Bachelor of Science in Computer Science & Engineering

Genetic Algorithm-based Optimal Deep Neural Network for Detecting Network Intrusions

by

Sourav Adhikary

ID: 1504020

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

April, 2021

Genetic Algorithm-based Optimal Deep Neural Network for Detecting Network Intrusions

Submitted in partial fulfilment of the requirements for
Degree of Bachelor of Science
in Computer Science & Engineering

by

Sourav Adhikary
ID: 1504020

Supervised by

Dr. Iqbal H. Sarker
Assistant Professor
Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)
Chattogram-4349, Bangladesh.

The thesis titled '**Genetic Algorithm-based Optimal Deep Neural Network for Detecting Network Intrusions'** submitted by ID: 1504020, Session 2019-2020 has been accepted as satisfactory in fulfilment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

Declaration of Originality

This is to certify that I am the sole author of this thesis and that neither any part of this thesis nor the whole of the thesis has been submitted for a degree to any other institution.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I am also aware that if any infringement of anyone's copyright is found, whether intentional or otherwise, I may be subject to legal and disciplinary action determined by Dept. of CSE, CUET.

I hereby assign every rights in the copyright of this thesis work to Dept. of CSE, CUET, who shall be the owner of the copyright of this work and any reproduction or use in any form or by any means whatsoever is prohibited without the consent of Dept. of CSE, CUET.

Signature of the candidate

Date:

Acknowledgements

The satisfaction that accompanies the successful completion of this work would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I am grateful to my honorable project Supervisor **Dr. Iqbal H. Sarker**,Assistant Professor, Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, for the guidance, inspiration and constructive suggestions which were helpful in the preparation of this project. I would also like to convey my gratitude towards our head of the department **Dr. Asaduzzaman**. I also convey special thanks and gratitude to all my respected teachers of the department. I would also like to thank friends and the staffs of the department for their valuable suggestion and assistance that has helped in successful completion of the project. Finally, I would like to thank my family for their steady love and support during my study period.

Abstract

Computer network attacks are evolving in parallel with the evolution of hardware and neural network architecture. One of the most significant risks to cyberspace is network intrusion. Despite major advancements in Network Intrusion Detection System(NIDS) technology, most implementations still depend on signature-based intrusion prevention, which can't identify unknown assaults. Deep learning can help NIDS detect novel threats since it has a strong generalization ability. The deep neural network's architecture has a significant impact on the model's results. To solve the problem of neural network architecture design, we propose a Genetic Algorithm to find the optimum no of hidden layers and the no of neurons in each layer of the Deep Neural Network(DNN) architecture for the intrusion detection binary classification problem. With the novel fitness function, the genetic algorithm converged into an optimal DNN architecture. The proposed DNN architecture shows better result than classical machine learning algorithms at a low computational cost.

Keywords— intrusion detection, neural network architecture, genetic algorithm,fitness function, deep neural network

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

A computer network is a set of computers for sharing resources. The Internet is the biggest network for sharing resources. About 4.57 billion people are active internet users [1]. With the popularization of the internet, Cybercrime is also increasing. Globally cyber attack damage costs will reach 6 trillion annually by 2021 [2]. A data breach cost is 3.86 million on average [3].

Attacks on computer networks are evolving with the advancement of hardware and network topologies. Network intrusion is one of the biggest threats to cyberspace. Network intrusion detection detects any unauthorized access to the computer network. There are two types of Network Intrusion Detection System (NIDS). They are signature-based and anomaly-based intrusion detection systems. Signature-based intrusion detection uses pattern matching techniques of known attacks. With increasing, cyberattack new unknown attacks are emerging. A signature-based system can not detect unknown attacks which makes it inefficient. An anomaly-based NIDS uses machine learning techniques to analyze and learn the normal network behavior of a system. Then it can be used to detect deviations from normal traffic behavior.

1.2 Motivation

Network intruders are using a new technique to attack the network. The current Network Intrusion Detection System (NIDS) is not optimistic. Despite the significant advances of the NIDS technology, most solutions operate on signature-based intrusion detection, which is incapable of detecting unknown attacks. With a good generalization ability, deep learning can enable NIDS to detect unknown attacks. Intrusion detection is an important research area in the field of network security. In recent years, one of the main focuses of NIDS research has been the application of machine learning and deep learning techniques [4]. Deep learning approaches show better results than conventional machine learning algorithms [5][6]. Based on deep learning models, the most common issues in the present solutions are; firstly, the models produce high false-positive rate with multiple labels of attacks; secondly, the models are not generalized, as an existing solution have mainly used only a single data set to report the performance of the deep learning model and lastly, most deep learning architectures in the literature are selected using trial and error method which is prone to error.

1.3 Design Overview

To overcome the neural network architecture selection problem, we have used genetic algorithm to find optimal neural architecture for intrusion detection problem. A custom fitness function is used for neural architecture search. Two selection process; rank selection and tournament selection are combined in the architecture selection process of the genetic algorithm to get better results. In crossover method of the genetic algorithm, two point crossover is used. The best neural architecture found in the search process is used to evaluate performance and compare with other classical machine learning algorithm.

1.4 Difficulties

Neural network architecture search is a computationally costly process. Limited hardware resource is one of the main difficulties we have faced. Imbalance dataset of intrusion detection is another problem we have faced. We have solved the imbalance problem using under sampling the dataset.

1.5 Applications

Over the past decade, the use of web-based and IoT-based technologies has skyrocketed. Security is a big concern for the huge stream of information used in everyday. Network Intrusion Detection System can be used to protect information from intruder. Our proposed deep learning model can be used in Network Intrusion Detection System.

1.6 Contribution of the thesis

We have proposed an anomaly-based intrusion detection deep learning model. To find the optimal network architectures genetic algorithm is used. Contributions of this paper are,

1. Finding optimal no of hidden layer and no of neurons in each hidden layers for for intrusion detection binary classification problem.
2. An novel deep learning model for intrusion detection classification problem.

1.7 Thesis Organization

The rest of this report is organized as follows:

- Chapter 2 gives a brief summary of previous research works in the field of network intrusion detection and neural architecture search.
- Chapter 3 gives description of proposed methodology. In the proposed methodology , a genetic algorithm with novel fitness function is used to find optimal neural architecture components; no of hidden layer and no neurons in each layer.
- Chapter 4 provides the description of the working dataset and analysis of the performance measure for the proposed methodology.
- Chapter 5 contains the overall summary of this thesis work and provides some future recommendations as well.

1.8 Conclusion

An analysis of genetic algorithm-based neural architecture search for network intrusion detection is presented in this chapter. This chapter discusses the project's inspiration as well as its challenges. Contribution and the application of the proposed system is described in this chapter. In the next chapter, background and present state of the problem will be provided.

Chapter 2

Literature Review

2.1 Introduction

We presented an outline of genetic algorithm-based optimal neural network architecture for intrusion detection in the previous chapter. The previous research on intrusion detection is described in this chapter. This chapter also covers research on neural architecture search.

2.2 Related Literature Review

Research on Cybersecurity gains popularity in recent years. Many researchers using machine learning techniques for anomaly-based network intrusion detection.

2.2.1 Network Intrusion Detection

A systematic mapping research to compare publications on Intrusion Detection Systems with Deep Learning discovered that Deep Neural Network is the most favoured deep learning algorithm of all deep learning algorithms [7].

Vinayakumar et al. [8] showed a comprehensive analysis of DNN and other machine learning techniques using publicly available datasets. DNN model shows better performance than other machine learning techniques. The authors select the DNN topology using the trial and error method from a small set of architecture search spaces.

A convolution neural network is proposed by [9], where network intrusion detection problems are converted into an image classification problem. Intrusion data is used to construct a grayscale image where CNN is used. The hyperparameter is found using a comparative analysis of different settings. The authors also claimed better performance of their proposed model than other machine learning techniques on KDDcup99 and NSL-KDD datasets.

Nathan Shone et al. [10] proposed a Non-Symmetric Deep Auto Encoder (NADE) for unsupervised feature learning. The authors used stacked NADEs for feature learning of intrusion data and used the Random Forest algorithm as a classifier. The authors claim some promising results on KDDcup99 and NSL-KDD datasets. M.A. Ferrag et al. [5] presented a detailed analysis of seven deep learning techniques on the CSE-CIC-IDS2018 data set with three performance indicators, namely detection rate, accuracy, false alarm rate. The deep neural network performed best among other deep discriminative models. In [11], Sarker et al. reduced feature dimensions by estimating feature importance and ranking them. The authors presented an Intrusion Detection Tree based on important features. Their multi-level IntruDTree shows better performance than traditional machine learning classification-based methods. While the final dense neural network served as the supervised classifier, an autoencoder facilitated an unsupervised pre-training on the data to provide compressed and less noisy representations of the input space in [12].To test the performance of a Network Intrusion Detection System in various adversarial contexts, a modular Evaluating Network Intrusion Detection architecture is proposed in [13]

Connected cars in smart cities communicate with one another.Aloqaily et al. proposes a three-phase intrusion prevention mechanism for linked cars [14].

A probabilistic data structures and Deep Learning intrusion detection system focusing on the extraction of control traffic functions in real time [15]. Papamartzivanos has suggested a novel approach that blends the advantages of self-training and MAPE-K framing in order to provide a flexible and autonomous misuse System for Intrusion Detection [16].

In [17],authors suggested a deep learning flow-based anomaly detection method. The NIDS model is used in an SDN context by a deep neural network (DNN). An optimal hyper-parameter for DNN is found through trial and error methods. In order to choose both subset of features and hyperparameters in one operation,

[18] proposed a double algorithm based on particle swarm optimization (PSO). In the pre-training stage, the above-mentioned algorithm is used to pick automated optimized functionality and model hyperparameters. The authors [19] suggested an adaptive database intrusion detection model based on evolutionary reinforcement learning (ERL), which combines reinforcement learning, a learning process of a human, with evolution learning, a learning process of a community, to look for the optimal model with the characteristics of each system. The intrusion is detected by the behavior network, and the appraisal network gives updates to the behavior network's identification. The weights of the networks are encrypted as individuals in order to determine the best model to produce better people over decades.

2.2.2 Neural Architecture Search(NAS)

The architecture of the deep neural network greatly influences the performance of the model. As mentioned above many researchers use the trial and error method to find an optimal network architecture that needs extensive knowledge of deep learning and also prone to error. In [20],authors discussed different Network Architecture Search (NAS) methods. The aim of NAS is usually to identify architectures that can forecast data that hasn't been seen before [21]. Baker et al. [22] used a metamodeling technique focused on reinforcement learning to simplify the process of CNN architecture selection. The authors created a novel Q-learning agent with the aim of discovering CNN architectures that work well on a given machine learning challenge without the human intervention.

An algorithm for optimizing a multilayered artificial neural network is proposed in [23]. It prunes neurons from the hidden layers as much as possible while retaining the same error rate.

Starting from an overly large network, pruning algorithms remove unimportant neurons before the optimum network emerges. Based on the backpropagation algorithm, the paper [24] proposes a pruning algorithm for determining the best topology for hidden layers in neural networks.

Yann et al.[25] introduced Optimal Brain Damage (OBD), a modern method for shrinking a learning network through selectively deleting weights, and demonstrated that OBD may be used as an automated network minimization protocol as well as an interactive method to propose alternative architectures.
In [26], Bergstra et al. shows trials on a grid are less effective for hyper-parameter optimization than trials chosen at random. Within a fraction of the computing time, random search over a domain will find models that are as good as or better than the original. Random search seeks better models by efficiently scanning a wider, less promising configuration space while given the same computational budget.

D. STATHAKIS [27] proposed a genetic algorithm to find an optimal network topology and showed comparative analysis with other NAS algorithms. A hybrid genetic algorithm with a stochastic layer shows promising results in affordable computation cost [6]. In [28], a modified evolutionary strategy is used to find optimal architecture for retinal vessel segmentation. In the world of automation, it is important to automate the architecture search. More research is needed to find a Network Architecture Search (NAS) algorithm that will provide an optimal solution at a reasonable computation cost. In this paper, genetic algorithm is used for Network Architecture Search (NAS).

2.3 Conclusion

A thorough literature review is explored in this chapter. This discussion has been split into sub-sections for the convenience. The researchers' intrusion detection and neural architecture search algorithms are listed in this paper. The next chapter contains the detailed explanation of the proposed methodology of optimal neural architecture for network intrusion binary classification problem.

Chapter 3

Methodology

3.1 Introduction

Many companies now need network intrusion detection systems (NIDS). To ensure the safety and protection of their communication and data, such systems track network traffic and detect suspicious activities or cyber attacks. As a computational model, we employ an artificial neural network, since it has the effect of incorporating intelligence into our proposed system by using biological neural network characteristics.

3.1.1 Artificial neural network(ANN)

The input layer, hidden layer, and output layer make up an artificial neural network (ANN). Artificial neural networks have connected nodes called neurons that pass information from one layer to another layer in a forward fashion. The performance of a deep learning model which has three or more hidden layers, depends on the network architectures and other parameters such as activation function used in each layer and optimization algorithm. We use a genetic algorithm to find the optimal neural network parameters. Our approach allows us to find,

- the number of hidden layers,
- number of possible neurons in each hidden layer

A combination of all those components creates a multidimensional search space of all possible neural network architectures. Genetic algorithms have been some of the most commonly utilized approaches in the studies of the evolution of neural network architectures [20]. We have used the non-linear activated feature in every hidden layer, ReLU, to deploy a deep neural network that reduces the problem of vanishing and error gradient issue.. The benefit of ReLU is that the DNN model with a large number of hidden layers are easily trained and quicker than other non-linear activation functions. Adam is an algorithm designed especially for the training of deep neural networks for adaptive learning rates. For its rapid convergence rate and adaptable learning skills, we have used Adam as an optimizer.

3.1.2 Genetic algorithm

A genetic algorithm is a parallel stochastic search algorithm that simulates biological system evolution. It's also a population-based search algorithm that can efficiently and quickly search a diverse and multidimensional search space for a global optimum or near-optimal solution. A chromosome is a binary or alphabet string that contains a solution candidate. At first, a population of problem solutions is generated at random. Each solution is evaluated and assigned a fitness value according to the fitness function.A series of genetic manipulations, such as selection, crossover, and mutation, etc are applied to the current population to generate a new population. Until a certain condition is met the process gets repeated.

3.1.3 Data Preprocessing

We used the CSE-CIC-IDS-2018 [29] dataset, collected by the Canadian Institute for Cybersecurity to develop a deep learning model for intrusion detection. The dataset has been organized per day into 9 CSV files. The dataset has 80 network traffic features such as Flow duration, Total packets in the forward and backward direction, Minimum and Maximum time between two flows, Average size of packet etc. It comprises seven distinct attacks such as brute-force, heartbleed, botnet, dos, DDoS, web assaults, and infiltration of the network within. The dataset has about 2 million attack data and about 4 million benign non-attack data.

The dataset has been distributed in 9 CSV files named according to the day the attacks and network traffics were recorded. We have combined these 9 files into one file which has 6 million connection records. Data preprocessing steps are shown in figure 3.1

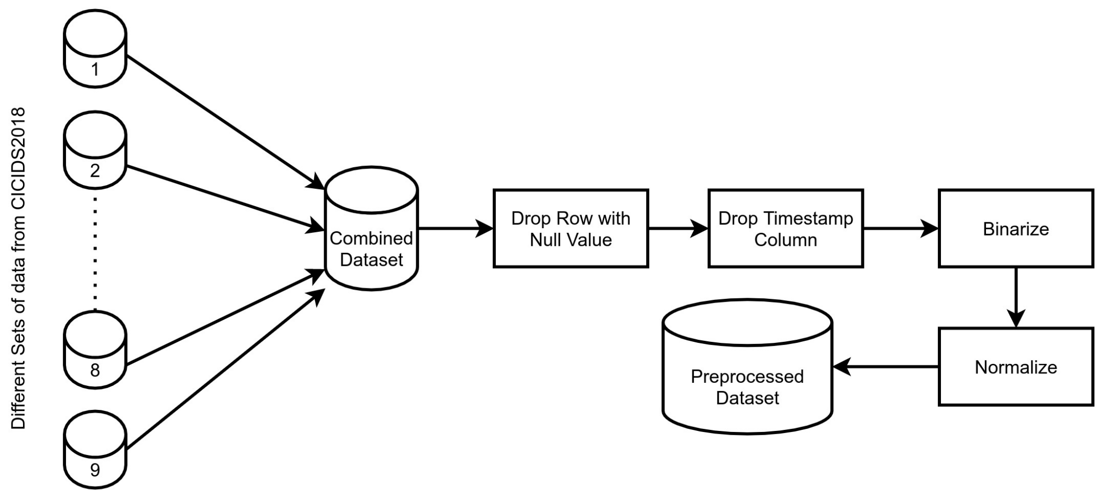

Figure 3.1: Data preprocessing

CSE-CIC-IDS2018 suffers from high data imbalance, null, and infinity value. To overcome data imbalance random under-sampling is used. We also cleaned duplicate data and data with null or infinity. Data is normalized to improve the generalization ability of DNN. Due to hardware restriction, We have randomly chosen 2 million connection records from the preprocessed dataset for further processing.

3.1.4 Network Architecture Search(NAS)

The input layer, hidden layer, and output layer are the three layers that make up the neural network. The input and output layer for a particular problem is fixed. The performance of an ANN largely depends on the design of hidden layer architecture design. The design of a hidden layer includes the number of hidden layers and the number of neurons in each layer and their activation function.

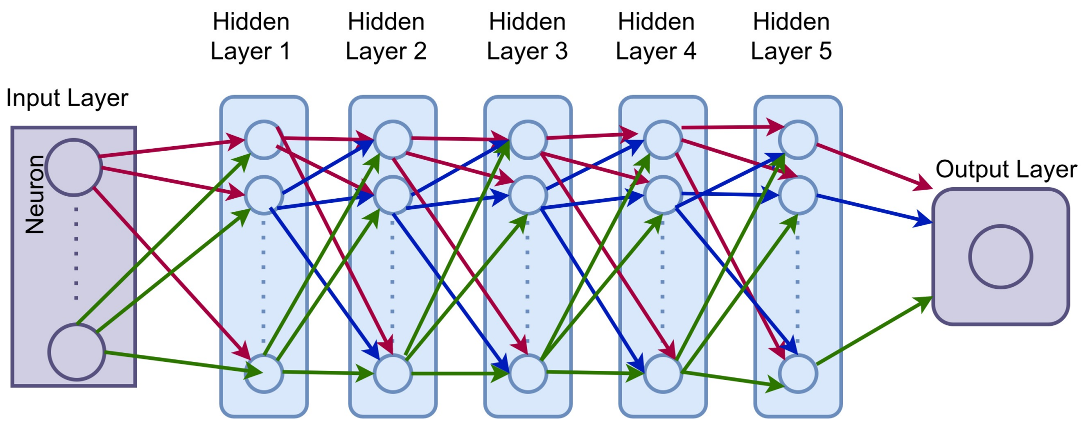

Figure 3.2: Deep Neural Network Architecture

Neural Architecture Search is a search problem where each point represents an architecture. There are many NAS algorithms such as genetic algorithm, trial, and error, particle swarm optimization, exhaustive search, pruning, and constructive algorithm, etc. In this paper, we use the genetic algorithm to automate the search process of DNN architectures for network intrusion problems.

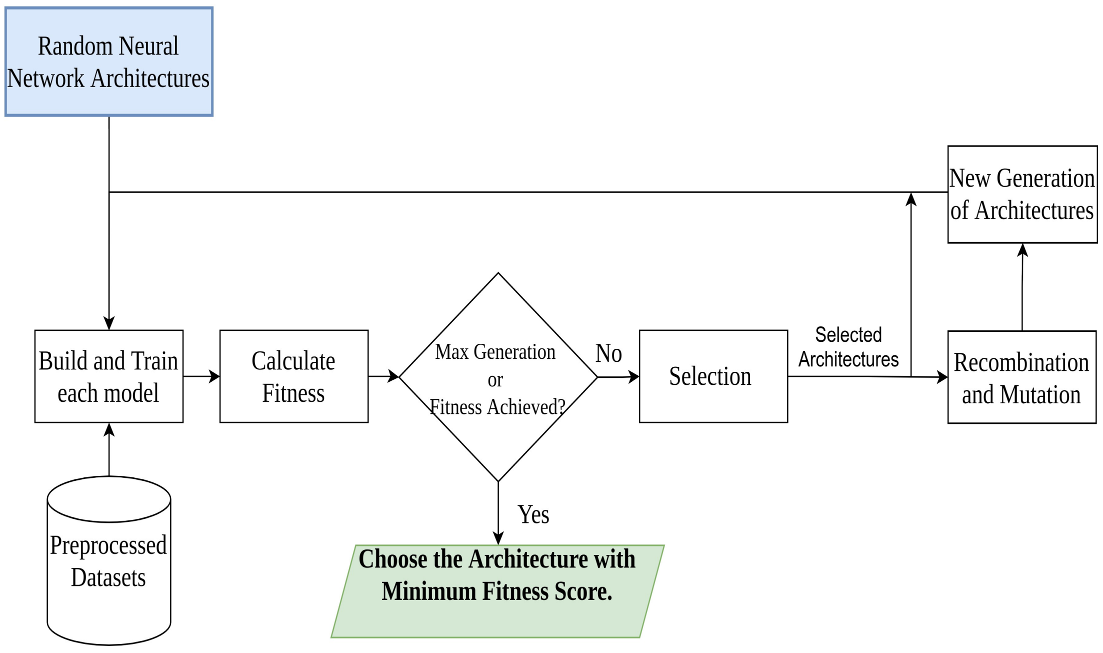
Random Neural Network Architectures
New Generation of Architectures
Build and Train each model
Calculate Fitness
Max Generation or Fitness Achieved?
No
Selection
Selected Architectures
Recombination and Mutation
Preprocessed Datasets
Yes
Choose the Architecture with Minimum Fitness Score.

Figure 3.3: Neural Network Architecture Search using Genetic Algorithm.

In figure 3.3, block diagram of Neural Network Architecture Search using Genetic Algorithm is shown.Encoded binary strings that represent the neural network architectures are initialized randomly. Each string is decoded to build a neural topology and trained using a prepossessed data set. The algorithm evaluates the model and calculates fitness using a fitness function. The goal of the algorithm is to minimize the fitness value. From the fitness value of the population of architectures, a certain number of chromosome strings is selected for further process. Those strings which are not selected are discarded for the population pool. Selected binary strings are recombined and mutated to generate a new set of binary strings that represent neural architecture. This new generation of chromosome strings are used to build new models. The whole process is repeated until the maximum generation is reached or a satisfactory fitness value is achieved.

3.1.4.1 Random Set of Neural Architectures

In the evolution of design tasks, the most important thing to remember is how to represent candidate solutions. The number of hidden layers and the number of neurons in each hidden layer is determined using the genetic algorithm. We used direct binary encoding to convert this data into a binary string. The initialization of 50 encoded binary strings that describe neural network architectures is done at random. In each hidden layer, we consider 5 potential hidden layers with a total of 1024 neurons. The secret layer architecture is represented by each chromosome, which is a binary string of length 50. The maximum number of hidden layers that can be used is five, each of 1023 neurons.

3.1.4.2 Neural Architecture Encoding:

Representing the candidate solutions is the most important issue to consider in the evolution of architecture tasks. We applied the genetic algorithm to find the number of hidden layers and the number of neurons in each hidden layer. We encoded this information into a binary string using direct binary encoding as shown in figure 3.4

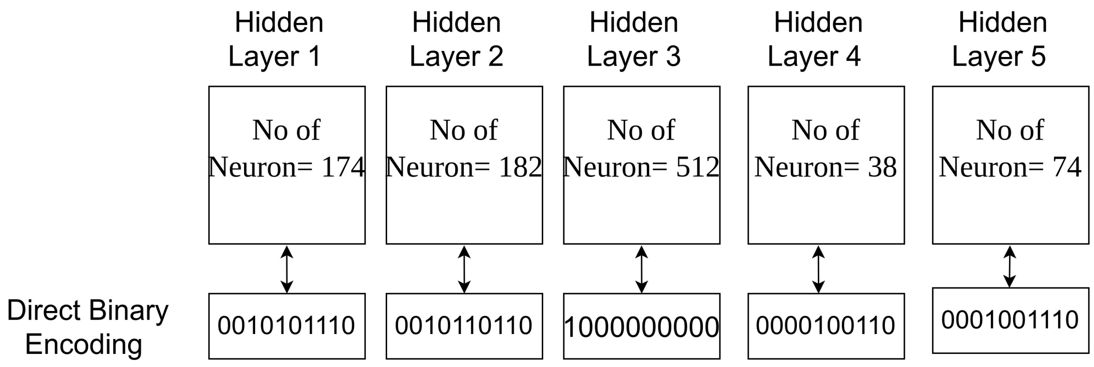

Figure 3.4: Binary representation of a neural network architecture.

We consider 5 possible hidden layers and 1024 maximum possible neurons in each hidden layer. Each chromosome is a binary string of length 50 and which represents the hidden layer architecture.

Table 3.1: Maximum Possible Architecture

Hidden Layer	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Neurons	1023	1023	1023	1023	1023

The maximum possible hidden layer consists of 5 hidden layers having 1023 neurons in each layer.

Table 3.2: Minimum Possible Architecture:

Hidden Layer	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Neurons	32	32	32	0	0

Minimum hidden layer representation will be all zero which means there will be no hidden layer between the input and output layer. To avoid this problem, 32 is added to the first three hidden layers when their values are zeros. When the values are zero fourth and fifth layers will not be used.So,the minimum possible architecture consists of 3 hidden layers having 32 neurons in each layer.

3.1.4.3 Build and Train

Back Propagation is used to train each neural network on the training set and performance is evaluated using a test set. For optimization we used Adam optimizer with default parameter. Binary cross-entropy is used as a loss function.

Activation function relu is used for inner layers and sigmoid activation function is used for the output layer.Each architecture is trained for 90 epochs with the batch size of 512.

3.1.4.4 Fitness Function

The fitness function calculates the fitness of each candidate solution. The genetic algorithm minimizes or maximizes fitness value according to the fitness function. The fitness function is the most crucial part of a genetic algorithm. Optimization of GA largely depends on the fitness function. A good intrusion detection model should detect intrusion accurately with less error rate and less computation cost. Our proposed fitness function is

$$F = W_{loss} \times loss + \frac{W_{specificity}}{specificity} + \frac{W_{parameter} \times (p - p_{min})}{(p_{max} - p_{min})} \tag{3.1}$$

For loss we used binary corss-entropy function. Binary cross-entropy loss is defined as

$$loss = -(y \times log(y^{pred}) + (1 - y)log(1 - y^{pred}))$$

where y is the true binary value and y^{pred} is the predicted value.
Specificity is the metric that evaluates a model's ability to predict the true negatives of each available category. In our proposed solution specificity specifies the attack class detection rate to all attacks. This fitness function tries to maximize the specificity value to get a good attack class detection rate.

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

The "p" represents parameter: the number of weights and bias in a deep artificial neural network. p_{max} and p_{min} represent the parameter of maximum and minimum possible neural network architecture of our proposed solution respectively. The overall objective of using p_{min} and p_{max} is to normalize the complexity of each topology tested to the interval (0,1) so that it can be smoothly incorporated into the fitness function.

$$parameter = \sum_{L=1}^{L=Total\,No\,Of\,Layer} (Neurons_L * Neurons_{L-1} + Bias_L)$$

$W_{loss}, W_{specificity}, W_{parameter}$, are three user-defined constants that represent the weight value of the loss, normalized parameter, and specificity to have more control over fitness function.

3.1.4.5 Selection

The selection process uses a hybrid method using the rank selection and the tournament selection method.

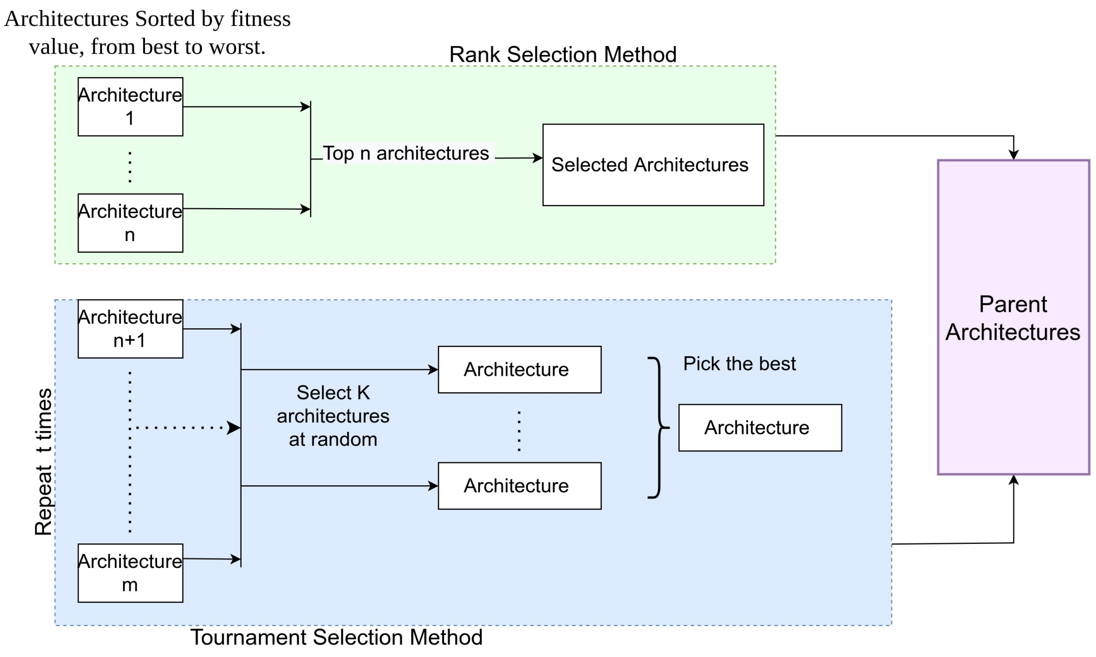

Figure 3.5: Selection process of the best architecture.

The algorithm selects several candidate solutions with the lowest fitness value. In the tournament selection method as described in Goldberg [30], the algorithm randomly selects two architectures and compares their fitness value. Architectures with the lowest fitness value are selected for recombination. This hybrid method ensures the diversity and effectiveness of the neural network architecture population pool.

3.1.4.6 Recombination

The selected architecture candidates are combined to get all possible groups of architecture candidates of size two called parents. From the pool of parents, several parents are selected randomly for recombination. The recombination process is shown in figure 3.6

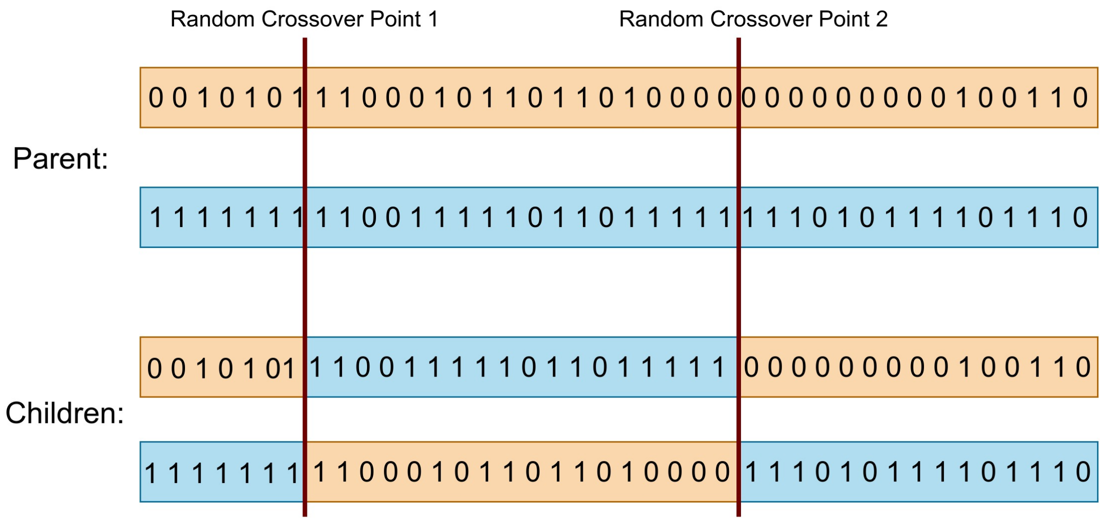

Figure 3.6: Two-point crossover of parent architectures

For recombination, a two-point crossover is used as prescribed in [31]. In a two-point crossover, two randomly chosen crossover points are used to exchange a string segment that falls between the two points. The recombination process generates a number of new chromosomes.

3.1.4.7 Mutation

A floating-point number in the range of (0,1) is chosen randomly and compared with a user-defined mutation probability value. If the random number is less or equal to the given number then mutation happens.

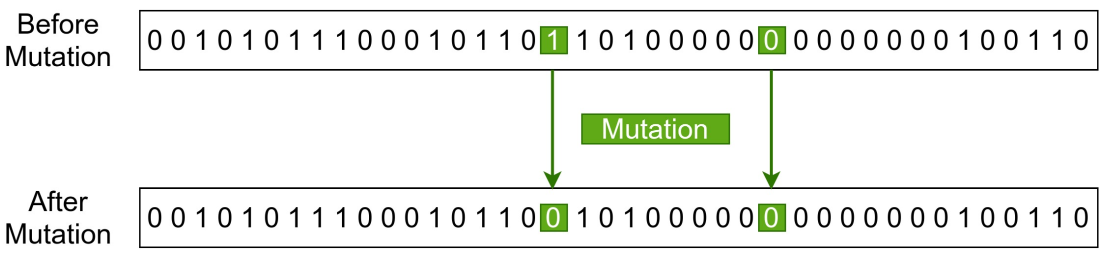

Figure 3.7: Mutation process of selected architecture

In mutation, a random binary string of the chromosome is complemented as shown in figure 3.7. A new generation is evolved by mutating the candidate strings from the recombination process.

3.1.4.8 New Generation of Nueral Architectures

From the recombination and mutation process a new generation of encoded neural architecture is evolved. 5 new random encoded archtectures are add to the set of new generation architectures. This new generation architecture will be used to built and train with proprocessed dataset to select the best architecture among them. The method will be repeated until the optimum number of hidden layers and nuerons are found.

3.2 Conclusion

In this chapter , genetic algorithm for network architecture search is discussed. The genetic algorithm finds optimal neural network architecture for intrusion detection. The next chapter is about the the experimental result analysis of the proposed framework.

Chapter 4

Results and Discussions

4.1 Introduction

This chapter addressed the implementation results of the suggested approach presented in the previous chapter. The dataset is preprocessed before being used for deep neural network testing and assessment. The suggested neural architecture search algorithm demonstrates improved architecture learning capabilities. At an affordable computational cost, the proposed deep learning model outperforms classical machine learning algorithms.

4.2 Dataset Description

We used the CSE-CIC-IDS-2018 [29] dataset, collected by the Canadian Institute for Cybersecurity to develop a deep learning model for intrusion detection. The dataset has been organized per day into 9 CSV files. The dataset has 80 network traffic features such as Flow duration, Total packets in the forward and backward direction, Minimum and Maximum time between two flows, Average size of packet etc. It comprises seven distinct attacks such as brute-force, heartbleed, botnet, dos, DDoS, web assaults, and infiltration of the network within shown in Table4.1. The dataset has about 2 million attack data and about 4 million benign non-attack data.

Table 4.1: Dataset Label Description

Class	Description
Brute-force	To gain unauthorized access, a brute force attack utilizes trial-and-error.
Heartbleed	Heartbleed is a vulnerability that causes servers to leak data stored in their database.
Botnet	A botnet is a group of malware-infected Internet devices that hacker control
Dos	A denial-of-service (DoS) attack attempts to make a network resource inaccessible to its potential users.
DDoS	A distributed denial-of-service (DDoS) attack is a form of dos attack in which many compromised computer systems attack a single target, such as a server, website, or other network resources.
Web assaults	Attacks to identify the vulnerability of a website
Infiltration of the network within	Infiltration of the network within.

We make two classes benign and attack for labeling of binary classification where the attack class has 7 distinct attacks data and the benign class has benign non-attack data. Benign class is labeled as 1 and attack class is labeled as 0.

Table 4.2: Training and testing CSE-CIC-IDS-2018 dataset

Class	**Train**	**Test**
Benign	1592052	957820
Attack	807948	478910

4.3 Evaluation

4.3.1 Evaluation Metrics

When evaluating the model's result, there are many widely used metrics, each with its own set of advantages and drawbacks, and some of them are mutually limited.
True Positive (TP): Normal network requests that are correctly detected.

True Negative (TN): Network intrusions that are correctly detected.

False Positive (FP): Intrusions that are mis-classified as normal requests.

False Negative (FN):Normal requests that are mis-classified as intrusions.

Table 4.3: Two label confusion Matrix

Actual	Prediction	
	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Accuracy rate The formula for accuracy rate is as follows:

$$Accuracy = (TP + TN)(TP + TN + FP + FG)$$

As can be seen from the above formula, the accuracy rate represents the proportion of the model's correct classification of the actual value of the sample to the data set.
Accuracy rate The formula for accuracy rate is as follows:

$$Accuracy = (TP + TN)/(TP + TN + FP + FG)$$

As can be seen from the above formula, the accuracy rate represents the proportion of the model's correct classification of the actual value of the sample to the data set.

Precision rate The formula for the precision rate is as follows:

$$Precision = TP/(TP + FP)$$

Precision rate represents the proportion of true positive samples to positive samples of the model classification.

Recall rate The formula for the recall rate is as follows:

$$Recall = TP/(TP + FN)$$

Recall rate shows the proportion of true positive samples in actual samples.

F1 value

$$F1 = (2 * Precision * Recall)/(Precision + Recall)$$

The F1 value is a harmonic average of the accuracy rate and the recall rate, which is equivalent to a comprehensive evaluation index of two values. The larger the value, the better the performance of the model.

Specificity

$$Specificity = TrueNegative/(TrueNegative + FalsePositive).$$

Specificity is also referred to as selectivity or true negative rate, and it is the percentage, or proportion, of the true negatives out of all the samples that do not have the condition (true negatives and false positives).

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$TPR = TP/(TP + FN)$$

False Positive Rate (FPR) is defined as follows:

$$TPR = FP/(FP + TN)$$

ROC Curve: A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

Area Under The Curve (AUC): That is, AUC measures the entire two-dimensional area underneath the entire ROC curve.AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting

4.3.2 Implementation

Back Propagation is used to train each neural network on the training set and performance is evaluated using a test set. For optimization we used Adam optimizer with default parameter as $learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999 and epsilon = 1e - 07$. Binary cross-entropy is used as a loss function. Activation function relu is used for all inner layer activation and sigmoid activation function is used for the output layer.
Dataset CSE-CIC-IDS2018 is preprocessed as described in the Data Preprocessing section. From the preprocessed dataset, a small subset of 70000 instances is selected randomly from which 50000 instances are used as train sets and the remaining 20000 instances are used as the test set. In the training process mentioned above, we used 512 as batch size, and each architecture is trained 90 epochs.
The free parameters W_{loss}, $W_{specificity}$, and $W_{parameter}$ of the fitness function defined in the previous section are important to converge into optimal fitness. Due to computational constraints, all three free parameters are set as $W_{loss} = 1, W_{specificity} = 2$, and $W_{parameter} = 0.01$. These three value is used to control the impact of loss,specificity and parameter over fitness value. As specificity value represents the ability to detect attack class, specificity value should have most impact on

fitness value.

The encoding scheme is introduced in the previous section and each encoded architecture represents a possible solution. A set of such solutions is called population in genetic algorithm terms. We initialized the population with 50 random encoded architectures. After the mutation process, 15 new generation architecture is selected and combined with their parents and 5 randomly generated architecture. The network architecture search using the genetic algorithm runs for 100 generations. So, the total number of architecture evaluated is 2030 from architecture search space size of 1.023×10^{15}.

4.4 Evaluation of Performance

4.4.1 Results of architecture evolution

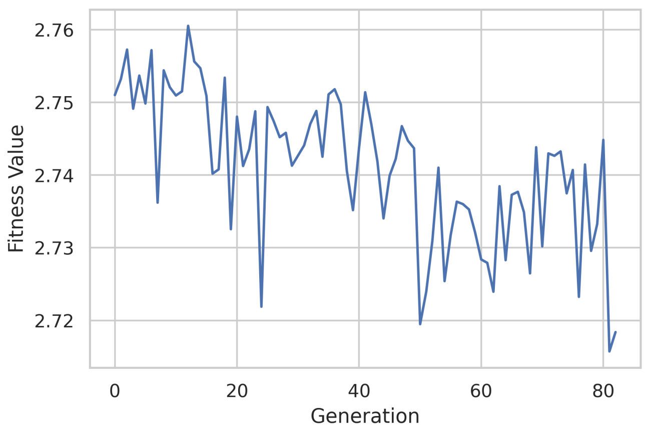

Figure 4.1: Min Fitness scores over generations

Figure 4.1 shows the smallest fitness score of each generation getting smaller over generations. The slope of the curve shows that the genetic algorithm has learned to find the neural network architecture of the smallest fitness value. We introduced random architecture in every generation which causes the many ups and downs of the curve. Here, 0 is the initial generation. The individuals in the population are getting better and better during the evolution process. After 82 generations the learning rate of the evolution algorithm becomes very slow. Due

Table 4.4: Optimal Neural Network Architecture

Hidden Layer	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Neurons	174	182	512	38	74

to computation hardware constraints, we stopped at the 82nd generation.

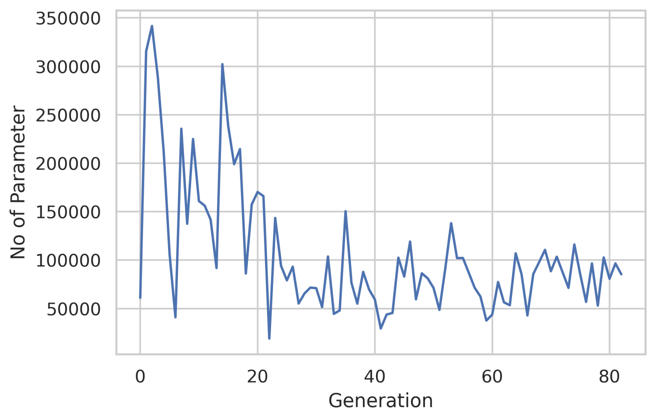

Figure 4.2: No of parameter over generations

The number of parameters reduces significantly in each generation as shown in figure 4.2. Here, the number of the parameters of the architecture of the smallest fitness value is plotted. The number of parameters represents the number of weights and biases in a deep artificial neural network architecture. The number of parameters is directly proportional to the computational cost of a model. So, the computational cost decreases as the number of parameters of the model decreases. The number of parameters has been reduced nearly 10 times from the model proposed by [8].

The fitness score of the 82th generation is the smallest and we take the individuals as the output of the neural network architecture search. Table 4.4 shows the optimal hidden layer architecture.

4.4.2 Results of Proposed Deep Learning Model

The attack class has been defined as the negative class in the preprocessing step. The accuracy of a test refers to how well it can detect true negatives. Specificity is also referred to as selectivity or true negative rate. A high specificity score represents the high attack class detecting ability. High true negative rate or specificity is crucial for intrusion detection problems.

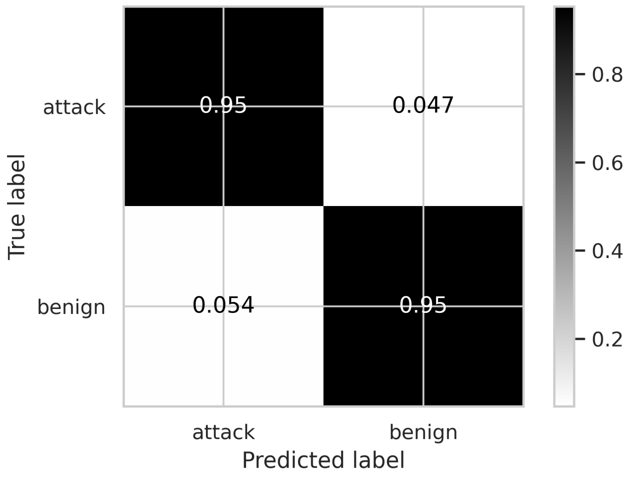

Figure 4.3: Confussion Matrix of the proposed model (row normalized)

The figure 4.3 shows the row normalized confusion matrix of the proposed deep neural network model. The proposed model can correctly identify 95% of the attacks and 95% of the benign or normal connection records.

The test results of various dnn hidden layer architectures are seen in Table 4.5. The highest f1 score was achieved by DNN with 5 and 7 hidden layers. With a minimal number of parameters, the proposed model had the highest specificty score.

Table 4.5: Test results of DNN hidden layer architectures

DNN Hidden Layer Architecture	Neurons	Accuracy	F1 Score	Specificity	Parameter
1 Layer	128	0.938	0.944	0.910	10241
2 Layer	128,256	0.951	0.977	0.882	43393
3 Layer	128,256,512	0.955	0.979	0.897	175233
4 Layer	128,256,512,1024	0.964	0.980	0.912	701057
5 Layer	128,256,512,1024,512	0.966	0.981	0.908	1225345
6 Layer	128,256,512,1024,512,256	0.967	0.978	0.890	1356417
7 Layer	128,256,512,1024,512,256,128	0.971	0.981	0.905	1389185
Proposed 5 layer	**174,182,512,38,74**	**0.954**	**0.969**	**0.954**	**161747**

The proposed model has the largest Area Under the Curve as shown in figure 4.4.

Table 4.6: Performance comparison of ML algorithms and DNN model

Algorithms	**Accuracy**	**Precision**	**Recall**	**F1 Value**	**Specificity**
Logistic Regression	0.86	0.86	0.98	0.91	**0.52**
KNeighbors	0.89	0.95	0.99	0.96	**0.84**
Decision Tree	0.9	0.95	0.95	0.95	**0.86**
Gaussian Naive Bayes	0.86	0.87	0.94	0.90	**0.56**
Random Forest	0.91	0.95	0.99	0.97	**0.85**
AdaBoost	0.86	0.93	0.99	0.97	**0.79**
Vinayakumar DNN [8]	0.94	0.94	0.99	0.96	**0.74**
Proposed DNN	**0.95**	**0.99**	**0.95**	**0.97**	**0.95**

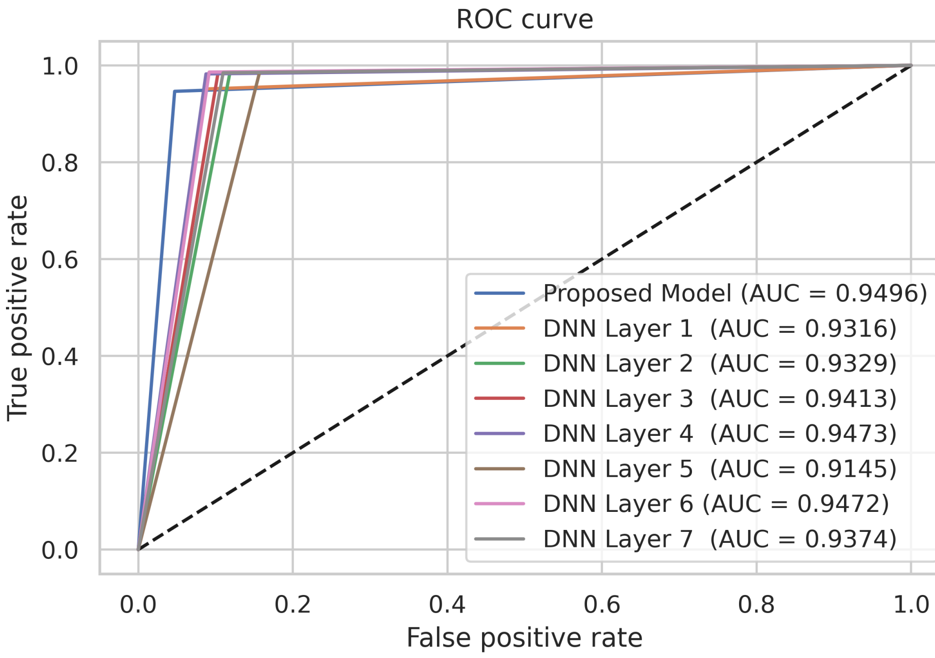

Figure 4.4: ROC curves of DNN layers

The Table 4.6 shows the accuracy, precision, recall,f1 score, and specificity of six machine learning algorithms and one deep neural network model [8] and compares it with the proposed deep neural network models. All these models have been trained and tested on the same set of test and train datasets used for the proposed model. The machine learning algorithms include Logistic Regression, Kneighborsclassifier, Decision Tree, Gaussian Naive Bayes, Random Forest, AdaBoost Classifier. Logistic Regression and Gaussian Naive Bayes have very low specificity scores which means that they have low detection capability of attack class. Although the Fitness scores of the Decision Tree, Random Forest algorithms, and the deep neural network [8] are 95% and 97%, and 96%, they have comparatively low specificity scores than the proposed model. The proposed model shows specificity score of 95%. Figure 4.5 showed that Classical machine learning algorithms are outperformed by the proposed model.

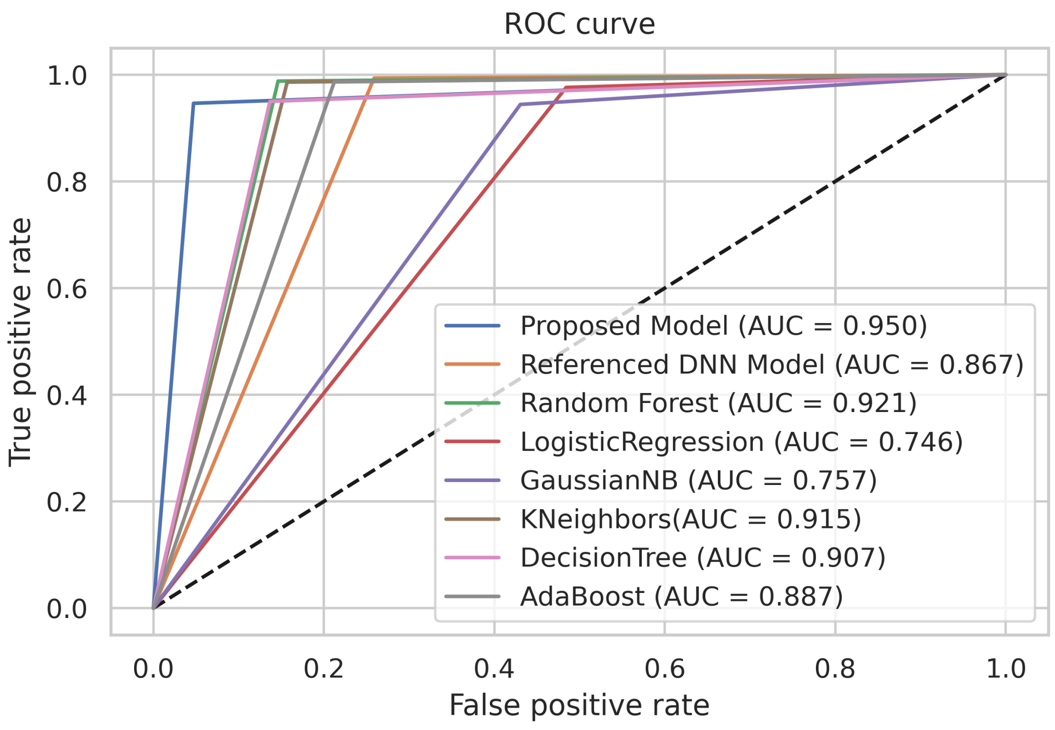

Figure 4.5: ROC curves of classical machine learning algorithm and proposed model

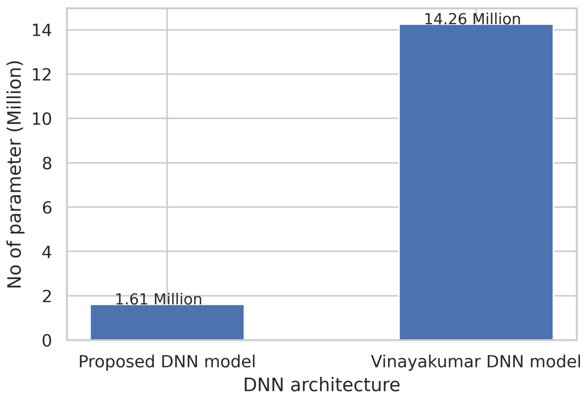

Figure 4.6: No of parameter of two DNN models

The no of parameters is another important performance metric that represents the total no of weights and biases of deep neural network architecture. A high no of parameters of a deep neural network architecture means the high computational cost of the architecture. The figure 4.6 shows a comparison between the no of parameters between the proposed deep neural network architecture and the deep neural network architecture referred to in [8]. The proposed model has 1.6 million parameters when the other model has 14.26 million parameters. The proposed model needs about 9 times less computational resources and also has better specificity, f1 score, and accuracy rate.

4.5 Conclusion

This chapter shows the performance evaluation result of proposed genetic algorithm and the proposed deep neural network architecture. Analyzing the performance we see that, our proposed methodology gives better intrusion detecting capacity than classical machine learning algorithm. In the next chapter, the conclusion is drawn on this thesis work.

Chapter 5

Conclusion

5.1 Conclusion

In this study, we have designed the deep neural network architecture by finding the optimal no of hidden layers and the no of neurons in each layer for intrusion detection classification problems on the CSE-CIC-IDS-2018 dataset. On the benchmark dataset CSE-CIC-IDS, the proposed model's output was compared to that of traditional machine learning classifiers. In every case, we found that the proposed DNNs outperformed traditional machine learning classifiers in terms of performance. Our proposed architecture outperforms previously implemented deep neural networks in terms of efficiency. To the best of our knowledge, the proposed deep neural network architecture outperforms previously implemented machine learning and deep learning algorithms in terms of attack class prediction capacity and computational cost.

5.2 Future Work

Overall, the performance of the genetic algorithm can be enhanced by expanding the search space on advanced hardware. They were not trained in this study using the benchmark IDS dataset CSE-CIC-IDS-2018 due to the high computational cost associated with the large search space. The proposed model's output can be evaluated using a variety of publicly accessible intrusion detection benchmark datasets. The use of the genetic algorithm to design deep neural network architecture for other classification problems is one of the important directions for future research.

References

[1] J. Johnson, *Internet users in the world 2021*, Apr. 2021. [Online]. Available: `https://www.statista.com/statistics/617136/digital-population-worldwide/` (cit. on p. 1).

[2] D. Freeze, *Top 5 cybersecurity facts, figures, predictions, and statistics for 2020 to 2021*, Mar. 2020. [Online]. Available: `https://cybersecurityventures.com/top-5-cybersecurity-facts-figures-predictions-and-statistics-for-2019-to-2021/` (cit. on p. 1).

[3] *Cost of a data breach study.* [Online]. Available: `https://www.ibm.com/security/data-breach` (cit. on p. 1).

[4] I. H. Sarker, A. Kayes, S. Badsha, H. Alqahtani, P. Watters and A. Ng, 'Cybersecurity data science: An overview from machine learning perspective,' *Journal of Big Data*, vol. 7, no. 1, pp. 1–29, 2020 (cit. on p. 2).

[5] M. A. Ferrag, L. Maglaras, H. Janicke and R. Smith, 'Deep learning techniques for cyber security intrusion detection: A detailed analysis,' in *6th International Symposium for ICS & SCADA Cyber Security Research 2019 6*, 2019, pp. 126–136 (cit. on pp. 2, 6).

[6] K. Kapanova, I. Dimov and J. Sellier, 'A genetic approach to automatic neural network architecture optimization,' *Neural Computing and Applications*, vol. 29, no. 5, pp. 1481–1492, 2018 (cit. on pp. 2, 8).

[7] S. Osken, E. N. Yildirim, G. Karatas and L. Cuhaci, 'Intrusion detection systems with deep learning: A systematic mapping study,' in *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, IEEE, 2019, pp. 1–4 (cit. on p. 5).

[8] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, 'Deep learning approach for intelligent intrusion detection system,' *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019 (cit. on pp. 5, 25, 27, 29).

[9] W. Tao, W. Zhang, C. Hu and C. Hu, 'A network intrusion detection model based on convolutional neural network,' in *International Conference on Security with Intelligent Computing and Big-data Services*, Springer, 2018, pp. 771–783 (cit. on p. 5).

[10] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, 'A deep learning approach to network intrusion detection,' *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018 (cit. on p. 6).

[11] I. H. Sarker, Y. B. Abushark, F. Alsolami and A. I. Khan, ‘Intrudtree: A machine learning based cyber security intrusion detection model,’ *Symmetry*, vol. 12, no. 5, p. 754, 2020 (cit. on p. 6).

[12] S. Rezvy, M. Petridis, A. Lasebae and T. Zebin, ‘Intrusion detection and classification with autoencoded deep neural network,’ in *International Conference on Security for Information Technology and Communications*, Springer, 2018, pp. 142–156 (cit. on p. 6).

[13] Y. Peng, J. Su, X. Shi and B. Zhao, ‘Evaluating deep learning based network intrusion detection system in adversarial environment,’ in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, IEEE, 2019, pp. 61–66 (cit. on p. 6).

[14] M. Aloqaily, S. Otoum, I. Al Ridhawi and Y. Jararweh, ‘An intrusion detection system for connected vehicles in smart cities,’ *Ad Hoc Networks*, vol. 90, p. 101 842, 2019 (cit. on p. 6).

[15] C. Callegari, E. Bucchianeri, S. Giordano and M. Pagano, ‘Real time attack detection with deep learning,’ in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, IEEE, 2019, pp. 1–5 (cit. on p. 6).

[16] D. Papamartzivanos, F. G. Mármol and G. Kambourakis, ‘Introducing deep learning self-adaptive misuse network intrusion detection systems,’ *IEEE Access*, vol. 7, pp. 13 546–13 560, 2019 (cit. on p. 6).

[17] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi and M. Ghogho, ‘Deep learning approach for network intrusion detection in software defined networking,’ in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263. DOI: `10.1109/WINCOM.2016.7777224` (cit. on p. 6).

[18] W. Elmasry, A. Akbulut and A. H. Zaim, ‘Evolving deep learning architectures for network intrusion detection using a double pso metaheuristic,’ *Computer Networks*, vol. 168, p. 107 042, 2020 (cit. on p. 7).

[19] S.-G. Choi and S.-B. Cho, ‘Adaptive database intrusion detection using evolutionary reinforcement learning,’ in *International Joint Conference SOCO’17-CISIS’17-ICEUTE’17 León, Spain, September 6–8, 2017, Proceeding*, Springer, 2017, pp. 547–556 (cit. on p. 7).

[20] M. Wistuba, A. Rawat and T. Pedapati, ‘A survey on neural architecture search,’ *CoRR*, vol. abs/1905.01392, 2019. arXiv: `1905.01392`. [Online]. Available: `http://arxiv.org/abs/1905.01392` (cit. on pp. 7, 9).

[21] T. Elsken, J. H. Metzen, F. Hutter *et al.*, ‘Neural architecture search: A survey.,’ *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019 (cit. on p. 7).

[22] B. Baker, O. Gupta, N. Naik and R. Raskar, ‘Designing neural network architectures using reinforcement learning,’ *arXiv preprint arXiv:1611.02167*, 2016 (cit. on p. 7).

[23] N. M. Wagarachchi, 'Mathematical modelling of hidden layer architecture in artificial neural networks,' Ph.D. dissertation, 2019 (cit. on p. 7).

[24] M. Wagarachchi and A. Karunananda, 'Optimization of artificial neural network architecture using neuroplasticity,' *International Journal of Artificial Intelligence*, vol. 15, no. 1, pp. 112–125, 2017 (cit. on p. 7).

[25] Y. LeCun, J. S. Denker and S. A. Solla, 'Optimal brain damage,' in *Advances in neural information processing systems*, 1990, pp. 598–605 (cit. on p. 8).

[26] J. Bergstra and Y. Bengio, 'Random search for hyper-parameter optimization.,' *Journal of machine learning research*, vol. 13, no. 2, 2012 (cit. on p. 8).

[27] D. Stathakis, 'How many hidden layers and nodes?' *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133–2147, 2009 (cit. on p. 8).

[28] Z. Fan, J. Wei, G. Zhu, J. Mo and W. Li, 'Evolutionary neural architecture search for retinal vessel segmentation,' *arXiv e-prints*, arXiv–2001, 2020 (cit. on p. 8).

[29] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, 'Toward generating a new intrusion detection dataset and intrusion traffic characterization.,' in *ICISSp*, 2018, pp. 108–116 (cit. on pp. 10, 19).

[30] D. Whitley, 'A genetic algorithm tutorial,' *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994 (cit. on p. 16).

[31] D. Thierens and D. Goldberg, 'Convergence models of genetic algorithm selection schemes,' in *International Conference on Parallel Problem Solving from Nature*, Springer, 1994, pp. 119–129 (cit. on p. 17).