POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Detection, Classification, and Modeling Evaluation of Network Intrusions

ALI HABIBZADEH MOTLAGH

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Juillet 2021

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé:

Detection, Classification, and Modeling Evaluation of Network Intrusions

présenté par **Ali HABIBZADEH MOTLAGH**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

DEDICATION

To my family

ACKNOWLEDGEMENTS

I would like to extend a huge thank you to my supervisor, Dr. Hanifa Boucheneb who has always supported me with great affection during my studies. Furthermore, I deeply appreciate the committee for reviewing my research and giving me valuable feedback. Moreover, my special thanks go to my supervisor in my previous master’s studies in Iran, Dr. Amir Naser Akhavan, for his continued kindness and support. Finally, I would like to thank my family for their endless support and encouragement during my life.

RÉSUMÉ

Les intrusions dans le réseau peuvent causer des dommages importants à différents systèmes. L'utilisation de méthodes d'apprentissage automatique nous aidera à identifier les intrusions au moment de l'occurrence de l'intrusion afin que nous puissions identifier les intrusions si tôt afin d'arrêter de manière significative les dommages. En outre, la classification nous aide à identifier et à reconnaître la catégorie et le type de l'intrusion concomitante. En conséquence, nous connaîtrons la stratégie appropriée pour vaincre l'intrusion puisque nous comprenons son type. Dans cette recherche, nous avons présenté différentes méthodes pour détecter les intrusions réseau ainsi que pour les classer. Nous présentons trois modèles de détection et de classification des intrusions réseau dans le but d'atteindre une précision supérieure à celle de la littérature existante. Nous avons utilisé trois méthodes, à savoir l'arbre de décision, la forêt aléatoire et l'approche d'apprentissage en profondeur. Tous les modèles proposés ont atteint des précisions supérieures à celles de la littérature existante, tandis que le modèle d'arbre de décision a obtenu le résultat le plus précis.

ABSTRACT

Network intrusions can cause significant damage to different systems. Using machine learning methods will help us to identify the intrusions at the time of the occurrence of the intrusion so that we can identify the intrusions so soon in order to significantly stop the damages. Besides, the classification assists us to identify and recognize the category and type of the concurring intrusion. As a result, we will know the proper strategy to defeat the intrusion since we comprehend its type. In this research, we have presented different methods for detecting network intrusions as well as classifying them. We present three models for the detection and classification of network intrusions with the aim of achieving higher accuracy than the existing literature. We have used three methods, namely Decision Tree, Random Forest, and Deep Learning Approach. All of the proposed models have achieved high accuracies than the existing literature while the decision tree model achieved the most accurate result.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction

As network intrusions are continuously present and the Internet users are increasing, it is essential to make the networks secure. Network security detects several kinds of intrusions such as viruses, worms, etc. in order to prevent the system from being harmed. There are several vulnerabilities associated with conventional methods of enforcing security in the system, such as authentication, access control, and encryption. To overcome these vulnerabilities, Intrusion detection systems are a useful and efficient solution. These systems monitor the network traffic and examine them for intrusion detection. (Alsaadi, Hedjam et al. 2020)

Network intrusions have caused notable damages in principal organizations in the US and Europe so far which has resulted in serious financial losses. To overcome this issue, intrusion detection systems have been developed. These systems frequently utilize machine learning methods for intrusion detection. (Piplai, Chukkapalli et al. 2020)

In this research, we aim to present different methods for detecting network intrusions as well as classifying them. Network intrusions can induce notable damage to various systems. Employing formal methods and deep learning methods will help us to identify the intrusions at the time of the beginning of the intrusion. This will lead to recognizing the intrusions so soon to significantly prevent damages. Moreover, we will classify the intrusions. The classification assists us to identify and recognize the category and type of the concurring intrusion. As a result, we will know the proper strategy to defeat the intrusion since we comprehend its type.

We present three methods for the detection and classification of network intrusions with the aim of achieving higher accuracy than the existing literature. The first method is the decision tree as a formal method. The second method is random forest and the third method is the deep neural network.

1.2 Research Question

The specific research question of this research is as follows:

- How can we detect network intrusions and classify them to a suitable class with higher accuracy than the current literature?

1.3 Objectives

We have the following objectives:

- Training a decision tree model on the KDD-99 dataset to classify and detect the network intrusions
- Training a random forest model on the KDD-99 dataset classify and detect the network intrusions
- Training a deep neural network on the KDD-99 dataset to classify and detect the network intrusions
- Comparing the accuracy of the three proposed models

1.4 Impact of Research

As time goes by, stronger cybersecurity threats are attacking different companies' systems that require more complex, professional, and updated approach to defeat them. This research will be beneficial for those companies in the industry to protect themselves against malicious intrusions. The benefit of this research will be to identify and classify network intrusions with higher accuracy.

1.5 Structure of the Thesis

In chapter 2, we present the literature review and previous work that is done by other researchers in order to understand the advantages and disadvantages of each proposed model. Chapter 3 includes the presentation of the three models namely decision tree, random forest, and deep neural network for detection and classification of network intrusions. In chapter 4, we evaluate the presented models. Finally, Chapter 5 includes the conclusion and suggestions for future work.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we will review the definitions of different intrusions. Moreover, we will review the related research that has been done so far in this area.

2.1 Network Intrusion Categories

Some of the categories of network intrusions are as follows (Vigneswaran, Vinayakumar et al. 2018):

- Denial of Service (DoS): In this type of intrusion, an attacker overloads the host by sending high volumes of requests to it. This action will lead to interrupting the host's services, temporarily or permanently.
- User to Root (U2R): In this type of intrusion, an attacker aims to use the network holes in order to manage the root. This action is done by targeting a user and obtaining his/her access to the network.
- Remote to Local (R2L): This is a type of intrusion in which an attacker aims to gain local access as a current user. In order to do so, the attacker seeks and utilizes a vulnerability by sending data packets to the machine while there is actually no real account.
- Probe: In this type of intrusion, an attacker tries to obtain root access by collecting specific information from the network such as its computers.

2.1.1 DoS

DoS category is comprised of several attacks. Some of them are as follows:

Smurf: The Smurf attack creates notable network traffic on the target system by spoofed broadcast ping messages. In this type of attack, the attacker sends echo requests to different addresses in the network including the spoofed source IP address of the target. As the hosts reply to the request, the target network will be flooded with large traffic. (Zargar and Kabiri 2009)

Neptune: In this attack, the attacker makes it improbable for the target to receive TCP connections from other hosts by sending lots of TCP connections to it. (Ahmad, Abdullah et al. 2009)

Land: In this attack, the attacker sends a spoofed SYN packet to the target. This will cause it to frequently synchronize with itself. (Ahmad, Abdullah et al. 2009)

Back: Back attack is an attack versus apache web servers in which the attacker's requests include several front slashes. The processing of these requests requires lots of time. (Ahmad, Abdullah et al. 2009)

Ping of Death: In this attack, the attacker sends oversized ICMP packets to the target host as ping requests. This will result in the unavailability of the target host. (Ahmad, Abdullah et al. 2009)

Teardrop: In this attack, the attacker sends overlapping IP parts that are wrongly administrated by older operating systems. This will result in the confusion of the target host. (Ahmad, Abdullah et al. 2009)

2.1.2 U2R

Some of the U2R attacks are as follows:

Buffer Overflow: This attack happens when lots of data is copied into a static buffer while no examination for the fitness of the data has been done. (Marinova-Boncheva and Robotics 2007)

Load Module: This is an attack on SunOS 4.1 systems that exploits in loadmodule program. As a result, the attacker can obtain root access. (Akasapu 2017)

PERL: This is an attack that utilizes a defect in Perl executions for obtaining root access. (Akasapu 2017)

Rootkit: In this attack, the attacker obtains user-level access by exploiting a vulnerability in a given system or breaking a password and then installs Rootkit that is a group of software tools for allowing the attacker to obtain controlling access to a computer. (Akasapu 2017)

2.1.3 R2L

R2L category include lots of different attacks. Some of them are as follows:

FTP Write: In this attack, the attacker appends some files into the ftp root directory so that it can obtain local access. (Xiao and Xiao 2007)

Guess Password: In this attack, the attacker finds out a guest user's password to have local access to the network. The password is usually easy to guess or none. (Akasapu 2017)

IMAP: In this attack, the attacker transmits packets to a machine in order to misuse its vulnerability for obtaining local access. (Wutyi and Thwin 2016)

PHF: This attack attempts to have access to some resources of a system. For instance, if a web server gives command shell access to someone without the certainty that the person is an authorized person, as a local administrator, this will be a PHF attack. This attack can be performed by either a local user or a remote user. (Wutyi and Thwin 2016)

Warezmaster: Warezmaster uses a bug in the file transfer protocol (FTP) server. This attack happens when the FTP server accidentally provides write permissions to users. Regularly, users do not have this permission, meaning that they do not have the permission to upload files on the server. In this attack, the attacker utilizes a guest account to connect to the server. The attacker will be able to upload copies of illegal software named that is called warez by generating a hidden directory. As a result, other users will download these files. Allocating the right permissions to the users is an easy way to prevent this attack. (Sabhnani and Serpen 2003)

Warez client: After the completion of the Warezmaster attack, any user will have the ability to perform a Warezclient attack. In this attack, users can download the warez software that was previously uploaded during the Warezmaster attack. This attack deceits to seem like a legal process as the files will be downloaded from the FTP server. However, there exists one fact that can reveal this attack. This attack can be detected by recognizing that the users are downloading files from the directories that are regularly not available for guest users. (Sabhnani and Serpen 2003)

2.1.4 Probe

Some of the Probe attacks are as follows:

Ipsweep: In this attack, the attacker discovers the hosts on the network in order to detect the vulnerabilities. (Akasapu 2017)

Nmap: Nmap is a tool that discovers the open and closed ports on the network by utilizing various scanning methods such as FIN, SYN, ACK, etc. (Akasapu 2017)

Portsweep: In this attack, the attacker discovers the open ports of a machine in a network. (Akasapu 2017)

Satan: Security Administrator Tool for Analyzing Networks (Satan) is a tool that investigates the loopholes in a system. (Akasapu 2017)

2.2 Related work

In this section, we have represented different methods that have been used for detection and classification of network intrusions along with most recent papers that have used the corresonding method.

2.2.1 Decision Tree

A Decision Tree is a multistage decision-making model that breaks up a complicated decision for classification into many small and easier decisions. This procedure will result in a decision that is supposed to resemble the expected decision. (Safavian, Landgrebe et al. 1991) Some papers that have used this method are as follows:

Bulavas utilized data visualization approach for intrusion detection. He deployed the Decision Tree method as well as PCA in their approach. It was concluded that this combination gives high accuracy in detecting certain attacks. As the future work, they mentioned deploying the model with another dataset. (Bulavas 2018)

Malik and Khan utilized particle swarm optimization (PSO) approach for pruning a decision tree for detecting network intrusions. They deployed single methods and multi-objective ones in the model. They compared their method with other classifiers such as rule-based, tree-based, Bayesian, and evolutionary ones and concluded that multi-objective optimized decision tree pruning gives the best results. (Malik and Khan 2018)

Corrêa et al. utilized Hoeffding Adaptive Tree for detecting the intrusions. Their model was binary, meaning that it could classify between normal and attack instances. They concluded that Hoeffding Adaptive Tree has a high accuracy of detection as well as having a high speed. As the future work, they suggested investigating the size reduction of the decision tree model created by Hoeffding Adaptive Tree. (Corrêa, Enembreck et al. 2017)

Papamartzivanos et al. presented the novel method of Dendron for classifying common intrusions as well as rare ones by creating novel detection rules for detecting network intrusions. To this end, They utilized Decision Trees and Genetic Algorithms. (Papamartzivanos, Mármol et al. 2018)

2.2.2 Random Forest

Random Forest mixes many decision trees and takes the average of the predictions in order to reach a reliable prediction. Random Forest works by utilizing the "divide and conquer" rule. It samples parts of the data and develops a randomized tree predictor on each part. Then, it adds the predictions together. (Biau and Scornet 2016) Some of the papers that have used this method are as follows:

Zhang et al. utilized the Random Forest method for detecting the intrusions. They adjusted the random forest method to Apache Spark for detection purposes on Netflow data. The proposed framework was comprised of different components such as kafka, logstash, and Spark cluster. They concluded that their model has a shorter detection time with higher accuracy. As the future work, they mentioned presenting a hybrid framework for achieving better performance. (Zhang, Dai et al. 2018)

Jiang et al. mixed the improved Rough Set Theory (RST) with Random Forests approach in order to create a hybrid model and named the proposed model RST-RF. They used the two methods individually for the purpose of classification and feature selection. They compared the proposed model with different classifiers such as Naïve Bayes, J48, and CART and found out that their model has a better performance. As the future work, they mentioned investigating ways to have faster optimization. (Jiang, Wang et al. 2018)

Nazir and Khan presented a new feature selection method that was based on the wrapper. In the proposed model, they utilized Tabu search for the searching prepose. Furthermore, they utilized random forest method for detecting the intrusions. They named the method Tabu Search - Random Forest (TS-RF). They compared the proposed model with Chi-Square, Gain Ratio, and Pearson Correlation methods and concluded that their model has a better performance. As the future work, they mentioned studying metaheurisitc algorithms for feature selection and detection of intrusions. (Nazir, Khan et al. 2021)

Min et al. presented TR-IDS as a network intrusion detection system utilizing statistical features as well as payload features. Furthermore, they implemented Word embedding and text-convolutional neural networks for obtaining data from payloads with natural language processing methods. In the next step, they mixed statistical features and payload features and implemented a random forest method on it for classification. (Min, Long et al. 2018)

Zhang et al. proposed a framework for intrusion detection consisting of the five modules of preprocessing module, autoencoder module, database module, classification module, and feedback module. Furthermore, they mixed Sparse Autoencoder and Random Forest methods in order to present a classification method. They performed binary classification as well as multiclass classification. They aim to enhance restore function and retraining function as the future work. (Zhang, Chen et al. 2021)

2.2.3 Support Vector Machine

The support vector machine method attempts to determine the optimal separating hyperplane between different classes. This is done by concentrating on the training instances that are located at the corner of the class descriptors. One of the advantages of this algorithm is that less training data is used. (Tzotsos and Argialas 2008) The papers that have used this method are as follows:

Sakr et al. used support vector machine for classifying and detecting network intrusions in the cloud environment. They utilized the binary-based Particle Swarm Optimization for choosing suitable features. Furthermore, they used the standard-based Particle Swarm Optimization for enhancing the parameters of the support vector machine. As the future work, they mentioned working on other methods for reaching optimal classification parameters and feature selection. (Sakr, Tawfeeq et al. 2019)

Ghanem et al. presented an unsupervised system for intrusion detection that utilizes statistical methods. Furthermore, they included support vector machine method in the system to improve its performance. The support vector machines that were used had different forms such as one-class and two-class, as well as deploying linear and non-linear functions. (Ghanem, Aparicio-Navarro et al. 2017)

Tao et al. deployed support vector machine method and genetic algorithm to present a new approach for intrusion detection. As a matter of fact, this approach optimizes the support vector machine based on genetic algorithm. They concluded that this approach speeds up convergence and classification. Furthermore, it improves the true positive rate as well as reducing the error rate. (Tao, Sun et al. 2018)

Sakr et al. aimed to improve intrusion detection systems by applying hybrid, filter, and wrapper feature selection methods. Furthermore, they utilized the support vector machine method for classification. Their model was binary, meaning that it could classify between normal and attack instances. They compared the feature selection methods and concluded that the wrapper method had the best performance. As the future work, they mentioned using a deep learning model for intrusion detection. Additionally, they mentioned using other feature selection methods as well as utilizing real data for the evaluating purpose. (Sakr, Tawfeeq et al. 2019)

2.2.4 Naïve Bayes

Naïve Bayes is a probabilistic classification method that considers the number and mixtures of instances and computes the probabilities. This algorithm utilizes Bayes's theorem. (Saritas, Yasar et al. 2019) The papers that have used this method are as follows:

Yang et al. utilized the Naïve Bayes algorithm for network intrusion detection. Particularly, they used to artificial bee colony algorithm to alter the Naïve Bayes method. They compared the proposed approach to the basic Naïve Bayes and other types Naïve Bayes algorithms that are based on different methods such as genetic algorithm, water wave optimization, and grey worlf optimizer. They concluded that the proposed method outperforms all of them. (Yang, Ye et al. 2018)

Najeeb et al. presented a novel feature selection method for network intrusion detection. The proposed binary method was called Firefly algorithm. They utilized this method on Naïve Bayes classifier and found out that this method decreases the rate of false alarms and therefore improves the detection. Moreover, they improved Firefly algorithm by hamming distance calculation approach. As the future work, they mentioned utilizing other methods for the improvement of feature selection. (Najeeb and Dhannoon 2018)

2.2.5 K Nearest Neighbors

K nearest neighbor algorithm estimates the relation between cases, determining how similar or different they are. It considers that an instance class will be the same as the class of the nearest instance. (Jiang, Cai et al. 2007) The papers that have used this method are as follows:

Rao and Swathi utilized two different types of K nearest neighbors method, namely Indexed Partial Distance Search K Nearest Neighbor (IKPDS) and Partial Distance Search K Nearest Neighbor (KPDS) for speeding up the classification procedure. They concluded that IKPDS consumes less time than the conventional K Nearest Neighbor method. As the future work, they suggested several improvements such as deploying feature selection and including a wavelet transformation on K nearest neighbors method in order to compare it with IKPDS. (Rao, Swathi et al. 2017)

Chen et al. utilized K nearest neighbor method for network intrusion detection. Moreover, they deployed treeseed algorithm (TSA) to obtain important features. As the future work, they mentioned applying TSA to other methods such as extreme learning machine, support vector machine, etc. They mentioned applying the model to other fields such as image classification and text classification, as well. (Chen, Ye et al. 2018)

Xu et al. utilized Distance-weighted K-Nearest Neighbor method and Moth-Flame Optimization (MFO) for intrusion detection and named it a MFO-KNN algorithm. The main reason that they added MFO to their algorithm was the fact that it was time efficient and could converge fast. Additionally, the distance formula of the K-Nearest Neighbor algorithm was optimized by MFO. Furthermore, they compared the proposed method with other methods such as Support Vector Machine, the distance-weighted KNN algorithm improved by Gray Wolf (GWO-KNN), the distance-weighted KNN algorithm Improved by Particle Swarm Optimization (PSO-KNN), KNN, and Naïve Bayes. They concluded that the proposed method had a better performance. (Xu, Fang et al. 2018)

2.2.6 Neural Networks

Artificial neural networks try to simply simulate the function of the human brain. They are powerful models that try to attain the optimal nonlinear function based on how complex the network is.

Neural networks are capable of learning complicated links between the input and output data. (Kavzoglu and Software 2009) Some papers that have used this method are as follows:

Vinayakumar et al. used convolutional neural networks for intrusion detection. They presented a time series model of transmission control protocol / internet protocol (TCP/IP) packets for network traffic and utilized various methods including CNN-long short-term memory (CNNLSTM), CNN, CNN-recurrent neural network (CNN-RNN), CNN-gated recurrent unit (GRU), and multi-layer perceptron (MLP) and assessed their effectiveness. They concluded that CNN outperforms other models because it obtains feature illustrations at a high level. (Vinayakumar, Soman et al. 2017)

Khan et al. introduced the new semi-supervised model of two-stage deep learning (TSDL) for detecting network intrusions including two steps. In the first stage, the softmax classifier classifies the normal and abnormal instances. This step includes two hidden layers. The second step is to classify between normal instances and various types of attacks. As the first step, this step is also comprised of two hidden layers. (Khan, Gumaei et al. 2019)

Yang and Wang utilized the improved convolutional neural network (ICNN) method for detecting network intrusions and named it ICNN Based Wireless Network Intrusion Detection Model (IBWNIDM). They used a stochastic gradient descent algorithm to achieve an optimized neural network. They concluded that their model has higher accuracy compared to previous models, namely Intrusion Detection Algorithm Based on Convolutional Neural Network (IDABCNN) and Network Intrusion Detection Model Based on Convolutional Neural Network (NIDMBCNN). (Yang and Wang 2019)

Yu et al. suggested a deep learning approach for detecting complicated attacks to networks. They accumulated dilated convolutional autoencoders to acquire representations of beneficial features. They conducted several classification tasks and examined the outcomes of different hyperparameters. (Yu, Long et al. 2017)

Mirza and Cosan presented a sequential long short term memory (LSTM) neural network for detecting network intrusions. Three kinds of pooling layers were included in their model, namely as Mean, Max, and Last layers. Their dynamic model has the capability of feature extraction as well. (Mirza and Cosan 2018)

Zhang et al. presented deep hierarchical network model for intrusion detection by combining LeNet-5 and LSTM designes. In this model, original flow data is used to obtain temporal features. They suggested creating a system for gatjering traffic data in the model. (Zhang, Chen et al. 2019)

Mohammadpour et al. presented a convolutional neural network model for network intrusion detection. They were able to utilize a convolutional neural network approach by interpreting a1D array to a 2D array so that the data could be fit for the model. They suggested assessing the model by implementing it on a real-time dataset as the future work. (Mohammadpour, Ling et al. 2018)

Vigneswaran et al. employed Deep Neural Networks for detecting network intrusions. They tested different architectures of deep neural networks on the KDD-99 dataset, classifying it exclusively to attack and benign categories, and inferred that including 3 hidden layers in the neural network is the optimal choice. Furthermore, they compared this choice with other classification methods such as SVM, Random Forest, Ada Boost, etc., and concluded that their model choice has the highest accuracy. (Vigneswaran, Vinayakumar et al. 2018)

Khan et al. presented a convolutional neural network for network intrusion detection. They compared the model to SVM and DBN models and concluded that they have reached higher accuracy. (Khan, Zhang et al. 2019)

Al-Zewairi et al. presented a deep learning model for detecting network intrusions. They utilized stochastic gradient descent and backpropagation methods in their model. Furthermore, they used tanh activation function in the proposed model. (Al-Zewairi, Almajali et al. 2017)

Yang et al. presented a deep neural network model for detecting network intrusions, particularly for Supervisory Control and Data Acquisition (SCADA) networks. SCADA systems are used in modern industrial control (ICS) systems. As the future work, they suggested including combined types of attacks and measuring the accuracy of the model. (Yang, Cheng et al. 2019)

Wu et al. utilized convolutional neural networks for detecting network intrusions in real-time. To this end, they transformed data into image format. However, the features were chosen from the raw data. They demonstrated that transforming data into image format decreases the computation cost. As the future work, they suggested enhancing the performance and therefore the accuracy of the proposed model. Moreover, they suggested ensuring that the intrusions will be detected at a suitable time while the accuracy will be improved. (Wu, Chen et al. 2018)

Wu and Guo presented LuNet as a deep neural network that combines convolutional neural network and recurrent neural network. LuNet employs convolutional neural networks for learning spatial features. Additionally, it utilizes LSTM in order to learn temporal features. They included batch normalization as well for improving the learning. As the future work, they mentioned working on the accuracy of LuNet for classifyin the attacks of small samples that currently do not have high accuracy. (Wu and Guo 2019)

Iqbal and Aftab used two types of neural networks, namely Feed Forward Neural Network and Pattern Recognition Neural Network for intrusion detection. They utilized the training functions of Bayesian Regularization and Scaled Conjugate Gradient in their models. As the future work, they suggested boosting the models and deploying them for other datasets. (Iqbal, Aftab et al. 2019)

Xu et al. presented a deep learning approach for network intrusion detection. Their model was a recurrent neural network. It was comprised of softmax module, Gated Recurrent Units (GRU), and Multilayer Perceptron (MLP). They concluded that GRU has better performance than LSTM as a memory for intrusion detection. As the future work, they suggested proving the model's practical utilization rather than theoretical. Additionally, they mentioned optimizing the model for real networks, as well. (Xu, Shen et al. 2018)

Zhang et al. deployed deep learning approach for intrusion detection. They decreased the size of features by extracting the important ones by a denoising autoencoder (DAE) that includes a weighted loss function. The classification was based on a Multi-Layer Perceptron classifier. As a result of the feature selection, the proposed model had low computational costs. (Zhang, Wu et al. 2018)

He et al. presented a network intrusion detection method that is comprised of hierarchical progressive network formation. This formation enables the model to combine the data of various features in a network connection. As a result, the data of similar network connections can be learned simultaneously. Multimodal Deep Auto Encoder (MDAE) and LSTM maintain this model. As the future work, they mentioned creating a new traffic acquisition system. (He, Sun et al. 2019)

Wu et al. presented Pelican as a deep neural network that is based on building-block for intrusion detection. Moreover, they included a recurrent neural network as well as a convolutional neural network in their model. Their work had the limitation of small training data and lack of strong computing tools. As the future work, they mentioned presenting a deeper model with a larger dataset. (Wu, Guo et al. 2020)

Hamid et al. deployed coupling Discrete Wavelet Transforms and Artificial Neural Network to present a hybrid approach. The proposed neural network was comprised of three layers. As the future work, they suggested decreasing the dimensionality to avoid the curse of dimensionality. (Hamid, Shah et al. 2019)

Shenfield et al. proposed a neural network for intrusion detection. Their approach was binary, meaning that the model could differentiate between normal and malicious traffic. The proposed neural network was comprised of two hidden layers. (Shenfield, Day et al. 2018)

Khan et al. utilized Spark ML and convolutional-LSTM network to present a hybrid approach for network intrusion detection. Their approach was comprised of two steps. In the first step, Spark ML detects anomalies while in the second step, the convolutional-LSTM network detects misuses. Since the proposed model deploys both convolutional neural network and LSTM, it includes both Anomaly-based and Signature-based methods. As the future work, they suggested testing the proposed model on a real-time streaming data. (Khan, Karim et al. 2019)

Zhiqiang et al. presented a deep neural network for network intrusion detection. The propoese model used feed-forward approach. As the future work, they mentioned presenting a Self Taught Learning System as well as Ending techniques for decreasing the dimentionality. (Zhiqiang, Mohi-Ud-Din et al. 2019)

Xu et al. utilized meta-learning framework to introduce a network intrusion detection approach. They created a deep neural network including a comparison network as well as a feature extraction network and named it FC-Net. This model uses network traffic samples to learn feature maps. It does the classification task by comparing them to the acquired feature maps. (Xu, Shen et al. 2020)

Kanimozhi and Jacob aimed to detect and classify botnet attacks by presenting a neural network with multi-layer perceptron algorithm. They classified the attacks based on two classes of benign or malicious. (Kanimozhi and Jacob 2019)

Rezvy et al. presented a deep neural network with autoencoded dense for intrusion detection. They were able to improve the accuracy as well as reducing the time of detection with the proposed model. For the future work, they mentioned improving the accuracy as well as decreasing the false positive and false negative rates. Additionally, they mentioned using more complex datasets in the future. Moreover, they stated presenting an extension of the proposed approach for mobile and IoT programs in the future. (Rezvy, Petridis et al. 2018)

Wang et al. presented a deep multi-scale convolutional neural network (DMCNN) for detecting the intrusions. They obtained various levels of features from the data. Additionally, they optimized the learning rate by batch normalization approach. They concluded that the proposed model has high accuracy. However, the problem associated with the model was that parameters converge in local optimal value instead of the global one. To this end, they mentioned considering methods such as particle swarm optimization and simulated annealing for tuning the parameters, as the future work. Moreover, they mentioned testing the model on other datasets, as well. (Wang, Yin et al. 2020)

Qu et al. utilized deep neural network for intrusion detection. They compared the presented method with the Support Vector Machine method and Back Propagation Neural Network and concluded that their model achieves better results as well as consuming less time for detection than the mentioned methods. As the future work, they suggested working more on determining the depth of the proposed neural network. Furthermore, they suggested mixing the proposed model with other models to achieve a more accurate model. (Qu, Zhang et al. 2017)

Xiao et al. presented a convolutional neural network for intrusion detection. They utilized PCA and auto-encoder methods for decreasing the dimensionality of the features. Furthermore, they transformed the data into the image format in order to decrease the calculation cost. Moreover, they used other machine learning methods such as Support Vector Machine, Random Forest, Adaboost, Naïve Bayes, Logistic Regression, and Decision Tree and concluded that the proposed method had a better performance in terms of accuracy and time. For the future work, they mentioned working more on U2R and R2L attacks as these categories include a small number of attack that causes low detection accuracy as well as difficult feature learning. (Xiao, Xing et al. 2019)

Andresini et al. proposed a deep neural network for detecting network intrusions. They utilized both supervised and unsupervised learning methods in their model. The model learned normal and attack autoencoders in the unsupervised step. Furthermore, they used a multi-channel parametric convolution in the unsupervised step. The main objective of this work was to select the important features in order to improve the accuracy. Their model could identify between normal and attack instances. As the future work, they suggested enhancing the model so that it could classify attacks into different categories. (Andresini, Appice et al. 2020)

Dong et al. presented a deep learning method for intrusion detection. Their model utilized natural language processing as well as big data. They utilized Flink for calculation and Flume log collection purposes. Furthermore, they presented a method for decreasing the data dimension. As the future work, they mentioned enhancing the model performance. (Dong, Wang et al. 2019)

Lie et al. presented a deep learning model based on random forest method for network intrusion detection. As a matter of fact, they utilized random forest in some layers to act as hidden layers. This characteristic resulted in having a better generalization ability as well as having high accuracy. They used the proposed model in the Spark environment. As the future work, they mentioned enhancing the accuracy of detection. (Liu, Su et al. 2020)

Hu et al. utilized adaptive synthetic sampling (ADASYN) method as well as convolutional neural network (CNN) to present a method for intrusion detection. The main reason for using ADASYN was the fact that it does not let the model neglect small samples. The CNN that they used was based on the split convolution module in order to enhance the variety of features. Furthermore, they implemented other methods such as Support Vector Machine and Naïve Bayes and concluded that the proposed method had a better performance. As the future work, they mentioned using the simplified residual network for increasing the detection capability. (Hu, Wang et al. 2020)

Zhang et al. used Bayesian Convolutional Neural Network for intrusion detection. Furthermore, they used the Support Vector Machine method, as well. As the future work, they mentioned utilizing other datasets for evaluating the method. Additionally, they mentioned enhancing the neural network design, as well. (Zhang, Li et al. 2020)

Sun et al. utilized Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) to present a hybrid model for obtaining important features in order to detect the intrusions. Since the samples had a different number of instances, they deployed the category weight optimization approach for decreasing the effect of being unbalanced as well as increasing the robustness of the proposed model. Moreover, they mentioned using Generative Adversarial

Networks (GAN) as well as adding some conventional traffic features in the future work. (Sun, Liu et al. 2020)

2.2.7 Combined Methods

There exist several papers that have used and combined several methods. These papers are addressed as follows:

Gao et al. presented a MultiTree approach by combining several decision trees for network intrusion detection. Furthermore, they selected some classifiers and presented an ensemble approach. The classifiers included decision tree, deep neural network, K nearest neighbors, and random forest. They compared MultiTree method with the ensemble method and concluded that the ensemble method performs better. Moreover, the ensemble method was capable of being generalized well. As the future work, they suggested optimizing the feature selection approach to obtain higher accuracy. (Gao, Shan et al. 2019)

Zhou et al. linked various classifiers that are based on modified adaptive boosting with an area under the curve (M-AdaBoost-A) algorithm to present an ensemble model. They used particle swarm optimization as well as other approaches for making an ensemble model. (Zhou, Mazzuchi et al. 2020)

Sukumar et al. proposed an improved genetic k-means algorithm (IGKM) for network intrusion detection. Moreover, they made a comparison between the presented algorithm and k-means++ algorithm considering 4 main types of attacks namely, Dos, R2L, U2R, and Probe attack. They concluded that their model performs better compared to k-means++ on KDD-99 dataset. (Sukumar, Pranav et al. 2018)

Wang et al. utilized equality constrained-optimization-based ELM (C-ELM) for detecting network intrusions. Furthermore, they presented a method for achieving the best possible hidden neuron quantity and obtaining the output weights. (Wang, Xu et al. 2018)

Dahiya and Srivastava presented an approach that utilizes feature reduction methods as well as implementing classification methods. The feature reduction algorithms were comprised of Canonical Correlation Analysis (CCA) and Linear Discriminant Analysis (LDA) algorithms. They realized that LDA was faster and gave higher accuracy than CCA. The classification methods included Naïve Bayes, Random Forest, Random Tree, etc. (Dahiya and Srivastava 2018)

Choi et al. deployed an unsupervised learning method for detecting network intrusions based on autoencoders. As a result, their proposed method solely requires inception from the abnormal data that was earlier noted, not manual labeling. They suggested using other unsupervised learning algorithms other than autoencoders as the future work. (Choi, Kim et al. 2019)

Roshan et al. presented an intrusion detection system base on Extreme Learning Machines. Their suggested method can detect both common and new attacks. Furthermore, security experts have the ability to modify and update it when novel data directions appear. (Roshan, Miche et al. 2018)

Kabir et al. utilized a Bayesian Network for detecting network intrusions. Furthermore, their method includes feature selection using wrapper method. This lead to the decrease of time and complexity and higher accuracy of the model. They observed that their proposed approach achieved better results than other methods such as K Nearest Neighbors, Markov chain, etc. As the future work, they suggested utilizing other methods of feature selection and improving the proposed framework. (Kabir, Onik et al. 2017)

Verma et al. utilized XGBoost and Adaboost methods for detecting network intrusions. They used these two methods in both cases of adding clustering and without it. They concluded that the suggested methods outperform the previous methods. (Verma, Anwar et al. 2018)

Singla et al. utilized adversarial domain adaptation method for detecting network intrusions. Their model had a small amount of training data. They classified the instances into the two classes of benign and attack. As the future work, they suggested studying the problems including semi-supervised as well as unsupervised domain adaption method. Moreover, they suggested using domain adaption for multi-class classification as well. (Singla, Bertino et al. 2020)

Zhang et al. utilized ensemble method and binary tree to present a hybrid model for classification of network intrusions. Stacked sparse autoencoder network was utilized in their model for obtaining features. Then, they used these features to create an Xgboost model and named it SSAE-XGB. Furthermore, this model considers unknown cases by adding sparsity constraint. This improved the generalization of the proposed model. (Zhang, Yu et al. 2018)

Verma and Ranga presented ELNIDS, which is a Ensemble Learning based Network Intrusion Detection System. They performed various classifiers such as Subspace Discriminant, Boosted Trees, RUSBoosted Trees, and Bagged Trees. They concluded that Boosted Trees and RUSBoosted Trees perform the best as an ensemble classifier. The proposed system can detect several attacks such as Clone ID, Selective Forwarding, and Sinkhole. As the future work, they mentioend executing the system on smart nodes. (Verma and Ranga 2019)

Timčenko and Gajin have tested various ensemble methods such as LogitBoost, Adaboost, and bagged trees, GentleBoost, and RUSBoost for network intrusion detection. They analyzed different methods and concluded and high accuracies can be achieved GentleBoost and Bagged tree methods in their study and the lowest accuracy is associated with RUSBoost. The classes were divided into either normal or anomaly. As the future work, they suggested presenting methods for detecting various intrusions. (Timčenko and Gajin 2017)

Belouch et al. utilized Apache Spark to measure different machine learning algorithms for network intrusion detection in terms of accuracy and creation time, and forecasting time. The algorithms were namely as Decision Tree, Support Vector Machine, Naïve Bayes, and Random Forest. In their study, the Random Forest classifier achieved the best performance. Their approach was classifying the instance into either an attack or normal. As the future work, they mentioned decreasing the time. Furthermore, they suggested creating an approach for the multi-class classification of intrusions based on different categories. (Belouch, El Hadaj et al. 2018)

Mirza deployed different classification methods such as Neural Networks, Logistic Regression, and Decision Trees for presenting a binary classification of network intrusions. Furthermore, he mixed the mentioned methods for increasing the performance by ensemble learning. To this end, he associated weights with each classifier. Moreover, the utilized PCA for decreasing the feature dimension. (Mirza 2018)

Lee et al. utilized an extreme machine learning method, namely constrained optimization-based extreme learning machine (C-ELM) for detecting intrusion. Their model was time-efficient during its creation due to the improvement of optimizing criteria. (Lee, Su et al. 2017)

Aiken and Scott-Hayward used different classification methods such as Support Vector Machine, Random Forest, K Nearest Neighbors, and Logistic Regression for detecting intrusions. They concluded that the K Nearest Neighbors method performs the best in their study. Furthermore, they presented a tool called Hydra for measuring the adversarial. (Aiken and Scott-Hayward 2019)

Musafer et al. presented a framework for optimizing the hyperparameters of back-propagation algorithm, such as the learning rate, number of the nodes of the hidden layers, etc. based on Hassan–Nelde–Mead method. The aim of their study was to increase the performance of sparse autoencoders which is utilized for classification and feature selection. (Musafer, Abuzneid et al. 2020)

Sahu et al. performed several data processing techniques such as normalization, feature correlation, feature reduction, and data balancing to understand the effect of these techniques on the classification performance of different classifiers. The classifiers included Logistic Regression, Decision Tree, Naïve Bayes, Support Vector Machine, Random Forest, Neural Network, etc. In their study, non-linear classifiers performed better than linear ones. They concluded that data balancing has effects on Neural Networks in some conditions. However, it does not have an effect on logistic regression. (Sahu, Mao et al. 2020)

Jo et al. presented three data preprocessing methods called "Direct", "Weighted", and "Compressed" for network intrusion detection. These methods can be used for Convolutional Neural Networks. Direct conversion requires minimum information. As the future work, they suggested combining the proposed methods. (Jo, Kim et al. 2020)

Yu et al. presented dual adaptive regularized online sequential extreme learning machine (DA-ROS-ELM). They resolved over-fitting as well as ill-posed problems by utilizing the ridge regression factor. The proposed model benfits from online sequential extreme learning machine (OS-ELM) by acquiring its sequential learning design. However, it is faster and more efficient than OS-ELM. (Yu, Kang et al. 2018)

Alzahrani et al. utilized Support Vector Machine and Structural Sparse Logistic Regression (SSPLR) for detecting and classifying intrusions. To this end, they aimed to select the most essential feature. They concluded that the SSPLR method was more efficient in terms of calculation. As the future work, they mentioned testing the proposed approach on a more up-to-date and realistic dataset. (Alzahrani, Shah et al. 2020)

Yin et al. deployed generative adversarial network to present a new supervised method for intrusion detection. The proposed framework helps the classifier to increase its accuracy by creating fake labeled examples. As the future work, they mentioned investigating the impact of the proposed approach on classifiers other than the utilized one. Moreover, the mentioned studying the hyperparameters of the proposed approach, as well. (Yin, Zhu et al. 2019)

Thaseen et al. deployed different machine learning methods such as Support Vector Machine, Naïve Bayes, Random Forest, and K Nearest Neighbors for network intrusion detection. They concluded that Random Forest provides the highest accuracy. They included 4 classes in their task as a normal packet or an encrypted one and a normal malicious or an encrypted one. As the future work, they suggested utilizing different deep neural networks for intrusion detection. (Thaseen, Poorva et al. 2020)

Aboueata et al. deployed Support Vector Machines and Artificial Neural Networks for detecting intrusions. Furthermore, they utilized feature selection methods to select the ones that gave the highest accuracy. This also led to decreasing the time and complexity of the model. They concluded that the artificial neural network method achieved somewhat better results than the support vector machine. As the future work, they mentioned performing multi-class classification. (Aboueata, Alrasbi et al. 2019)

Ioannou and Fahmy proposed a neural network model for intrusion detection. Their model was binary, meaning that it could classify between malicious instances and normal ones. As the future work, they mentioned utilizing deeper neural networks and utilizing datasets other than the used ones. (Ioannou and Fahmy 2019)

Saia et al. proposed a Discretized Extended Feature Space (DEFS) model for detecting the intrusions. Their model had two benefits. The similar instances would be grouped together and therefore the number of event patterns would be decreased. Furthermore, the discretization will be balanced by adding some meta-information that can characterize them. They concluded that the proposed method enhances the classification. As the future work, they mentioned evaluating the proposed model in other fields, for instance, fraud detection. (Saia, Carta et al. 2019)

Moghanian et al. presented an artificial neural network and Grasshopper Optimization Algorithm (GOA) for network intrusion detection. They decreased the error rate by utilizing a metaheuristic algorithm with the swarm-based method. Furthermore, they deployed GOA for improving the accuracy of the artificial neural network. As the future work, they mentioned enhancing metaheuristic algorithms as well as presenting a deep neural network. (Moghanian, Saravi et al. 2020)

Hashemi and Keller presented the novel approach of Reconstruction from Partial Observation (RePO) for intrusion detection. Furthermore, they utilized denoising autoencoders in their approach. Denoising autoencoders could identify attacks with high accuracy. They concluded that the proposed approach improved intrusion detection. (Hashemi and Keller 2020)

Peng et al. aimed to study the robustness of intrusion detection systems by presenting the Evaluating Network Intrusion Detection System (ENIDS) approach. They considered the models that are based on the deep learning method. They utilized various methods such as support vector machine, logistic regression, and random forest methods in order to test the proposed approach. Moreover, they used high-level approaches such as Momentum Iterative Fast Gradient Sign, L-BFGS attack, and Projected Gradient Descent attack, as well. They concluded that network intrusion detection systems that are based on deep learning, do not have a good performance, especially while facing the MIFGSM attack. As the future work, they mentioned improving defensive methods. (Peng, Su et al. 2019)

Tan et al. proposed an approach for detecting attacks by utilizing time slot-based features. The proposed approach was based on the neural attention mechanism. Furthermore, they deployed conditional random fields and bidirectional LSTM methods in order to make a comparison with their method. They concluded that the proposed method has a better performance. As the future work, they mentioned considering include attention mechanisms in unsupervised learning methods. (Tan, Iacovazzi et al. 2019)

Natesan et al. presented a Hadoop Based Parallel Binary Bat Algorithm for intrusion detection concerning big data. Furthermore, they used Naïve Bayes method, as well. Using these methods, they concluded that the complexity of computations would be decreased for both classification and feature selection. Moreover, they would decrease the detection time as well as increasing its accuracy. (Natesan, Rajalaxmi et al. 2017)

Zhang et al. utilized directed acyclic graph (DAG) and belief rule base (BRB) methods to present an intrusion detection model and named it DAG-BRB. Moreover, they presented a constraint covariance matrix adaption evolution strategy (CMA-ES) for determining the best parameters for the model. As the future work, they mentioned decreasing the calculation complexity of the proposed model. (Zhang, Hu et al. 2017)

Jabbar et al. presented an ensemble classifier with K Nearest Neighbors and ADTree method for detecting the intrusions. Additionally, they utilized K means clustering method, as well. As the future work, they mentioned mixing the intrusion detection system with evolutionary methods. (Jabbar, Aluvalu et al. 2017)

Lopez-Martin et al. used a conditional variational autoencoder to present an approach for detecting the intrusions of the Internet of Things. Their approach was less complicated than similar unsupervised approaches. Additionally, the proposed approach had the ability to regenerate the missing features in the case that the training set was deficient. Moreover, they used different methods such as Support Vector Machine, logistic regression, and random forest for comparison purposes. They concluded that the proposed method offers better classification results. As the future work, they mentioned utilizing ladder variational autoencoder and structured variational autoencoder. (Lopez-Martin, Carro et al. 2017)

Rajadurai and Gandhi mixed different machine learning methods as a stacked ensemble model for intrusion detection. Their method was comprised of two levels. The first level included base classifiers which were namely Random Forest and Gradient Boosting methods. The second level included a Meta classifier. Furthermore, they used other machine learning methods as Convolutional Neural Network, Multi-Layer Perceptron, Naive Bayes, Support Vector Machine, etc., and concluded that the proposed method had better performance. They mentioned combining other methods such as Decision Tree, Probabilistic, etc. to present an ensemble method in the future. (Rajadurai, Gandhi et al. 2020)

Alsaadi et al. utilized matched filter optimization to present an intrusion detection method. They named the proposed model NIDeMFO. Their model could classify between normal and attack instances which means it was binary. Furthermore, they used other methods such as support vector machine, random forests, LSTM. They concluded that the proposed model consumes a short time for training. As the future work, they mentioned working on multi-class classification. (Alsaadi, Hedjam et al. 2020)

Krishnaveni et al. utilized univariate ensemble feature selection to present an ensemble method for detecting intrusions in the cloud environment. They utilized the voting method for classification. The proposed model was binary, classifying between normal and attack instances. As the future work, they mentioned using deep neural networks instead of the ensemble model. (Krishnaveni, Sivamohan et al. 2021)

Yu and Bian used Few-Shot Learning (FSL) to present an intrusion detection approach. Furthermore, they used CenterLoss function in order to increase the accuracy of the proposed model. They tested the model on both modes of binary and multi-class classification. Moreover, they implemented other machine learning models for comparison. The models included Support Vector Machine, Naïve Bayes, Random Forest, Recurrent Neural Network, etc. They concluded that the proposed model gives higher accuracy. As the future work, they mentioned using an initialization-based FSL model. (Yu and Bian 2020)

Pamukov et al. presented a classification approach for detecting the intrusions of IoT network and named it Negative Selection Neural Network (NSNN). Their model was comprised of two layers. First, they considered the operation of a normal network and utilized a Negative Selection method to generate a training set. Then, they used a Neural Network for the classification. (Pamukov, Poulkov et al. 2018)

Wang et al. presented a deep belief network (DBN) for detecting the intrusions. They deployed the Kernel-based Extreme Learning Machine (KELM) instead of the Back Propagation (BP) method. One of the benefits of this approach was its shorter training time. Furthermore, they presented an enhanced grey wolf optimizer (EGWO) for achieving the optimal parameters of KELM. For the future work, they mentioned studying dimensionality reduction as well as extending the field of utilization of the proposed model. (Wang, Zeng et al. 2021)

Verma et al. utilized a Convolution Neural Network for detecting the network intrusions. They deployed oversampling method as the classes were imbalanced. Their model was binary and it could detect between normal and anomaly instances. Moreover, they utilized other classification methods such as Support Vector Machine, Naïve Bayes, Random Forest, etc. for comparison purposes. They aimed to increase the performance of the model in the future work. (Verma, Kaushik et al. 2019)

El-Sappagh et al. used various machine learning methods to detect and classify cyber threats in KDD-99 dataset. Their methods were Multilayer Perceptron, Rule based model, support vector machines, Naïve Bayes, Apriori, K Means, and Decision Tree with ID3 algorithm. They considered the overall category of attacks including 5 classes. They achieved the accuracy of 92.03 percent in classification with Multilayer Perceptron classifier. (El-Sappagh, Mohammed et al. 2019)

Rashid et al. proposed a new method for intrusion detection based on DNA sequence disease detection. To this end, they took advantage of Cryptography encoding to transform the data into DNA sequence form. They used Teiresias algorithm to identify Short Tandem Repeats (STR) patterns. Finally, they deployed Horspool algorithm and performed the classification process. They tested the proposed model on KDD-99 test set and achieved an accuracy of 77.65%. (Rashid, Othman et al. 2017)

2.3 Discussion of Related Work

In conclusion, various machine learning methods are used for detection and classification of network intrusions that are trained and tested on different datasets. There exist some papers that used the KDD-99 dataset such as (Moghanian, Saravi et al. 2020), (El-Sappagh, Mohammed et al. 2019), (Rashid, Othman et al. 2017), etc. Most of them classified the attacks considering the overall five categories (DoS, R2L, U2R, Probe, and normal). On the other hand, one paper have considered all of the 37 distinct attacks of the test set is (Rashid, Othman et al. 2017). Taking only five classes into consideration leads to missing some vital information about the attacks. For instance, if we know the exact subset of category, this will help us to gain better knowledge of the intrusions in order to defeat them more properly. Therefore, it would be more beneficial to classify them to the most detailed classification to understand the attacks better which will assist us in facing and stopping them. Therefore, we will classify the attacks considering 37 attack classes and one normal class of the test set.

CHAPTER 3 METHODOLOGY

In this chapter, we will illustrate the methodology of our research. We have deployed three approaches, namely Decision Tree, Random Forest, and Deep Learning Approach.

3.1 Dataset

MIT Lincoln Labs planned and provided the 1998 DARPA Intrusion Detection Evaluation Program with the aim of studying and assessing research in the field of intrusion detection. They presented a standard dataset comprising of numerous and diverse intrusions. The KDD-99 utilizes a subset of it. They formed a training data by observing and retrieving network traffic data for seven weeks. The initial data was approximately four gigabytes. The KDD-99 dataset is comprised of intrusions that are categorized into four main classes: DoS, R2L, U2R, and Probe. (Stolfo, Fan et al. 2000)

This dataset is one of the largest datasets that includes a wide variety of intrusions. There exists many different types of intrusions from different categories in this dataset. Additionally, there exists many recent papers that have used this dataset. Therefore, we decided to use this dataset in our research. Other datasets may be used in future work.

3.1.1 Training and Test data

The KDD-99 dataset is divided into to training and test sets by itself. The training set is comprised of 4898431 records and the test set is comprised of 311029 records. The probability of the test set is different from the training set. An interesting point about the test set is that it includes several types of attacks that are not included in the training set. There exists 14 additional attack types in the test set that are not included in the training set. This characteristic is to simulate a real environment to observe the performance of the classifier while faces completely unknown attacks. (Stolfo, Fan et al. 2000)

3.1.1.1 Training data

Table 3.1 illustrates the number of instances from each attack in the train set. As it is shown, the Smurf attack has the most instances with the number of 2807886 instances. Neptune attack has the second rank with 1072017 instances, and the number of normal instances in the training set is

972781, as the third rank. The attacks with least population are spy, perl, and phf that respectively have 2, 3, and 4 instances in the training set. It can be inferred that the range of the number of each attack is so wide.

3.1.1.2 Test data

The number instances from each attack is shown in Table 3.1. In the test set, the smurf attack has the most instances with the number of 164091 instances. However, Normal instances has the second rank with 60593 instances, and the number of neptune attacks in the training set is 58001, as the third rank. The attack with least population is imap that have only 1 instance in the training set. As the training set, the range of the number of each attack is so wide. As mentioned before, there exists a large number of attacks in the test set that are not included in the training set.

3.1.2 Features

Table 3.2 represent different features of the dataset. Generally, the features can be split into the following divisions (Xu, Shen et al. 2018):

- Basic features: These are features that are derived from packet headers.
- Content-based derived features: These features are base on content and are proposed by domain information.
- Same host features: The connections in the past 2 seconds that their destination host is the same as the existing connection are measured by same host features.
- Same service features: The connections in the past 2 seconds that their service is the same as the existing connection are measured by same host features.

Table 3.1 Comparison of Training Set and Test Set instances (Stolfo, Fan et al. 2000)

	Attack	**Training Set**	**Test Set**
1.	Smurf	2807886	164091
2.	Neptune	1072017	58001
3.	Normal	972781	60593
4.	Satan	15892	1633
5.	Ipsweep	12481	306
6.	Portsweep	10413	354
7.	Nmap	2316	84
8.	Back	2203	1098
9.	Warezclient	1020	-
10.	Teardrop	979	12
11.	Pod	264	87
12.	guess_passwd	53	4367
13.	buffer_overflow	30	22
14.	Land	21	9
15.	Warezmaster	20	1602
16.	Imap	12	1
17.	Rootkit	10	13
18.	Loadmodule	9	2
19.	ftp_write	8	3
20.	Multihop	7	18
21.	Phf	4	2
22.	Perl	3	2
23.	Spy	2	-
24.	Snmpgetattack	-	7741
25.	Mailbomb	-	5000
26.	Snmpguess	-	2406
27.	Mscan	-	1053
28.	apache2	-	794
29.	Processtable	-	759
30.	Saint	-	736
31.	Httptunnel	-	158
32.	Named	-	17
33.	Sendmail	-	17
34.	Ps	-	16
35.	Xterm	-	13
36.	Xlock	-	9
37.	Xsnoop	-	4
38.	Sqlattack	-	2
39.	Udpstorm	-	2
40.	Worm	-	2

Table 3.2 Features of the dataset (Stolfo, Fan et al. 2000)

	Feature	**Type**
1.	duration	Continuous
2.	protocol type	Symbolic
3.	service	Symbolic
4.	flag	Symbolic
5.	src bytes	Continuous
6.	dst bytes	Continuous
7.	land	Symbolic
8.	wrong fragment	Continuous
9.	urgent	Continuous
10.	hot	Continuous
11.	num failed logins	Continuous
12.	logged in	Symbolic
13.	num compromised	Continuous
14.	root shell	Continuous
15.	su attempted	Continuous
16.	num root	Continuous
17.	num file creations	Continuous
18.	num shells	Symbolic
19.	num access files	Symbolic
20.	num_outbound_cmds	Continuous
21.	is host login	Symbolic
22.	is guest login	Symbolic
23.	count	Continuous
24.	srv count	Continuous
25.	serror rate	Continuous
26.	srv serror rate	Continuous
27.	rerror rate	Continuous
28.	srv rerror rate	Continuous
29.	same srv rate	Continuous
30.	diff srv rate	Continuous
31.	srv diff host rate	Continuous
32.	dst host count	Continuous
33.	dst host srv count	Continuous
34.	dst host same srv rate	Continuous
35.	dst host diff srv rate	Continuous
36.	dst host same src port rate	Continuous
37.	dst host srv diff host rate	Continuous
38.	dst host serror rate	Continuous
39.	dst host srv serror rate	Continuous

40.	dst host rerror rate	Continuous
41.	dst host srv rerror rate	Continuous

3.2 Data Preprocessing

Some of the features are both symbolic and written as strings, such as protocol type, service, flag. In order to process them, we have converted them to arrays of binaries by designating either 1 or 0 to them. For the deep learning approach, we have done additional preprocessing as normalization and feature scaling since these are required for achieving the optimal result for deep learning.

3.3 Methods

In this section, we will explain the methods that we have used in order to detect and classify the intrusions. These methods are Decision Tree, Random Forest, and Deep Neural Network.

3.3.1 Decision Tree

A Decision Tree is a multistage decision-making method that breaks up a difficult and complicated decision for classification into many small and easier decisions. This approach will result in a decision that is supposed to match the expected decision. (Safavian, Landgrebe et al. 1991) The general form of a decision tree can be found in figure 3.1:

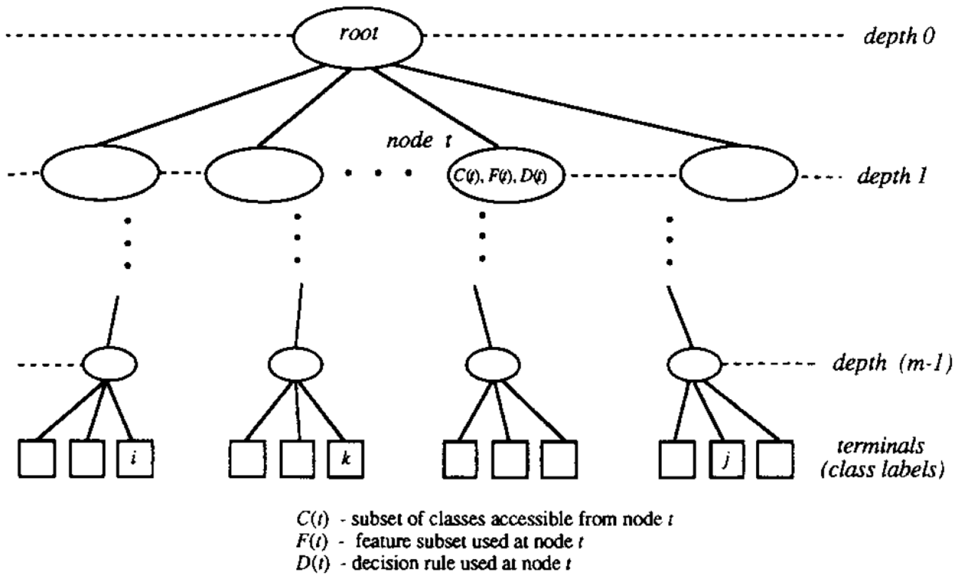
root
depth 0
node t
C(t), F(t), D(t)
depth 1
depth (m-1)
i
k
j
terminals (class labels)
C(t) - subset of classes accessible from node t
F(t) - feature subset used at node t
D(t) - decision rule used at node t

Figure 3.1 Structure of a Decision Tree (Safavian, Landgrebe et al. 1991)

The Decision Tree method is a powerful machine learning method that has accomplished excellent results in other similar network intrusion detection tasks such as (Belouch, El Hadaj et al. 2018) and (Gao, Shan et al. 2019). That being said, we have used this robust method in our research to detect and classify network intrusions. Additionally, this method has various advantages for the classification purpose which is the other reason that we have used it. The advantages of the Decision Tree method can be defined as follows (Safavian, Landgrebe et al. 1991):

1. A combination of local decision areas, that are meant to be simple, can estimate the global decision areas that are complicated and difficult. This estimation is done on several levels in decision trees.
2. Decision tree examines each sample on particular subsets of classes in order to exclude redundant calculations. This characteristic is unlike traditional classifiers that examine each sample on all classes, causing low performance.
3. Decision Tree classifier takes various feature subsets that will distinguish between classes in a certain node with high accuracy. Determining feature subsets is done at various internal nodes. The flexibility of the Decision Tree classifier in this case enhances its performance which is against the traditional single-stage ones. Those classifiers singularly utilize one features subset.
4. Decision Tree classifier has the ability of utilizing fewer features at internal nodes while avoiding extreme reduction of accuracy. This characteristic is useful the evade high-dimensionality problem while being implemented on small training data set.

On the other hand, this method has some disadvantages as follows (Safavian, Landgrebe et al. 1991):

1. In some cases, the number of terminal nodes may be greater than the number of data classes, which will result in higher time consuming for matching new data to one of the classes.
2. Large decision trees may suffer from the problem of growing errors as levels go down. Furthermore, it is not possible to improve accuracy and performance concurrently.

Finally, considering the high accuracy of the decision tree method in the previous work such as (Belouch, El Hadaj et al. 2018) and (Gao, Shan et al. 2019) ,and its various advantages, we have used it as one of our methods for classification.

3.3.1.1 Design of a Decision Tree

In decision tree classification, we aim to reach the following goals (Safavian, Landgrebe et al. 1991):

1. Classification on the training set should be as precise as possible.
2. Classification could be generalized with high accuracy to new data.
3. Decision tree should have the capability of being updated with new training examples if it was available.
4. The composition of the decision tree should be uncomplicated and simple.

Following the Bayes math, the optimized configuration of a decision tree can be modeled as follows(Safavian, Landgrebe et al. 1991):

$$\min_{T,F,d}\ p_e(T, F\ d) \qquad 3.1$$

subject to: Limited training sample size

In problem 3.1, *P_e* denotes total error probability, which is supposed to be minimized. *T* is the structure selection of the decision tree, *F* denotes feature subsets and d is decision rules of internal nodes.We can resolve problem 3.1 in two steps(Safavian, Landgrebe et al. 1991):

First Step: For a given *T* and *F*, find *d* = d* (T, F)* such that:

$$p_e(T, F, d^*(T, F)) = \min_d P_e(T, F, d) \qquad 3.2$$

Second Step: Find *T** and *F** such that:

$$P_e(T^*, F^*, d^*(T^*, F^*)) = \min_{T,F}\ p_e(T, F\ d^*(T, F)) \qquad 3.3$$

3.3.1.2 CART Algorithm

CART algorithm splits features into binary and it is inspired by Hunt's algorithm. Several splitting patterns are deployed to learn the most suitable split in this algorithm. (Priyam, Abhijeeta et al. 2013) This algorithm has the ability to work with both categorical and numerical variables. Furthermore, it recognizes and selects the important variables and excludes the ones that are not important. Moreover, it manages outliers efficiently. (Singh, Gupta et al. 2014)

CART is comprised of numerous single-variable splitting criteria: Gini, Symmetric Gini, Twoing, Ordered Twoing, Class Probability, Least Squares, and Least Absolute Deviation, one multi-variable splitting , the Linear Combinations approach. (Waheed, Bonnell et al. 2006)

3.3.1.3 Entropy

The impurity in a dataset is determined by an impurity function. Entropy is a common impurity function. (Jin and Agrawal 2003) If we have two classes, entropy can be defined as follows (Jin and Agrawal 2003) :

$$Entropy(D) = -p_1 \times \log p_1 - p_2 \times \log p_2 = -p_1 \times \log p_1 - (1-p_1) \times \log(1-p_1) \qquad 3.4$$

Where D denotes dataset, p_1 and p_2 denote the number of examples in class 1 and 2. If we split each node of the decision tree into the two subsets of D_L and D_R that individually denote the subsets of left and right, we would have the following equation (Jin and Agrawal 2003) :

$$g_c = g(D_L, D_R) = Entropy(D) - ((p_L \times Entropy(D_L)) + (p_R \times Entropy(D_R))) \qquad 3.5$$

Where g_c denotes the gain that is achieved by utilizing the predicate c for splitting, and p_L and p_R are the number of examples associated with D_L and D_R. Moreover, if we denote p_{1L} as the number of examples of class 1 in the node D_L, p_{1R} as the number of examples of class 2 in the node D_R, and do the respective assignments for p_{2R} and p_{2L}, the following equation can be achieved (Jin and Agrawal 2003) :

$$g_c = g(p_L, p_{1L}, p_{1R}) = Entropy(D) - p_L \times (-p_{1L} \times \log_{p1L} - (1-p_{1L}) \times \log(1-p_{1L}))$$
$$-(1-p_L) \times (-p_{1R} \times \log p_{1R} - (1-p_{1R}) \times \log(1-p_{1R})) \qquad 3.6$$

3.3.1.4 Gini

Another impurity criterion is Gini index that is defined as follows:

$$Gini\ Index = 1 - \sum_{i=1}^{n} (P_i)^2 \qquad 3.7$$

It is worth mentioning that both Entropy and Gini index criteria can be used to measure the impurity of a node in a decision tree. We have considered both of these two criteria separately to determine the one that gives higher accuracy.

3.3.1.5 Minimal cost complexity pruning

The cart algorithm fits a large decision tree to the data at first. Then, it computes different subsets of the decision tree T_i that are the reduced sized of the initial tree. A complexity measure is then linked to the quantity of the terminal nodes in order to calculate the subsets of the decision tree (T_i). In this procedure, the cost complexity metric ($R_\alpha(T)$), which is comprised of the two factors of size and error rate of the decision tree, will be minimized. This approach is called cost complexity pruning.(Prodromidis and Stolfo 2000)

The formula of $R_\alpha(T)$ is as follows(Prodromidis and Stolfo 2000):

$$R_\alpha(T) = R(T) + \alpha \, . \, C(T) \qquad 3.8$$

In this equation, the cost of misclassification is indicated by $R(T)$, and the complexity parameter is denoted by α. $C(T)$ denotes the complexity factor (quantity of terminal nodes in decision tree T) and α acts as its weight, that can control the pruning degree of the original tree T0. Smaller values of α correspond to smaller penalties for including more terminal nodes. If α rises, some of the terminal nodes will be eliminated by deploying the cost complexity pruning method and sebsets

(T_i) will be created. This algorithm is meant to find the optimal pruned decision tree. (Prodromidis and Stolfo 2000)

3.3.1.6 Determining the α rate

To determine the optimal α, we have made a comparison between different α rates and their corresponding accuracy. We have considered both gini and entropy criteria for determining the optimal α.

3.3.1.6.1 Gini index

Different accuracies of α rates are illustrated in figure 3.2. If we want to look at the figure in more details, table 3.3 represents some of the α rates with their associated accuracy. As a result, the highest accuracy belongs to the α rate of 0.000001183360580.

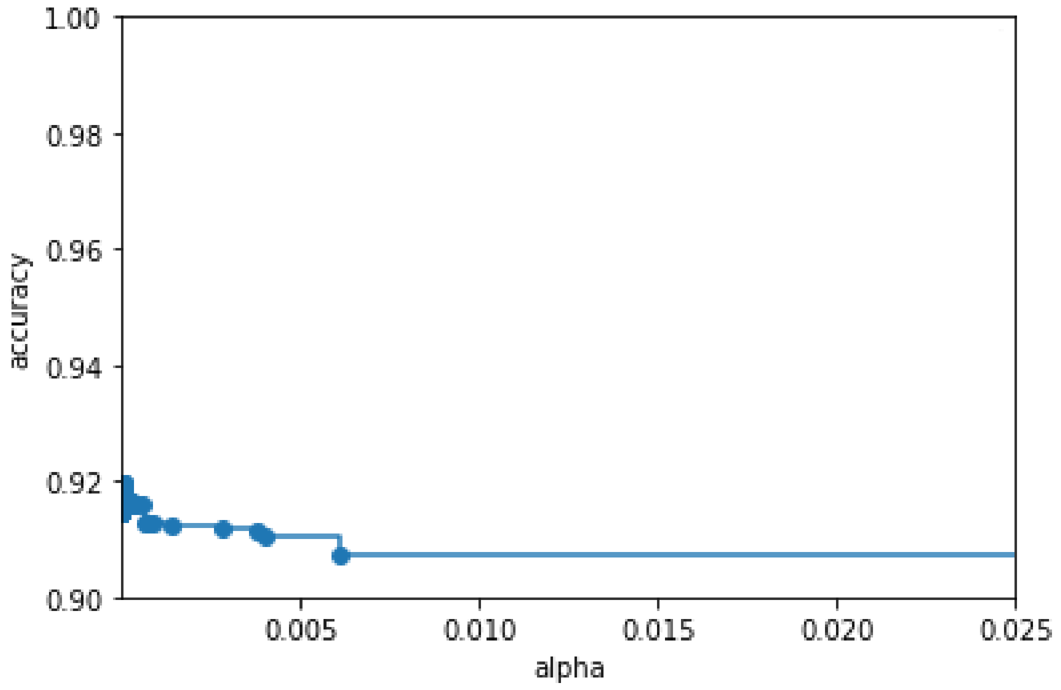

Figure 3.2 Alpha rates of gini

Table 3.3 Alpha rates of gini

α	Accuracy
0.000000000000000	0.914589958000000
0.000000203110832	0.914606034000000
0.000000407877801	0.916789110000000
0.000000544392004	0.918631382000000
0.000000680490005	0.919666655000000
0.000000815053546	0.919669870000000
0.000000951857172	0.919673085000000
0.000001075532360	0.919660225000000
0.000001183360580	0.919695591000000
0.000002634446890	0.919682731000000
0.000003418355050	0.919679515000000
0.000006053880610	0.919412659000000
0.000024682704700	0.918763202000000
0.000132813683000	0.916615492000000
0.000685397833000	0.912760546000000

3.3.1.6.2 Entropy

Figure 3.3 illustrates the α rate while utilizing the Entropy criterion. Furthermore, the detailed accuracy of different α rates are represented in table 3.4. The highest accuracy belongs to the α rate of 0.000004695108580.

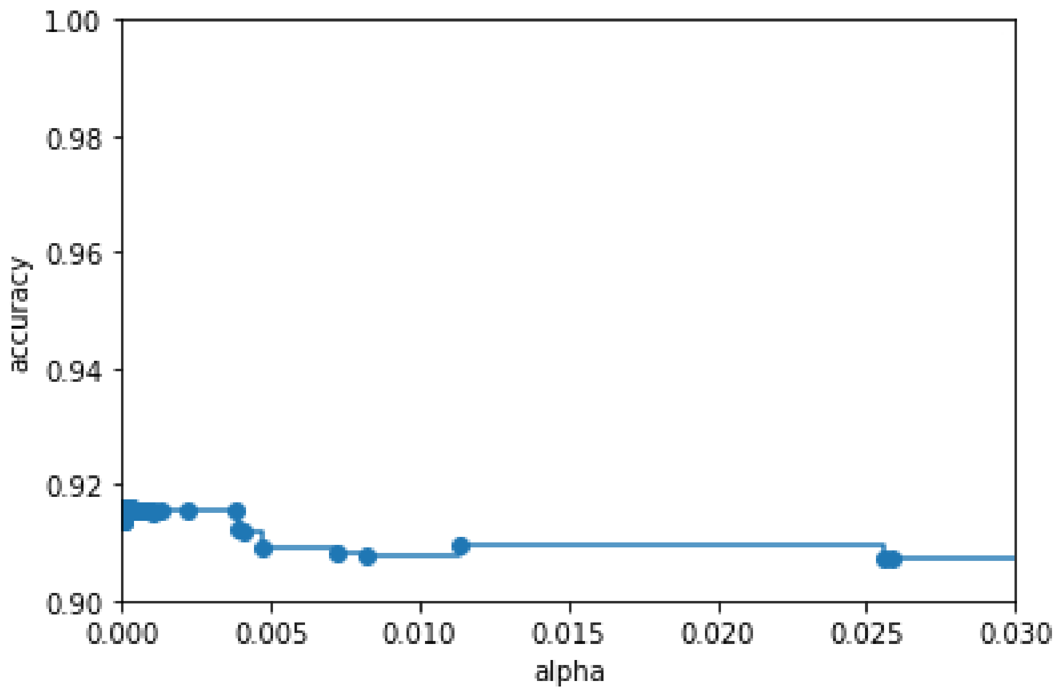

Figure 3.3 Alpha rates of entropy

Table 3.4 Alpha rates of entropy

α	Accuracy
0.000000000000000	0.915715255000000
0.000000163882258	0.915715255000000
0.000000351273644	0.915728115000000
0.000000425496497	0.915728115000000
0.000000818641446	0.915734546000000
0.000002565282890	0.915840645000000
0.000003638456290	0.915837430000000
0.000004695108580	0.916207170000000
0.000006082521780	0.914262014000000
0.000007949805040	0.914323102000000
0.000008842767350	0.914326317000000
0.000028315695200	0.914114118000000
0.000077553352100	0.914043385000000

3.3.1.7 Depth of the tree

In order to determine the best model with the highest accuracy, we have fitted different decision trees with different depths and criterion. We have considered the depths from 1 to 20 and the criteria of entropy and gini. Figure 3.4 represents the accuracy score of each tree from 2 to 20.

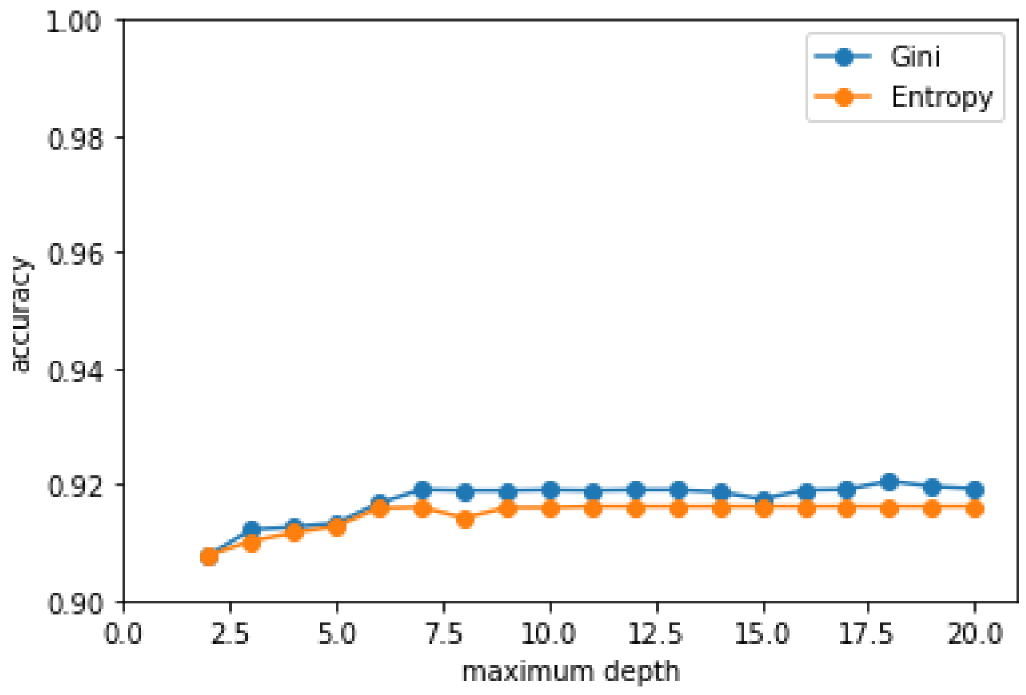

Figure 3.4 Comparison of different depths of decision tree

As it is represented in Figure 3.4, considering the gini criterion gives higher accuracy than the entropy. The accurate corresponding accuracies of different the maximum depths are represented in Table 3.5. As it is show, the decision tree with the maximum depth of 18 gives us the highest accuracy.

Table 3.5 Comparison of different depths of decision tree

Maximum Depth	Accuracy
1	0.7141
2	0.9078
3	0.9122
4	0.9126
5	0.9133
6	0.9168
7	0.9191
8	0.9189
9	0.9191
10	0.9191
11	0.9189
12	0.9191
13	0.9191
14	0.9187
15	0.9175
16	0.919
17	0.9191
18	0.9206
19	0.9197
20	0.9193

3.3.2 Random Forest

The Random Forest method has been praised as a powerful algorithm for classification and regression. (Biau and Scornet 2016) Random forest is an ensemble learning algorithm. Ensemble learning algorithms are based on the theory that using a number of classifiers will have a better performance compared to being individually used. (Rodriguez-Galiano, Ghimire et al. 2012) Random forest, as a bagging method, utilizes bootstrapping to create lots of trees. This algorithm does not suffer from the overfitting problem because if one permutation occurs, its associated classification error can be defeated by the whole permutations. As a result, if we have lots of trees, the generalization error will decrease. (Baudron, Alonso-Sarría et al. 2013)

Random Forest mixes numerous decision trees and takes the average of the predictions in order to achieve a reliable prediction. We can easily apply this algorithm to large-scale problems. Random Forest works by utilizing the “divide and conquer” rule. It samples parts of the data and develops a randomized tree predictor on each part. Then, it accumulates the predictions together. One of the important aspects of Random forest is its simplicity. Additionally, it provides high accuracy. This algorithm has the ability to work with real and large datasets. (Biau and Scornet 2016)

The advantages of Random Forest algorithm can be defined as follows (Rodriguez-Galiano, Ghimire et al. 2012):

- It has the ability to run efficiently when it comes to large databases.
- It has the ability to control many input variables.
- It provides approximations of essential variables in the classification.
- Generalization error can be estimated internally by this algorithm.
- It can determine outliers by calculating closeness between pairs of samples.
- It is moderately a strong and robust algorithm to outliers.
- It is not computationally expensive.

Furthermore, random forest is a powerful method that has shown promising results in some previous research such as (Zhang, Chen et al. 2021) and (Liu, Su et al. 2020). Considering the advantages of this method and its encouraging results in the previous work, we have been inclined to utilize it.

In order to reach an optimized random forest, we will consider different possible modes. First, we will implement a random forest method based on the gini index and then we will implement it considering the entropy.

3.3.2.1 Gini

3.3.2.1.1 Number of trees

We have analyzed the test score of different quantities of trees in the forest as represented in figure 3.5. The highest accuracy belongs to the forest with 1400 trees with the accuracy of 0.919381.

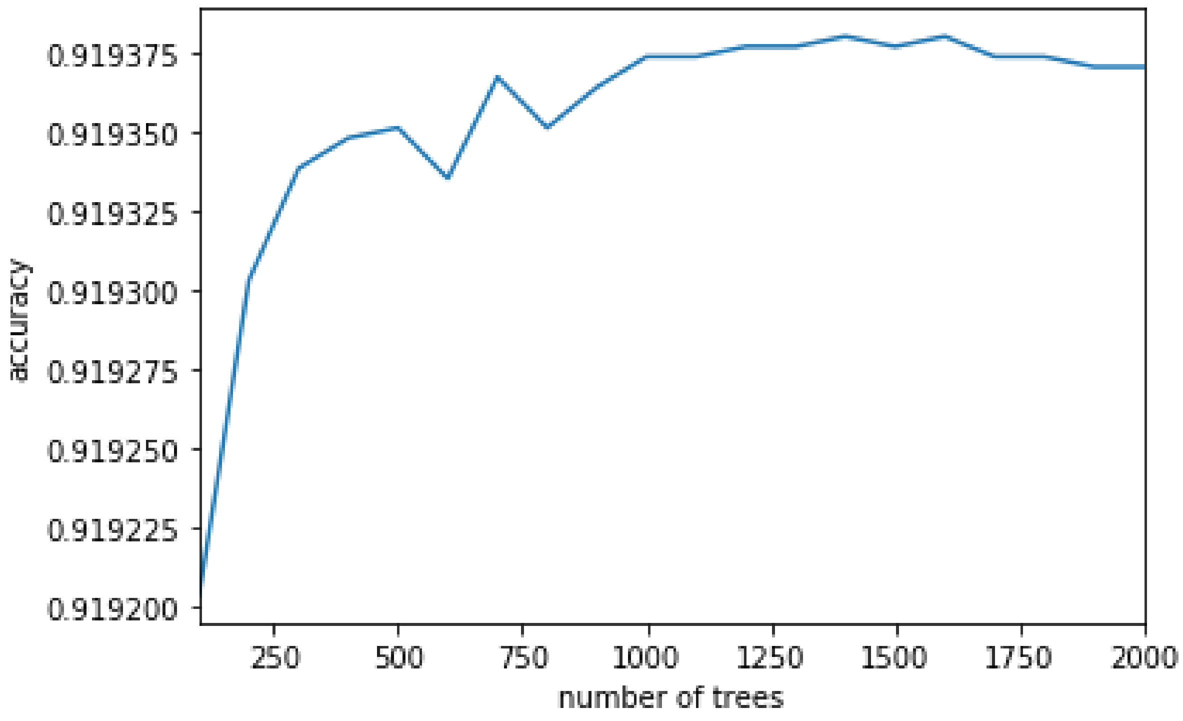

Figure 3.5 Number of trees vs accuracy considering gini

3.3.2.1.2 Maximum features in each split

We have illustrated forests with consideration of different fraction of features when searching for the best split in figure 3.6. As represented the highest accuracy belongs to the forest with the features fraction of 0.5 in each split with the accuracy of 0.919454.

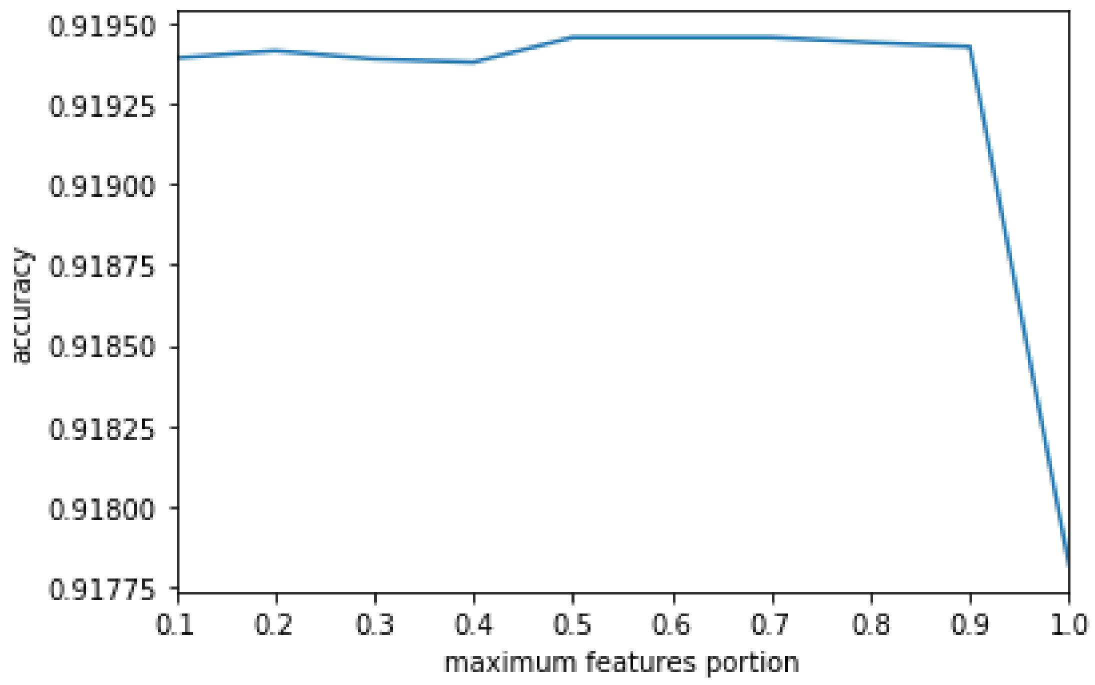

Figure 3.6 Maximum features portion vs considering gini

3.3.2.1.3 Minimum number of samples in a leaf

Forests with different minimum number of samples in a leaf are represented in figure 3.7. The highest accuracy belongs to the forest with 3 as the minimum number of samples in a leaf with the accuracy of 0.919824.

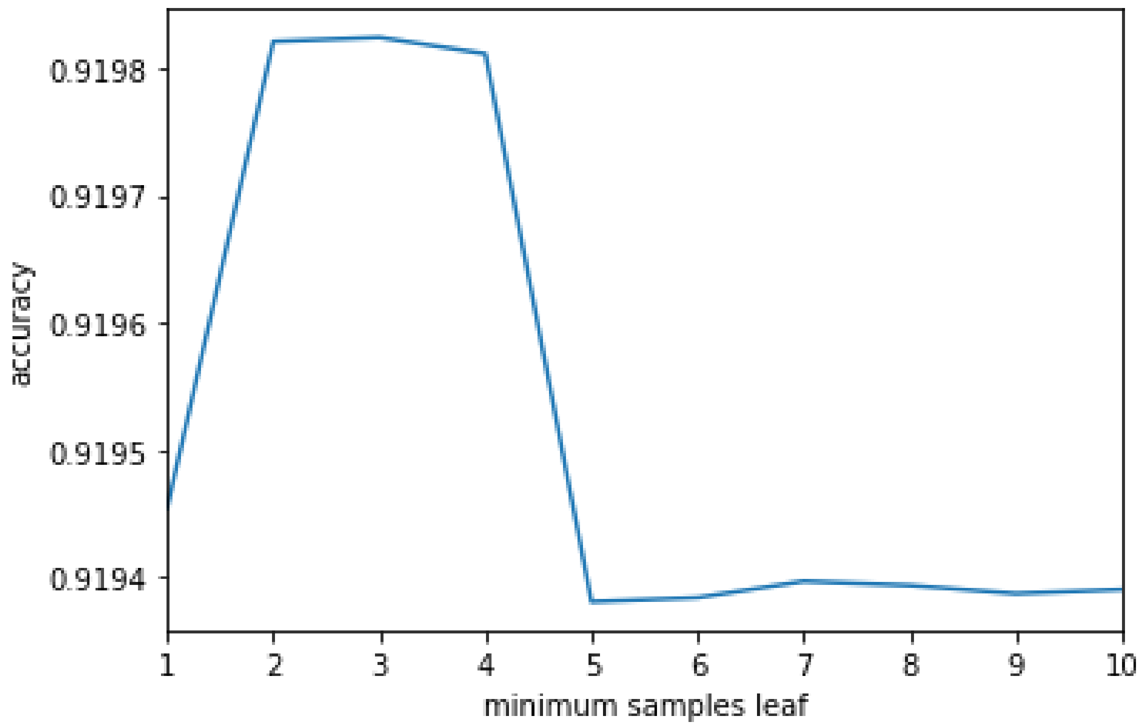

Figure 3.7 Minimum number of samples in a leaf vs accuracy considering gini

3.3.2.2 Entropy

3.3.2.2.1 Number of trees

The test score of different quantities of trees in the forest are represented in figure 3.8. The highest accuracy belongs to the forest with 900 trees with the accuracy of 0.919355.

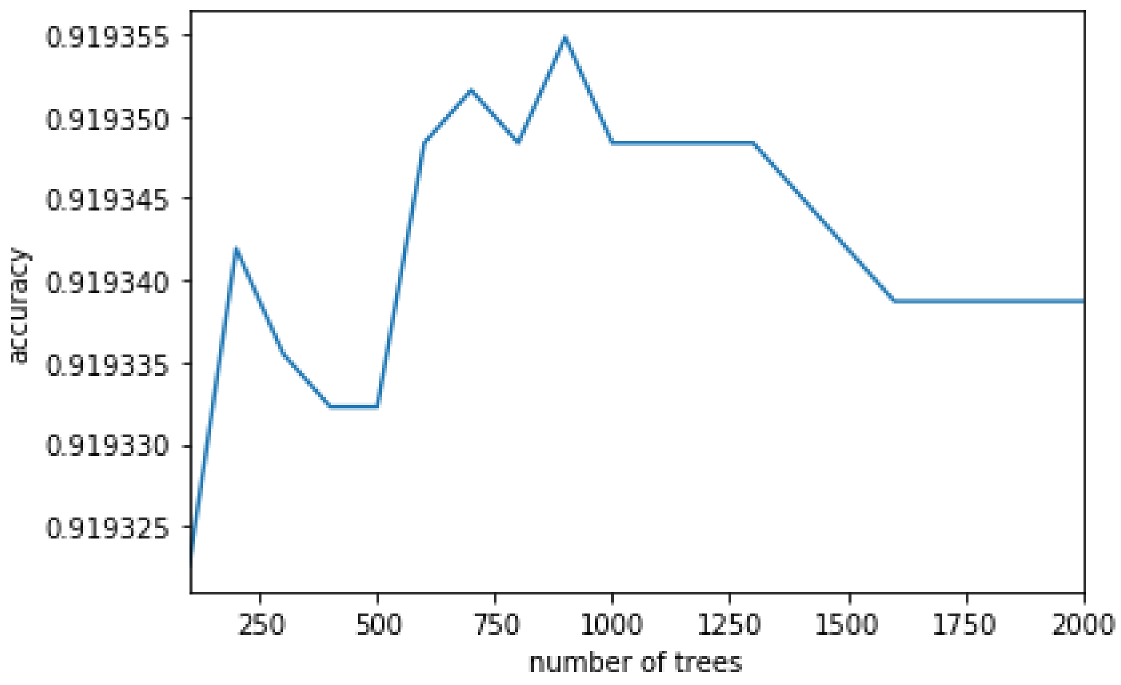

Figure 3.8 Number of trees vs accuracy considering entropy

3.3.2.2.2 Maximum features in each split

Forests with consideration of different fractions of features when searching for the best split are illustrated in figure 3.9. As represented, the highest accuracy belongs to the forest with the fraction of features of 0.4 in each split which gives the accuracy of 0.919393.

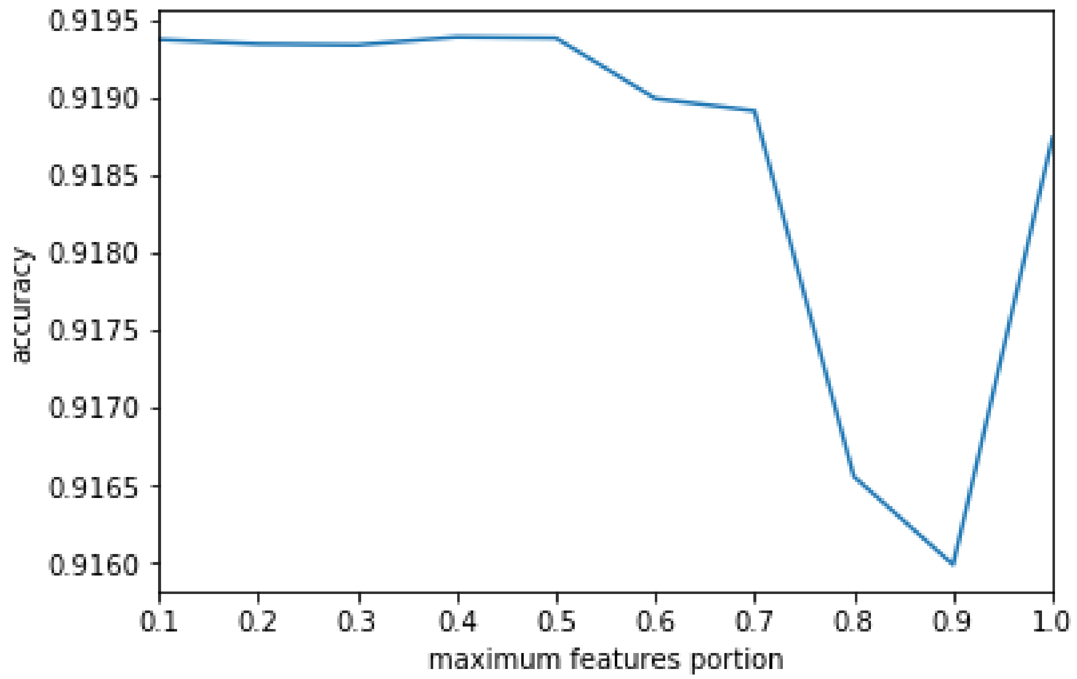

Figure 3.9 Maximum features portion vs considering entropy

3.3.2.2.3 Minimum number of samples in a leaf

We have analyzed Forests with different minimum number of samples in a leaf. As represented in figure 3.10, the highest accuracy belongs to the forest with 2 as the minimum number of samples in a leaf with the accuracy of 0.919432.

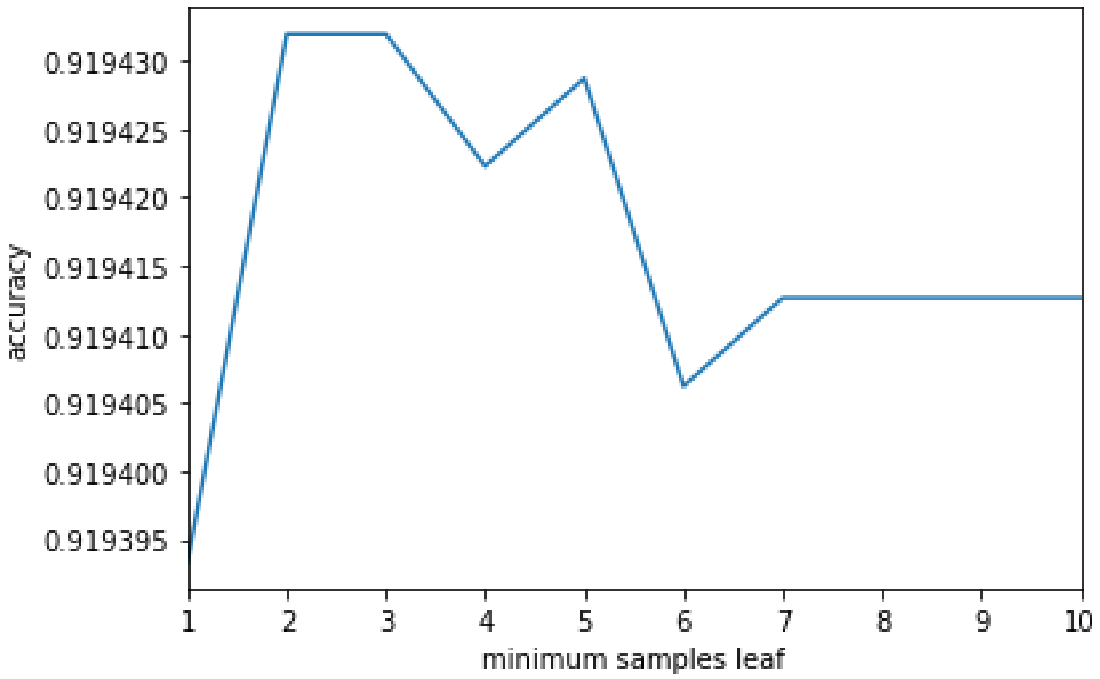

Figure 3.10 Minimum number of samples in a leaf vs accuracy considering entropy

As a result, we have two random forest models considering the gini and entropy criteria the accuracies of 0.9198 and 0.9194. Hence, considering the gini criterion gave us a higher accuracy. Therefore, the optimal random forest would be the one considering the gini criterion with 1400 trees, having the feature fraction of 0.5 in each split and considering 3 as the minimum number of samples in a leaf.

3.3.3 Deep Learning

Deep learning, as a subdivision of machine learning, utilizes artificial neural networks that are stimulated by the human brain. Deep learning has accomplished much higher than machine learning in different domains. Traditional classification approaches are supposed to be 0 or 1. However, deep learning presents results between 0 and 1 as well, which will result in higher accuracies. (Ertam and Aydın 2017)

Traditional machine learning methods had limited capabilities. They were not able to easily convert raw data into a proper representation to be considered as a suitable input for classification or detection. This necessitated significant expertise in a certain field. On the other hand, there exist some methods called representation learning in which we can give raw data to a machine so that it will automatically identify the necessary representations. Deep Learning, as a representation learning method, includes several representation levels. We can achieve these representations by creating non-linear modules that have the ability to transform the representation to higher levels. We can learn complicated functions through these transformations. If we have higher representation levels in the classification, this will lead to increasing the input characters that are meant to be essential for differentiation as well as defeat unnecessary differences. In the second layer, the themes will be detected without considering small differences. In the third layer, the themes will be aggregated and the following layers will mix these parts and do the detection task. The important thing about deep learning is the fact that a general learning method is utilized for learning these feature layers and they are not created by humans. (LeCun, Bengio et al. 2015)

Deep Learning can resolve the problem that previously could not be solved despite several efforts. It can detect complex patterns and can be applied to various domains. Deep learning has proven to have better performance compared to traditional machine learning methods in several fields such as image recognition, natural language understanding, language translation, question answering, speech recognition, reconstructing brain circuits and so on. (LeCun, Bengio et al. 2015)

3.3.3.1 Backpropagation

The backpropagation algorithm is the calculation of the objective function gradient concerning the weights. We can calculate the gradient of the objective function concerning the input by going backward concerning the output, which is illustrated in Figure 3.11. The backpropagation formula can be used frequently for all modules from the output to the input. We can easily calculate the gradients concerning the weights of each module when the gradients from the backpropagation formula are calculated. (LeCun, Bengio et al. 2015)

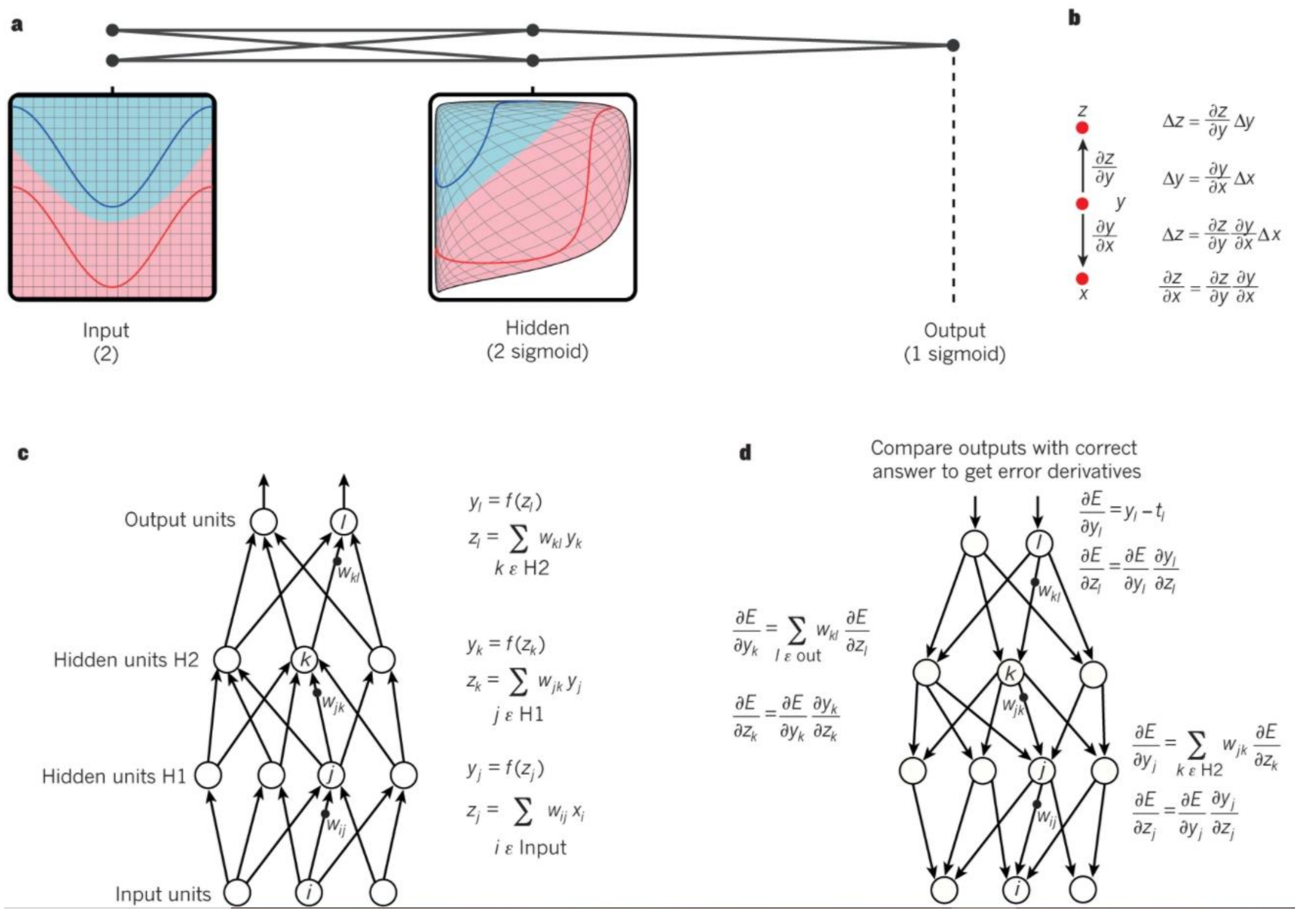
a
Input (2)
Hidden (2 sigmoid)
Output (1 sigmoid)
b
$\Delta z = \frac{\partial z}{\partial y}\Delta y$
$\Delta y = \frac{\partial y}{\partial x}\Delta x$
$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$
$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$
c
Output units
Hidden units H2
Hidden units H1
Input units
$y_l = f(z_l)$
$z_l = \sum_{k \in H2} w_{kl} y_k$
$y_k = f(z_k)$
$z_k = \sum_{j \in H1} w_{jk} y_j$
$y_j = f(z_j)$
$z_j = \sum_{i \in Input} w_{ij} x_i$
d
Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \in out} w_{kl}\frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk}\frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$

Figure 3.11 Deep neural network structure (LeCun, Bengio et al. 2015)

A feedforward neural network design learns to assign an input to an output. In this case, both input and output are fixed-size. In each layer, some units calculate a weighted sum of inputs and use a non-linear function for transforming the result. Currently, the most common non-linear function for deep learning is the rectified linear unit (ReLU) which is defined as f(z) = max(z, 0). Another function is tanh(z) or 1/(1 + exp(−z)). However, in deep neural networks comprised of many layers, ReLU has a higher speed in learning. Hidden layers are the ones that do not exist in the input layer or the output layer. As a matter of fact, hidden layers change the input by non-linear functions. Finally, the input will be linearly divisible in the final layer. (LeCun, Bengio et al. 2015)

3.3.3.2 Softmax

Softmax regression creates vector σ(*x*) from vector X while both vectors are K-dimensional. The formula of σ(*x*) is represented in equation 3.9 (Xu, Shen et al. 2018) :

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \qquad j = 1, \ldots, K \tag{3.9}$$

In multi-class classification, softmax is a common function to be used at the final layers, so that it could determine the class of each instance.

3.3.3.3 The proposed Deep Neural Network

Deep learning is one of the most powerful approaches that has achieved encouraging results in previous work such as (Shenfield, Day et al. 2018), (Wang, Zeng et al. 2021), (Khan, Karim et al. 2019), and (Zhiqiang, Mohi-Ud-Din et al. 2019). As a result, we were encouraged to build a deep neural network in our research as well.

The proposed structure of the deep neural network is as figure 3.12. There are 123 neurons in the input layer since we have 123 features after preprocessing. Furthermore, as we have 40 different instances (37 attacks in the test, 2 additional attacks in the training set and 1 normal instance), the output layer has 40 neurons. Through several experiments, we determined that adding three hidden layers with 121 neurons gives us the highest accuracy. Furthermore, we have chosen the Relu activation function in the input and hidden layer as it gave a higher accuracy over other activation functions such as Tanh in our experiments. The activation function of the output layer must be softmax since it is the only activation function for categorical outputs.

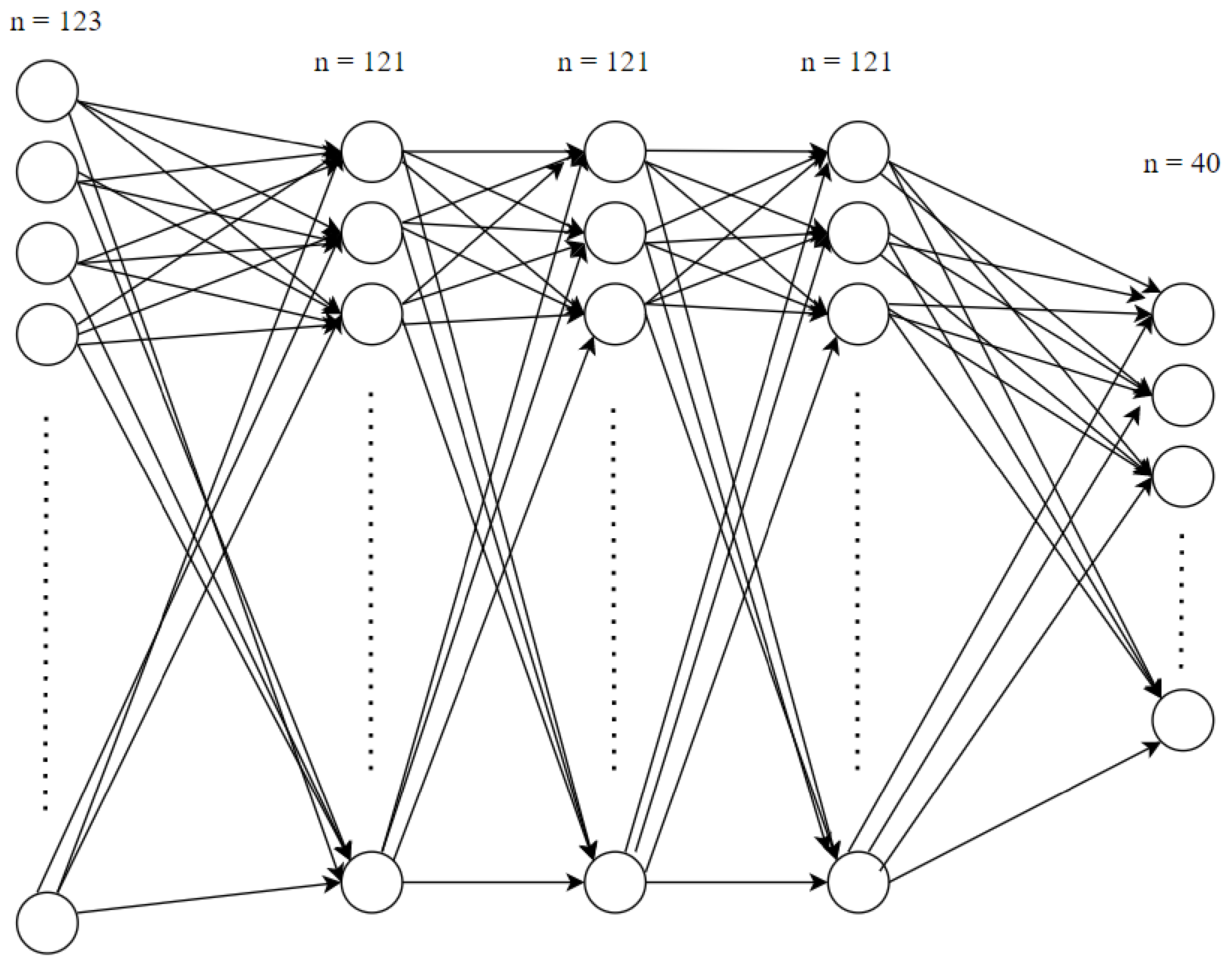

Figure 3.12 The proposed deep neural network

CHAPTER 4 EVALUATION OF THE METHODS

In this chapter, we will evaluate the models that we have proposed in chapter 3. We will introduce different evaluation criteria and compare the models.

4.1 Evaluation Criteria

The main evaluation criteria for machine learning methods is F-score. Another reliable criterion for evaluation is the Accuracy score. In multi-class classification, the f score and accuracy score give the same result. In order to calculate the F-score, we have to calculate precision and recall first. The equations of each of the mentioned criteria is as follows:

$$Precision = \frac{TP}{TP + TF} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1\,Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.3}$$

$$Accuracy\ Score = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

In these equations, *TP* denotes true positive, *TN* denotes true negative, *FP* denotes false positive and *FN* denotes false negative.

4.2 Evaluation

4.2.1 Decision Tree

Table 4.1 represents the confusion matrix of the proposed decision tree model.

Table 4.1 Confusion matrix of the decision tree model

	apache2.	back.	buffer_overflow.	ftp_write.	guess_passwd.	httptunnel.	imap.	ipsweep.	land.	loadmodule.	mailbomb.	mscan.	multihop.	named.	neptune.	nmap.	normal.	perl.	phf.	pod.	portsweep.	processtable.	ps.	rootkit.	saint.	satan.	sendmail.	smurf.	snmpgetattack.	snmpguess.	sqlattack.	teardrop.	udpstorm.	warezclient.	warezmaster.	worm.	xlock.	xsnoop.	xterm.
apache2.	0	475	0	0	0	0	0	0	0	0	0	0	0	0	9	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
back.	0	1091	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buffer_overflow.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd.	0	0	0	0	409	0	0	0	0	0	0	0	0	0	0	0	3957	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
httptunnel.	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	157	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep.	0	0	0	0	0	0	0	303	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
land.	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mailbomb.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mscan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	213	0	448	0	0	0	163	0	0	0	0	229	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
named.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
neptune.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57973	0	14	0	0	0	7	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	83	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
normal.	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	60305	0	0	31	63	0	0	0	0	151	0	0	0	0	0	39	0	0	1	0	0	0	0
perl.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	3	0	0	0	340	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
processtable.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	756	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ps.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
saint.	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	628	0	0	0	0	0	0	0	0	0	0	0	0	0
satan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	1630	0	0	0	0	0	0	0	0	0	0	0	0	0
sendmail.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164091	0	0	0	0	0	0	0	0	0	0	0
snmpgetattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7741	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snmpguess.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2406	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sqlattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
teardrop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0
udpstorm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster.	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	1586	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0
worm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xlock.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xsnoop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xterm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.2 illustrates the detailed classification report of each class in the dataset. Different measures of recall, precision, and f score in represented in the table. The proposed classifier has a good performance on detecting attacks with high number of instances, such as Neptune, Smurf, and Satan with the F1-scores of 0.9978, 1.0000, and 0.7604 respectively. Even on the attacks with low number of instances, such as pod and nmap, it still has a highly reasonable performance. The total Accuracy is 92.06%.

Table 4.2 Classification report of the decision tree model

	Precision	**Recall**	**F1-score**	**Support**
apache2.	0.0000	0.0000	0.0000	794
back.	0.6962	0.9936	0.8188	1098
buffer_overflow.	0.0000	0.0000	0.0000	22
ftp_write.	0.0000	0.0000	0.0000	3
guess_passwd.	1.0000	0.0937	0.1713	4367
httptunnel.	0.0000	0.0000	0.0000	158
imap.	0.0000	0.0000	0.0000	1
ipsweep.	0.7301	0.9902	0.8405	306
land.	1.0000	0.1111	0.2000	9
loadmodule.	0.0000	0.0000	0.0000	2
mailbomb.	0.0000	0.0000	0.0000	5000
mscan.	0.0000	0.0000	0.0000	1053
multihop.	0.0000	0.0000	0.0000	18
named.	0.0000	0.0000	0.0000	17
neptune.	0.9961	0.9995	0.9978	58001
nmap.	1.0000	0.9881	0.9940	84
normal.	0.7278	0.9952	0.8408	60593
perl.	0.0000	0.0000	0.0000	2
phf.	0.0000	0.0000	0.0000	2
pod.	0.7232	0.9310	0.8141	87
portsweep.	0.5893	0.9605	0.7304	354
processtable.	0.0000	0.0000	0.0000	759
ps.	0.0000	0.0000	0.0000	16
rootkit.	0.0000	0.0000	0.0000	13
saint.	0.0000	0.0000	0.0000	736
satan.	0.6142	0.9982	0.7604	1633
sendmail.	0.0000	0.0000	0.0000	17
smurf.	1.0000	1.0000	1.0000	164091
snmpgetattack.	0.0000	0.0000	0.0000	7741
snmpguess.	0.0000	0.0000	0.0000	2406
sqlattack.	0.0000	0.0000	0.0000	2
teardrop.	0.2353	1.0000	0.3810	12

udpstorm.	0.0000	0.0000	0.0000	2
warezclient.	0.0000	0.0000	0.0000	0
warezmaster.	0.0000	0.0000	0.0000	1602
worm.	0.0000	0.0000	0.0000	2
xlock.	0.0000	0.0000	0.0000	9
xsnoop.	0.0000	0.0000	0.0000	4
xterm.	0.0000	0.0000	0.0000	13

4.2.2 Random Forest

The confusion matrix of the results from the proposed random forest method is illustrated in table 4.3.

Table 4.3 Confusion matrix of the random forest model

	apache2.	back.	buffer_overflow.	ftp_write.	guess_passwd.	httptunnel.	imap.	ipsweep.	land.	loadmodule.	mailbomb.	mscan.	multihop.	named.	neptune.	nmap.	normal.	perl.	phf.	pod.	portsweep.	processtable.	ps.	rootkit.	saint.	satan.	sendmail.	smurf.	snmpgetattack.	snmpguess.	sqlattack.	teardrop.	udpstorm.	warezclient.	warezmaster.	worm.	xlock.	xsnoop.	xterm.
apache2.	0	475	0	0	0	0	0	0	0	0	0	0	0	0	94	0	225	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
back.	0	1098	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buffer_overflow.	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd.	0	0	0	0	131	0	0	0	0	0	0	0	0	0	0	0	4236	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
httptunnel.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	158	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep.	0	0	0	0	0	0	0	303	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
land.	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mailbomb.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mscan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	178	0	743	0	0	0	1	0	0	0	0	131	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
named.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
neptune.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	57992	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
normal.	0	3	0	0	0	0	0	2	0	0	0	0	0	0	0	0	60306	0	0	15	66	0	0	0	0	145	0	16	0	0	0	39	0	1	0	0	0	0	0
perl.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	349	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
processtable.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	756	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ps.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
saint.	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	631	0	0	0	0	0	0	0	0	0	0	0	0	0
satan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1632	0	0	0	0	0	0	0	0	0	0	0	0	0
sendmail.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164091	0	0	0	0	0	0	0	0	0	0	0
snmpgetattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7741	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snmpguess.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2406	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sqlattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
teardrop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0
udpstorm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1191	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	410	1	0	0	0	0
worm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xlock.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xsnoop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xterm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The classification measures of each class are represented in table 4.4. The random forest classifier can detect the attacks that have high sample well such as neptune, satan, etc. There exist some attacks that have zero measures such as apache2 which are because there were not included in the training dataset. The total accuracy is 91.98%.

Table 4.4 Classification report of the random forest model

	Precision	**Recall**	**F1-score**	**Support**
apache2.	0.0000	0.0000	0.0000	794
back.	0.6967	1.0000	0.8212	1098
buffer_overflow.	1.0000	0.0455	0.0870	22
ftp_write.	0.0000	0.0000	0.0000	3
guess_passwd.	1.0000	0.0300	0.0582	4367
httptunnel.	0.0000	0.0000	0.0000	158
imap.	0.0000	0.0000	0.0000	1
ipsweep.	0.7463	0.9902	0.8511	306
land.	1.0000	0.5556	0.7143	9
loadmodule.	0.0000	0.0000	0.0000	2
mailbomb.	0.0000	0.0000	0.0000	5000
mscan.	0.0000	0.0000	0.0000	1053
multihop.	0.0000	0.0000	0.0000	18
named.	0.0000	0.0000	0.0000	17
neptune.	0.9953	0.9998	0.9976	58001
nmap.	1.0000	1.0000	1.0000	84
normal.	0.7273	0.9953	0.8404	60593
perl.	0.5000	0.5000	0.5000	2
phf.	1.0000	0.5000	0.6667	2
pod.	0.8500	0.9770	0.9091	87
portsweep.	0.8329	0.9859	0.9030	354
processtable.	0.0000	0.0000	0.0000	759
ps.	0.0000	0.0000	0.0000	16
rootkit.	0.0000	0.0000	0.0000	13
saint.	0.0000	0.0000	0.0000	736
satan.	0.6410	0.9994	0.7810	1633
sendmail.	0.0000	0.0000	0.0000	17
smurf.	0.9999	1.0000	0.9999	164091
snmpgetattack.	0.0000	0.0000	0.0000	7741
snmpguess.	0.0000	0.0000	0.0000	2406
sqlattack.	0.0000	0.0000	0.0000	2
teardrop.	0.2353	1.0000	0.3810	12

udpstorm.	0.0000	0.0000	0.0000	2
warezclient.	0.0000	0.0000	0.0000	0
warezmaster.	0.5000	0.0006	0.0012	1602
worm.	0.0000	0.0000	0.0000	2
xlock.	0.0000	0.0000	0.0000	9
xsnoop.	0.0000	0.0000	0.0000	4
xterm.	0.0000	0.0000	0.0000	13

4.2.3 Deep Learning

The confusion matrix of the results from the proposed deep neural network method is presented in table 4.5.

Table 4.5 Confusion matrix of the deep neural network model

	apache2.	back.	buffer_overflow.	ftp_write.	guess_passwd.	httptunnel.	imap.	ipsweep.	land.	loadmodule.	mailbomb.	mscan.	multihop.	named.	neptune.	nmap.	normal.	perl.	phf.	pod.	portsweep.	processtable.	ps.	rootkit.	saint.	satan.	sendmail.	smurf.	snmpgetattack.	snmpguess.	sqlattack.	teardrop.	udpstorm.	warezmaster.	worm.	xlock.	xsnoop.	xterm.
apache2.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	0	629	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
back.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1098	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buffer_overflow.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd.	0	0	0	0	0	0	0	100	0	0	0	0	0	0	607	0	3657	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
httptunnel.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	158	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep.	0	0	0	0	0	0	0	217	0	0	0	0	0	0	0	0	86	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mailbomb.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mscan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	341	0	712	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	15	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
named.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	13	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
neptune.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56462	0	1539	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
normal.	0	0	0	0	0	0	0	38	0	0	0	0	0	0	7	0	59847	0	0	0	60	0	0	0	0	15	0	626	0	0	0	0	0	0	0	0	0	0
perl.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	7	0	28	0	0	0	221	0	0	0	0	97	0	0	0	0	0	0	0	0	0	0	0	0
processtable.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	759	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ps.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
saint.	0	0	0	0	0	0	0	93	0	0	0	0	0	0	35	0	51	0	0	0	0	0	0	0	0	557	0	0	0	0	0	0	0	0	0	0	0	0
satan.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	83	0	213	0	0	0	0	0	0	0	0	1337	0	0	0	0	0	0	0	0	0	0	0	0
sendmail.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164091	0	0	0	0	0	0	0	0	0	0
snmpgetattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7741	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snmpguess.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2405	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
sqlattack.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
teardrop.	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
udpstorm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1601	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
worm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xlock.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xsnoop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
xterm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	10	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0

Table 4.6 illustrates the classification report of the proposed deep neural network regarding different measures of precision, recall, and f score. The total accuracy is 90.72 percent. As the two previous methods, deep neural network provides high accuracy in detection and classification, especially on portsweep, neptune, smurf, and normal instances.

Table 4.6 Classification report of the deep neural network model

	Precision	**Recall**	**F1-score**	**Support**
apache2	0.0000	0.0000	0.0000	794
back	0.0000	0.0000	0.0000	1098
buffer_overflow	0.0000	0.0000	0.0000	22
ftp_write	0.0000	0.0000	0.0000	3
guess_passwd	0.0000	0.0000	0.0000	4367
httptunnel	0.0000	0.0000	0.0000	158
imap	0.0000	0.0000	0.0000	1
ipsweep	0.4759	0.7092	0.5696	306
land	0.0000	0.0000	0.0000	9
loadmodule	0.0000	0.0000	0.0000	2
mailbomb	0.0000	0.0000	0.0000	5000
mscan	0.0000	0.0000	0.0000	1053
multihop	0.0000	0.0000	0.0000	18
named	0.0000	0.0000	0.0000	17
neptune	0.9766	0.9735	0.9750	58001
nmap	0.0000	0.0000	0.0000	84
normal	0.6980	0.9877	0.8180	60593
perl	0.0000	0.0000	0.0000	2
phf	0.0000	0.0000	0.0000	2
pod	0.0000	0.0000	0.0000	87
portsweep	0.7782	0.6243	0.6928	354
processtable	0.0000	0.0000	0.0000	759
ps	0.0000	0.0000	0.0000	16
rootkit	0.0000	0.0000	0.0000	13
saint	0.0000	0.0000	0.0000	736
satan	0.6665	0.8187	0.7348	1633
sendmail	0.0000	0.0000	0.0000	17
smurf	0.9961	1.0000	0.9981	164091
snmpgetattack	0.0000	0.0000	0.0000	7741
snmpguess	0.0000	0.0000	0.0000	2406
sqlattack	0.0000	0.0000	0.0000	2
teardrop	0.0000	0.0000	0.0000	12
udpstorm	0.0000	0.0000	0.0000	2
warezmaster	0.0000	0.0000	0.0000	1602

worm	0.0000	0.0000	0.0000	2
xlock	0.0000	0.0000	0.0000	9
xsnoop	0.0000	0.0000	0.0000	4
xterm	0.0000	0.0000	0.0000	13

CHAPTER 5 DISCUSSION, CONCLUSION AND RECOMMANDATIONS

5.1 Discussion

Most of the previous work have classified the attacks considering the overall five categories (DoS, R2L, U2R, Probe, and normal). On the other hand, one paper has considered all of the 37 distinct attacks of the test set: (Rashid, Othman et al. 2017) Taking only five classes into consideration leads to missing some necessary information about the attacks. Therefore, it would be more beneficial to classify them to the most detailed classification to understand the attacks better which will assist us in facing and stopping them. This will result in detecting and classifying the attacks in the most possible detailed approach which is definitely more advantageous in every aspect and will support in identifying the exact name of attacks such as Smurf, Neptune, etc.

All of our proposed models provide higher accuracy than the existing literature. Our most accurate model, the decision tree model, has the accuracy of 92.06 percent, which is 14.41 percent higher than the result of the most recent similar paper that tested an algorithm on the same test dataset: (Rashid, Othman et al. 2017). The mentioned paper has achieved an accuracy of 77.65 percent considering all of the 38 instances of the test set as class labels including 37 attacks and one normal instance which is the same class definition as our research. Most of the papers in the literature such as have considered five classes including the overall categories as DoS, U2R, R2L, Probe, and Normal. Unlike most of the papers in the literature, we have considered 38 classes instead of the overall five classes. This will result in detecting and classifying the attacks in the most possible detailed approach which is definitely more beneficial in every aspect. As a result, we have achieved 14.41 percent higher accuracy which is the highest accuracy to the best of our knowledge.

5.2 Conclusion

In this research, we have presented three methods for detecting network intrusions as well as classifying them. Network intrusions have the ability to cause significant damage to different systems. Utilizing the presented methods will help us to detect the intrusions at the time of the beginning of the attack. This will lead to identifying the intrusions so soon in order to prevent damages notably. Furthermore, we have classified the intrusions. This classification helps us to identify and understand the category and type of the concurring intrusion. As a result, we will know the suitable approach to defeat the intrusion since we are aware of its type.

The three methods that we have used in this research are decision tree, random forest, and deep learning approach. The decision tree gives the best classification with an accuracy of 92.06%. The random forest method gives an accuracy of 91.98% and the deep neural networks give the maximum accuracy of 90.72%. To the best of our knowledge, these are the highest accuracy achieved so far while considering the individual attack classes of KDD-99. All of the three presented methods provide us with high accuracies while the decision tree method provides the most accurate model for this dataset. The mentioned models work well on identifying various attacks such as back, ipsweep, nmap, neptune, etc. Some of the attacks, such as worm attack, were not classified correctly because there were no similar instances in the training set. However, as mentioned earlier, the overall performance of the presented models is so high and the decision tree method has provided us with higher accuracy than the literature.

5.3 Limitations and Future work

In this research, we have trained and tested different models on KDD-99 dataset and we were able to detect and classify many intrusions with high accuracy. Generally speaking, machine learning models are highly dependent on dataset. Therefore, our proposed models may not achieve same accuracies on other datasets. For instance, one of the characteristics of KDD-99 is that it includes some intrusions in the test set while they do not exist in the training set. This issue have resulted in lowering the accuracy of our models. Other datasets may not have this characteristic and result in achieving higher accuracies. Moreover, other datasets may include some other types of them that are not included in the current dataset. Therefore, future work may consider other datasets to work on.

REFERENCES

Aboueata, N., et al. (2019). Supervised machine learning techniques for efficient network intrusion detection. 2019 28th International Conference on Computer Communication and Networks (ICCCN), IEEE.

Ahmad, I., et al. (2009). Application of artificial neural network in detection of DOS attacks. Proceedings of the 2nd international conference on Security of information and networks.

Aiken, J. and S. Scott-Hayward (2019). Investigating adversarial attacks against network intrusion detection systems in sdns. 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE.

Akasapu, S. J. n. J. (2017). "An Integrated Approach for detecting DDoS attacks in Cloud Computing." 258-261.

Al-Zewairi, M., et al. (2017). Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. 2017 International Conference on New Trends in Computing Sciences (ICTCS), IEEE.

Alsaadi, H. S., et al. (2020). Fast Binary Network Intrusion Detection based on Matched Filter Optimization. 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), IEEE.

Alzahrani, A. S., et al. (2020). "A novel method for feature learning and network intrusion classification." **59**(3): 1159-1169.

Andresini, G., et al. (2020). "Multi-channel deep feature learning for intrusion detection." **8**: 53346-53359.

Baudron, P., et al. (2013). "Identifying the origin of groundwater samples in a multi-layer aquifer system with Random Forest classification." **499**: 303-315.

Belouch, M., et al. (2018). "Performance evaluation of intrusion detection based on machine learning using Apache Spark." **127**: 1-6.

Biau, G. and E. J. T. Scornet (2016). "A random forest guided tour." **25**(2): 197-227.

Bulavas, V. (2018). Investigation of network intrusion detection using data visualization methods. 2018 59th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), IEEE.

Chen, F., et al. (2018). A feature selection approach for network intrusion detection based on tree-seed algorithm and k-nearest neighbor. 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), IEEE.

Choi, H., et al. (2019). "Unsupervised learning approach for network intrusion detection system using autoencoders." **75**(9): 5597-5621.

Corrêa, D. G., et al. (2017). An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. 2017 International Joint Conference on Neural Networks (IJCNN), IEEE.

Dahiya, P. and D. K. J. P. c. s. Srivastava (2018). "Network intrusion detection in big dataset using spark." **132**: 253-262.

Dong, Y., et al. (2019). Real-Time Network Intrusion Detection System Based on Deep Learning. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), IEEE.

El-Sappagh, S., et al. (2019). "Classification procedures for intrusion detection based on KDD cup 99 data set." **11**.

Ertam, F. and G. Aydın (2017). Data classification with deep learning using Tensorflow. 2017 international conference on computer science and engineering (UBMK), IEEE.

Gao, X., et al. (2019). "An adaptive ensemble machine learning model for intrusion detection." **7**: 82512-82521.

Ghanem, K., et al. (2017). Support vector machine for network intrusion and cyber-attack detection. 2017 sensor signal processing for defence conference (SSPD), IEEE.

Hamid, Y., et al. (2019). "Wavelet neural network model for network intrusion detection system." **11**(2): 251-263.

Hashemi, M. J. and E. Keller (2020). Enhancing Robustness Against Adversarial Examples in Network Intrusion Detection Systems. 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE.

He, H., et al. (2019). "A novel multimodal-sequential approach based on multi-view features for network intrusion detection." **7**: 183207-183221.

Hu, Z., et al. (2020). "A Novel Wireless Network Intrusion Detection Method Based on Adaptive Synthetic Sampling and an Improved Convolutional Neural Network." **8**: 195741-195751.

Ioannou, L. and S. A. Fahmy (2019). Network intrusion detection using neural networks on FPGA SoCs. 2019 29th International Conference on Field Programmable Logic and Applications (FPL), IEEE.

Iqbal, A., et al. (2019). "A Feed-Forward and Pattern Recognition ANN Model for Network Intrusion Detection." **11**(4).

Jabbar, M. A., et al. (2017). Cluster based ensemble classification for intrusion detection system. Proceedings of the 9th International Conference on Machine Learning and Computing.

Jiang, J., et al. (2018). RST-RF: a hybrid model based on rough set theory and random forest for network intrusion detection. Proceedings of the 2nd International Conference on Cryptography, Security and Privacy.

Jiang, L., et al. (2007). Survey of improving k-nearest-neighbor for classification. Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007), IEEE.

Jin, R. and G. Agrawal (2003). Efficient decision tree construction on streaming data. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.

Jo, W., et al. (2020). "Packet Preprocessing in CNN-Based Network Intrusion Detection System." **9**(7): 1151.

Kabir, M. R., et al. (2017). "A network intrusion detection framework based on Bayesian network using wrapper approach." **166**(4): 13-17.

Kanimozhi, V. and T. P. Jacob (2019). Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. 2019 International Conference on Communication and Signal Processing (ICCSP), IEEE.

Kavzoglu, T. J. E. M. and Software (2009). "Increasing the accuracy of neural network classification using refined training data." **24**(7): 850-858.

Khan, F. A., et al. (2019). "A novel two-stage deep learning model for efficient network intrusion detection." **7**: 30373-30385.

Khan, M. A., et al. (2019). "A scalable and hybrid intrusion detection system based on the convolutional-LSTM network." **11**(4): 583.

Khan, R. U., et al. (2019). An improved convolutional neural network model for intrusion detection in networks. 2019 Cybersecurity and cyberforensics conference (CCC), IEEE.

Krishnaveni, S., et al. (2021). "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing." 1-19.

LeCun, Y., et al. (2015). "Deep learning." **521**(7553): 436-444.

Lee, C.-H., et al. (2017). Machine learning based network intrusion detection. 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), IEEE.

Liu, Z., et al. (2020). "A Deep Random Forest Model on Spark for Network Intrusion Detection." **2020**.

Lopez-Martin, M., et al. (2017). "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot." **17**(9): 1967.

Malik, A. J. and F. A. J. C. C. Khan (2018). "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection." **21**(1): 667-680.

Marinova-Boncheva, V. J. p. o. E. C. and Robotics (2007). "A short survey of intrusion detection systems." **58**: 23-30.

Min, E., et al. (2018). "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest." **2018**.

Mirza, A. H. (2018). Computer network intrusion detection using various classifiers and ensemble learning. 2018 26th Signal Processing and Communications Applications Conference (SIU), IEEE.

Mirza, A. H. and S. Cosan (2018). Computer network intrusion detection using sequential LSTM neural networks autoencoders. 2018 26th signal processing and communications applications conference (SIU), IEEE.

Moghanian, S., et al. (2020). "GOAMLP: Network Intrusion Detection With Multilayer Perceptron and Grasshopper Optimization Algorithm." **8**: 215202-215213.

Mohammadpour, L., et al. (2018). "A convolutional neural network for network intrusion detection system." **46**: 50-55.

Musafer, H., et al. (2020). "An enhanced design of sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems." **9**(2): 259.

Najeeb, R. F. and B. N. J. A. J. E. A. S. Dhannoon (2018). "A feature selection approach using binary firefly algorithm for network intrusion detection system." **13**(6): 2347-2352.

Natesan, P., et al. (2017). "Hadoop based parallel binary bat algorithm for network intrusion detection." **45**(5): 1194-1213.

Nazir, A., et al. (2021). "A novel combinatorial optimization based feature selection method for network intrusion detection." **102**: 102164.

Pamukov, M. E., et al. (2018). Negative selection and neural network based algorithm for intrusion detection in iot. 2018 41st International Conference on Telecommunications and Signal Processing (TSP), IEEE.

Papamartzivanos, D., et al. (2018). "Dendron: Genetic trees driven rule induction for network intrusion detection systems." **79**: 558-574.

Peng, Y., et al. (2019). Evaluating deep learning based network intrusion detection system in adversarial environment. 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), IEEE.

Piplai, A., et al. (2020). NAttack! Adversarial Attacks to bypass a GAN based classifier trained to detect Network intrusion. 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), IEEE.

Priyam, A., et al. (2013). "Comparative analysis of decision tree classification algorithms." **3**(2): 334-337.

Prodromidis, A. L. and S. J. Stolfo (2000). "Cost Complexity Pruning of Ensemble Classifiers."

Qu, F., et al. (2017). An intrusion detection model based on deep belief network. Proceedings of the 2017 VI International Conference on Network, Communication and Computing.

Rajadurai, H., et al. (2020). "A stacked ensemble learning model for intrusion detection in wireless network." 1-9.

Rao, B. B., et al. (2017). "Fast kNN classifiers for network intrusion detection system." **10**(14): 1-10.

Rashid, O. F., et al. (2017). "A novel DNA sequence approach for network intrusion detection system based on cryptography encoding method." **7**(1): 183-189.

Rezvy, S., et al. (2018). Intrusion detection and classification with autoencoded deep neural network. International Conference on Security for Information Technology and Communications, Springer.

Rodriguez-Galiano, V. F., et al. (2012). "An assessment of the effectiveness of a random forest classifier for land-cover classification." **67**: 93-104.

Roshan, S., et al. (2018). "Adaptive and online network intrusion detection system using clustering and extreme learning machines." **355**(4): 1752-1779.

Sabhnani, M. and G. Serpen (2003). KDD Feature Set Complaint Heuristic Rules for R2L Attack Detection. Security and Management, Citeseer.

Safavian, S. R., et al. (1991). "A survey of decision tree classifier methodology." **21**(3): 660-674.

Sahu, A., et al. (2020). Data processing and model selection for machine learning-based network intrusion detection. 2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), IEEE.

Saia, R., et al. (2019). A Discretized Extended Feature Space (DEFS) Model to Improve the Anomaly Detection Performance in Network Intrusion Detection Systems. KDIR.

Sakr, M. M., et al. (2019). "An efficiency optimization for network intrusion detection system." **11**(10): 1.

Sakr, M. M., et al. (2019). "Network intrusion detection system based PSO-SVM for cloud computing." **10**(3): 22.

Saritas, M. M., et al. (2019). "Performance analysis of ANN and Naive Bayes classification algorithm for data classification." **7**(2): 88-91.

Shenfield, A., et al. (2018). "Intelligent intrusion detection systems using artificial neural networks." **4**(2): 95-99.

Singh, S., et al. (2014). "Comparative study ID3, cart and C4. 5 decision tree algorithm: a survey." **27**(27): 97-103.

Singla, A., et al. (2020). Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. Proceedings of the 15th ACM Asia Conference on Computer and Communications Security.

Stolfo, J., et al. (2000). "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection." 1-15.

Sukumar, J. A., et al. (2018). Network intrusion detection using improved genetic k-means algorithm. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE.

Sun, P., et al. (2020). "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system." **2020**.

Tan, M., et al. (2019). A neural attention model for real-time network intrusion detection. 2019 IEEE 44th Conference on Local Computer Networks (LCN), IEEE.

Tao, P., et al. (2018). "An improved intrusion detection algorithm based on GA and SVM." **6**: 13624-13631.

Thaseen, I. S., et al. (2020). Network Intrusion Detection using Machine Learning Techniques. 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), IEEE.

Timčenko, V. and S. Gajin (2017). Ensemble classifiers for supervised anomaly based network intrusion detection. 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE.

Tzotsos, A. and D. Argialas (2008). Support vector machine classification for object-based image analysis. Object-Based Image Analysis, Springer**:** 663-677.

Verma, A. and V. Ranga (2019). ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things. 2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU), IEEE.

Verma, A. K., et al. (2019). A Network Intrusion Detection Approach Using Variant of Convolution Neural Network. 2019 International Conference on Communication and Electronics Systems (ICCES), IEEE.

Verma, P., et al. (2018). Network intrusion detection using clustering and gradient boosting. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE.

Vigneswaran, R. K., et al. (2018). Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. 2018 9th International conference on computing, communication and networking technologies (ICCCNT), IEEE.

Vinayakumar, R., et al. (2017). Applying convolutional neural network for network intrusion detection. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE.

Waheed, T., et al. (2006). "Measuring performance in precision agriculture: CART—A decision tree approach." **84**(1-2): 173-185.

Wang, C.-R., et al. (2018). "Network intrusion detection using equality constrained-optimization-based extreme learning machines." **147**: 68-80.

Wang, X., et al. (2020). "A Network Intrusion Detection Method Based on Deep Multi-scale Convolutional Neural Network." **27**(4): 503-517.

Wang, Z., et al. (2021). "Deep Belief Network Integrating Improved Kernel-Based Extreme Learning Machine for Network Intrusion Detection." **9**: 16062-16091.

Wu, K., et al. (2018). "A novel intrusion detection model for a massive network using convolutional neural networks." **6**: 50850-50859.

Wu, P. and H. Guo (2019). LuNET: a deep neural network for network intrusion detection. 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE.

Wu, P., et al. (2020). Pelican: A deep residual network for network intrusion detection. 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), IEEE.

Wutyi, K. S. and M. M. S. Thwin (2016). Heuristic rules for attack detection charged by NSL KDD dataset. Genetic and Evolutionary Computing, Springer**:** 137-153.

Xiao, M. and D. Xiao (2007). Alert verification based on attack classification in collaborative intrusion detection. Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), IEEE.

Xiao, Y., et al. (2019). "An intrusion detection model based on feature reduction and convolutional neural networks." **7**: 42210-42219.

Xu, C., et al. (2018). "An intrusion detection system using a deep neural network with gated recurrent units." **6**: 48697-48707.

Xu, C., et al. (2020). "A method of few-shot network intrusion detection based on meta-learning framework." **15**: 3540-3552.

Xu, H., et al. (2018). Application of a distance-weighted KNN algorithm improved by moth-flame optimization in network intrusion detection. 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), IEEE.

Yang, H., et al. (2019). Deep-learning-based network intrusion detection for SCADA systems. 2019 IEEE Conference on Communications and Network Security (CNS), IEEE.

Yang, H. and F. J. I. A. Wang (2019). "Wireless network intrusion detection based on improved convolutional neural network." **7**: 64366-64374.

Yang, J., et al. (2018). Modified naive bayes algorithm for network intrusion detection based on artificial bee colony algorithm. 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), IEEE.

Yin, C., et al. (2019). "Enhancing network intrusion detection classifiers using supervised adversarial training." 1-30.

Yu, Y. and N. J. I. A. Bian (2020). "An intrusion detection method using few-shot learning." **8**: 49730-49740.

Yu, Y., et al. (2018). "A new network intrusion detection algorithm: DA-ROS-ELM." **13**(4): 602-612.

Yu, Y., et al. (2017). "Network intrusion detection through stacking dilated convolutional autoencoders." **2017**.

Zargar, G. R. and P. Kabiri (2009). Identification of effective network features to detect Smurf attacks. 2009 IEEE Student Conference on Research and Development (SCOReD), IEEE.

Zhang, B., et al. (2018). Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method. 2018 IEEE International Conference on Communications Workshops (ICC Workshops), IEEE.

Zhang, B. C., et al. (2017). "Network intrusion detection based on directed acyclic graph and belief rule base." **39**(4): 592-604.

Zhang, C., et al. (2021). "A Novel Framework Design of Network Intrusion Detection Based on Machine Learning Techniques." **2021**.

Zhang, H., et al. (2018). Real-time distributed-random-forest-based network intrusion detection system using Apache spark. 2018 IEEE 37th international performance computing and communications conference (IPCCC), IEEE.

Zhang, H., et al. (2018). An effective deep learning based scheme for network intrusion detection. 2018 24th International Conference on Pattern Recognition (ICPR), IEEE.

Zhang, J., et al. (2020). An Ensemble-based Network Intrusion Detection Scheme with Bayesian Deep Learning. ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE.

Zhang, Y., et al. (2019). "Network intrusion detection: Based on deep hierarchical network and original flow data." **7**: 37004-37016.

Zhiqiang, L., et al. (2019). Modeling Network Intrusion Detection System Using Feed-Forward Neural Network Using UNSW-NB15 Dataset. 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE), IEEE.

Zhou, Y., et al. (2020). "M-AdaBoost-A based ensemble system for network intrusion detection." **162**: 113864.