performance was better compared to other algorithms. And MSE for AF algorithm was 0.16025. While when calculating the accuracy that depends on finding the best cut-off point, the performance of the AF algorithm was higher, as its accuracy reached 81.338%, while the accuracy of the GAJF algorithm was 69.095%. This means that we were able to reach a cutoff point suitable for the AF algorithm and we were unable to reach the best cutoff point for the GAJF algorithm. In any case, the accuracy of all the algorithms was high and we recommend in future research to find a mechanism to reach the best possible cut-off point. Compared to the previous researches that was discussed.

Conclusion

This study aims to detect attacks on the network, for that the deep neural network was developed, where the backpropagation stage in the neural network was replaced by hybrid optimization algorithms. This paper contributed to finding other methods and mechanisms to improve the multi-layered neural network by updating its weights using six hybrid algorithms that were designed based on four other known optimization algorithms, which are the genetic algorithm, the firefly algorithm, the ABC algorithm, and the Jaya algorithm. These proposed models are a network intrusion detection methodology in which the proposed neural network was trained on an NSL KDD database that monitors network traffic. After that, the six proposed algorithms were

compared and the AFJ algorithm achieving the highest accuracy and performance. Several metrics were used to evaluate the performance of the proposed models. The results show that the proposed six hybrid algorithms performed well and helped in training the deep neural network. And Principal component analysis technique was used for feature selection. Where the proposed algorithms performed well with accuracy ranging from 77.177% to 85.077%. Whereas the performance of hybrid algorithms containing Jaya algorithm and firefly algorithm was higher than hybrid algorithms containing genetic algorithm. The proposed algorithms also achieved a low MSE loss rate close to zero with an MSE value of 9.99868e-05. Also, precision values range from 82 to 97%, and most of the proposed algorithms achieved high rates when calculating the sensitivity and specificity. We note that the proposed models have contributed to finding a high-performance methodology for detecting intrusion on the network and also contributed to proving that the hybrid algorithms used are effective and help to the optimization can be used in other studies. In future research and work I strongly recommend working towards finding a suitable and correct cutoff point with using the same algorithms because, as we can see in the results of the JF algorithm, for example, the performance of the algorithm was very high and the MSE loss rate was close to zero, while its accuracy rate was 77.177% due to the lack of a precise cut-off point. Compared to other models, the cut points were good, good accuracy was obtained, and could be improved in the future.

References

Alzahrani, A. & Alenazi, M. (2021). Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. MDPI, 13(111), 1-18. https://doi.org/10.3390/fi13050111

Anderson, J. (1980). Computer Security Threat Monitoring and Surveillance. Fort Washington, 1-56.

Bachar, A., Makhfi, N. & Bannay, O. (2020). Machine Learning for Network Intrusion Detection Based on SVM Binary Classification Model. Advances in Science, Technology and Engineering Systems Journal, 5(4), 638-644. https://doi.org/10.25046/aj050476

Choudharya, S. & Kesswanib, N. (2020). Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. International Conference on Computational Intelligence and Data Science, 9(10), 167 (2020), 1561-1573. https://doi.org/10.1016/j.procs.2020.03.367

Denning, D. (1987). An intrusion-detection model. IEEE Transactions on Software Engineering, 13(2), 222-232

Dhanabal, L. & Shantharajah, S. (2015). A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. International Journal of Advanced Research in Computer and Communication Engineering,4(6), 446-452. https://doi.org/10.17148/IJARCCE.2015.4696

Ghanem, H., Jantan, A., Ghaleb, S. & Nasser, A. (2020). An Efficient Intrusion Detection Model Based on Hybridization of Artificial Bee Colony and Dragonfly Algorithms for Training Multilayer Perceptrons. IEEE Access, 8(2020), 130452-130475. https://doi.org/10.1109/ACCESS.2020.3009533

Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J., Bayne, E. & Bellekens, X. (2020). Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. MDPI Journal, 9(10), 1-16. https://doi.org/10.3390/electronics9101684

Hosseini, S. & Zade, B. (2020). New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN. Computer Networks, 173(2020), 107-168. https://doi.org/10.1016/j.comnet.2020.107168

Ingre, B., Yadav, A. & Soni, A. (2017). Decision Tree Based Intrusion Detection System for NSLKDD Dataset. Smart Innovation, 2(84), 207-218. https://doi.org/10.1007/978-3-319-63645-0_23

Istiaque, S., Khan, A., Hassan, Z. & Waheed, S. (2021). Performance Evaluation of a Smart Intrusion Detection System (IDS) Model, European Journal of Engineering and Technology Research, 6(2), 148-152. https://doi.org/10.24018/ejers.2021.6.2.2371

Karaboga, D. & Basturk, B. (2007). Artificial Bee Colony (ABC) OptimizationAlgorithm for Solving Constrained OptimizationProblems. Lecture Notes in Computer Science, 4529, 789-798. https://doi.org/10.1007/978-3-540-72950-1_77

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J. & Alazab, A. (2020). Hybrid intrusion detection system based on the stacking ensemble of C5 decision tree classifier and one class support vector machine, MDPI, 9(1), 173-191. https://doi.org/10.3390/electronics9010173

Mohmmadzadeh, H. & Gharehchopogh, F. (2020). A novel hybrid whale optimization algorithm with flower pollination algorithm for feature selection: Case study Email spam detection, Preprints, 1-28. https://doi.org/10.1111/coin.12397

Niyaz, Q., Sun, W., Javaid, A. & Alam, M. (2016). A Deep Learning Approach for Network Intrusion Detection System. EAI Endorsed Trans. Security Safety, 16(9), 1-6. https://doi.org/10.4108/eai.3-12-2015.2262516

Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7(2016), 19-34. https://doi.org/10.5267/j.ijiec.2015.8.004

Rawat, S., Srinivasan, A. & Ravi, V. (2020). Intrusion detection systems using classical machine learning techniques versus integrated unsupervised feature learning and deep neural network. InternetTechnology Letters, e232, 1-9. https://doi.org/10.1002/itl2.232

Reeves, C. (2010). Genetic Algorithms. Handbook of Metaheuristics, 109-139. https://doi.org/10.1007/978-1-4419-1665-5

Tang, C., Luktarhan, N. & Zhao, Y. (2020). SAAE-DNN: Deep Learning Method on Intrusion Detection. MDPI, 12, (1695), 1-20. https://doi.org/10.3390/sym12101695

Tang, T., Mhamdi, L., McLernon, D., Zaidi, S & Ghogho, M. (2016). Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. IEEE. https://doi.org/10.1109/WINCOM.2016.7777224

Ucar, M., Ucar, E. & Incetas, M. (2021). A Stacking Ensemble Learning Approach for Intrusion Detection System, Düzce University Journal of Science & Technology, 9(2021), 1329-1341. https://doi.org/10.29130/dubited.737211

Yang, X. & He, X. (2013). Firefly algorithm: recent advances and applications. International Journal of Swarm Intelligence, 1(1), 36-50. https://doi.org/10.1504/IJSI.2013.055801