

A COMPARATIVE STUDY OF NEURAL NETWORK APPROACHES IN
NETWORK ANOMALY DETECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

MEHMET UĞUR ÖNEY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2019

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

Prof. Dr. Ali KARA
Director, The Graduate School of Natural and Applied Science

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Computer Engineering, Atılım University**.

Prof. Dr. Kamil İbrahim AKMAN
Head of Computer Engineering Department

This is to certify that we have read the thesis **A Comparative Study of Neural Network Approaches in Network Anomaly Detection** submitted by **MEHMET UĞUR ÖNEY** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of **Master of Science**.

Asst. Prof. Dr. Serhat PEKER
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Murat KOYUNCU
Dept. of Information Systems Engineering, Atılım University

Asst. Prof. Dr. Bilge SAY
Dept. of Software Engineering, Atılım University

Asst. Prof. Dr. Çiğdem TURHAN
Dept. of Software Engineering, Atılım University

Asst. Prof. Dr. Erol ÖZÇELİK
Dept. of Psychology, Çankaya University

Asst. Prof. Dr. Serhat PEKER
Dept. of Software Engineering, Atılım University

Date:

01.02.2019



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MEHMET UĞUR ÖNEY

Signature :

ABSTRACT

A COMPARATIVE STUDY OF NEURAL NETWORK APPROACHES IN NETWORK ANOMALY DETECTION

Öney, Mehmet Uğur

Computer Engineering

Supervisor : Asst. Prof. Dr. Serhat PEKER

February 2019, 62 pages

Network intrusion detection is an important research field, and artificial neural networks have become increasingly popular in this subject. Despite this, the research concerning comparison of artificial neural network architectures in the network intrusion detection is a relatively insufficient. To make up for this, this study aims to examine the neural network architectures in network intrusion detection to determine which architecture produces high accuracy and low false positive rate, and what are the effects of the architectural components such as optimization functions, activation functions, the momentum of the learning rate. For this purpose, we have generated 6480 neural networks and, we evaluated them KDD99 dataset and, near-real-time simulation environment. This thesis provides a roadmap to guide future research on network intrusion detection using artificial neural networks.

Keywords: artificial neural network, network anomaly, attack identification, intrusion detection

ÖZ

AĞ ANOMALİLERİNİN TESPİTİNDE KULLANILAN YAPAY SİNİR AĞLARININ KARŞILAŞTIRILMASI

Öney, Mehmet Uğur

Bilgisayar Mühendisliği

Tez Yöneticisi : Dr. Öğr. Üyesi. Serhat PEKER

Şubat 2019 , 62 sayfa

Ağ saldırı tespit sistemleri günümüz bilişim sistemlerinde kritik bir yer teşkil ederken önemli bir araştırma alanı olarak yükselmeye ve yapay sinir ağlarının kullanımı bu alanda giderek daha popüler hale gelmeye başlamıştır. Buna rağmen, bu alanda yapay sinir ağı mimarileri ve bu mimarilerin bileşen parametreleri hakkında kapsamlı bir karşılaştırmalı çalışmasının eksikliği vardır. Bu çalışmada, ağ saldırı tespit sistemleri alanında kullanılan yapay sinir ağları mimarileri ve bu mimarilerin bileşenleri olan optimizasyon fonksiyonları, aktivasyon fonksiyonları, öğrenme kat sayısı ve momentum değişiminin doğruluk ve hatalı uyarı üretme oranlarına göre kıyaslayarak ileride yapılacak olan mühendislik ve akademik çalışmalar için bir temel oluşturması amaçlanmıştır. Bu doğrultuda, 6480 adet yapay sinir ağı oluşturularak kıyaslama veri kümesi olarak kabul edilen KDD99 ve yakın gerçek zamanlı simülasyon ortamı yardımıyla her bir yapay sinir ağı değerlendirilmiştir. Bu tezin, yapay sinir ağları kullanılarak geliştirilecek ağ saldırı tespit sistemleri araştırmalarına rehberlik edecek bir yol haritası sağlayacaktır.

Anahtar Kelimeler: yapay sinir ağları, ağ anomali, atak tespit, saldırı tespit



*Dedicated to
my inspiring parents and sister,*

ACKNOWLEDGMENTS

This thesis would not have been possible without the inspiration and support of a number of wonderful individuals — my thanks and appreciation to all of them for being part of this journey and making this thesis possible. I owe my deepest gratitude to my supervisors Dr. Serhat Peker. Without his enthusiasm, encouragement, support and endless optimism this thesis would hardly have been completed.

I would also like thank to my thesis jury members Dr. Murat Koyuncu, Dr. Bilge Say, Dr. Çiğdem Turhan and Dr. Erol Özçelik for their advice and guidance in helping to shape this dissertation.

I am forever thankful to my colleagues for their friendship and support, and for creating a cordial working environment. Specially, I would also like thank to my colleague Volkan Nergiz for his helps.

Importantly, I would like to thank my beloved, Başak Akın for the support, advice, hours of struggling through this challenging process, and for all her love.

Finally, my deep and sincere gratitude to my mom Atiye Öney and my father Fazlı Öney for their continuous and unparalleled love, help and support. I am grateful to my sister Dr. Merve Dilara Öney, M.D. for always being there for me as a friend. I am forever indebted to my parents for giving me the opportunities and experiences that have made me who I am. They selflessly encouraged me to explore new directions in life and seek my destiny. This journey would not have been possible if not for them, and I dedicate this milestone to them.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vi
ACKNOWLEDGMENTS	viii
TABLE OF CONTENTS	ix
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF ALGORITHMS	xiv
LIST OF ABBREVIATIONS	xv

CHAPTERS

1	INTRODUCTION	1
1.1	Purpose of the Study and Research Questions	3
1.2	Significance of the Study	4
1.3	Thesis Organization	4
2	LITERATURE SURVEY	5
2.1	Network Intrusion Detection	5
2.2	Artificial Neural Networks	7

2.2.1	Artificial Neural Network Architectures	8
2.2.2	Parameters of an Artificial Neural Network	10
2.3	Usage of the Artificial Neural Networks in Intrusion Detection	13
3	METHODOLOGY	18
3.1	Network Intrusion Detection Model	20
3.2	Dataset	22
3.2.1	KDD99 Dataset	23
3.3	Data Preprocessing	24
3.3.1	Feature Extraction	25
3.4	Neural Network Generator	26
3.4.1	Network Attack Simulator	29
3.5	Software Used	31
3.6	Evaluation Measures	32
3.6.1	True Negative Rate	32
3.6.2	True Positive Rate	32
3.6.3	False Positive Rate	33
3.6.4	False Negative Rate	33
3.6.5	Precision, Recall, and F-Measure	33
3.7	Experiment	34
4	RESULTS	36
4.1	Results of the Architectural Comparison	36

4.2	Top 10 of the Neural Networks	39
4.3	Effects of the Parameters	42
5	DISCUSSION AND CONCLUSION	45
5.1	Discussion of Results	46
5.2	Limitations and Further Research	48
	CURRICULUM VITAE	62



LIST OF TABLES

TABLES

Table 2.1	List of common Neural Network architectures.	9
Table 2.2	Activation Functions	12
Table 2.3	Distribution of studies based on neural network architecture.	16
Table 3.1	List of Neural Network architectures in the NIDS domain.	21
Table 3.2	Common NIDS Benchmark Datasets	22
Table 3.3	Attacks listed in the KDD99 dataset	24
Table 3.4	Extracted features.	27
Table 3.5	Neural Network Architectures	28
Table 3.6	Confusion Matrix	32
Table 4.1	Top 10 rated network.	40

LIST OF FIGURES

FIGURES

Figure 2.1	Distribution of studies based on neural network architecture.	16
Figure 2.2	Distribution of studies based on dataset.	17
Figure 3.1	Overview of the methodology	19
Figure 3.2	Simulator network topology	30
Figure 4.1	Architecture average training time.	37
Figure 4.2	Architecture average detection time.	38
Figure 4.3	Architecture average False Positive Rate.	38
Figure 4.4	Architecture average Accuracy.	39
Figure 4.5	Top 10 Neural Network's accuracy.	41
Figure 4.6	Top 10 Neural Network's false positive rate.	41
Figure 4.7	Compression of optimizers.	42
Figure 4.8	Compression of learning decay rates.	43
Figure 4.9	Compression of dropout rates.	44
Figure 4.10	Hidden layer activation function performance counters.	44

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1	Back-propagation	13
-------------	----------------------------	----



LIST OF ABBREVIATIONS

Adam	Adaptive Moment Estimation
AEN	Auto Encoder Networks
ANN	Artificial Neural Network
API	Application Programming Interface
CAIDA	Center for Applied Internet Data Analysis
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRISP-DM	Cross Industry Standard Process for Data Mining
CSAS	Cyber Security Attack Simulator
DARPA	Defense Advanced Research Projects Agency
DDOS	Denial of Service Attack
EEN	Evolutionary Neural Network
FAR	False Alarm Rate
FFNN	Feed-Forward Neural Network
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GPU	Graphical Processing Unit
KDD	Knowledge Discovery and Data Mining
NIDS	Network Intrusion Detection System
NNG	Neural Network Generator
PRA	Platform Result Analyzer
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Decent
SOM	Self-Organizing Map
TCP	Transmission Control Protocol
TN	True Negative

TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate



CHAPTER 1

INTRODUCTION

Every passing year, malicious attacks and threats on the networks are increasing. As a result of this, protecting the systems and determining attacks are gaining importance. A single case of an intrusion can steal, damage or delete the valuable data from the computer system of an organization. Attacks which prevent access to the data can discredit the organizations, even more, lead to financial consequences and eventually end up losing the customer. On the other hand, stealing critical information of a private organization or governmental organization may cause huge losses. For these reasons, network intrusion detection systems (NIDS) are one of the essential parts of computer networks.

The conventional security system which is used in computer networks is starting to fail to detect sophisticated attacks and malware. Because of this reason, NIDS has become a vital component of the computer network security infrastructure. However, several issues are needed to be considered during the construction of a network intrusion detection system. Data collection, data preprocessing and recognition of the intrusion are the necessary parts of a network intrusion system. Among them, detection of an intrusion is the most indispensable. Finding the patterns of an intrusive action and classification capability of this action are the hearth of the system [102].

Creating a model from the data is not a simple process. The volume of the data and continuously changing attack behavior are the main difficulties in this domain. For this reason, researchers have been starting to consider machine learning [97]. One of the popular machine learning platform is artificial neural networks (ANN) [102]. ANNs have the ability to learn by example, and generalize from limited, noisy,

and incomplete data which is commonly encountered in the NIDS domain. Neural networks have also been applied in the wide spectrum of data-intensive applications [59].

However, finding the best suited ANN topology for a problem is an important issue for the researchers and the engineers [39]. An ANN topology contains critical components like architectural structure of the neural network, optimization algorithms, activation functions, learning momentum and prevention of overfitting. Finding the best combination of this component and testing in well-prepared benchmark data is a time-consuming process. In this research, we aim to create a guideline and performance comparison basis for the researchers and practitioners employing ANNs in NIDS field.

Network intrusion detection systems (NIDS) are the intrusion alarms of the computer network security [11]. These systems intend to monitor and inform the cybersecurity officers or sometimes take automatic actions like changing the particular rules of other prevention systems in the network. It should be considered that attacks on a network can be one of many different types. These attacks can be performed from outside of the computer network or can be done by a person who is a legitimate user of the system.

Determining the number of alarms generated by the NIDS and the accuracy of those alarms are essential. The number of false alarms affects the systems and cybersecurity officer' behaviors and also may damage the NIDS reliability and trustworthiness. A NIDS should be adapted to new kind of attacks like zero-day attacks¹. Consequently, there are two major principals which are anomaly detection and signature detection.

Signature-based detection heavily depends on the previous attacks structures. Simulated attacks or actualized attacks are examined and features of these attacks are extracted for future detections. NIDS monitors and compares the traffic with the known attacks' signatures. In this model, the false alarm rate of the system is low, and accuracy of the detection is high. However, different attack strategies or new kind of attacks can eliminate the NIDS and successfully conduct the intrusion without cre-

¹ A zero-day vulnerability is a computer-software vulnerability that is unknown to those who would be interested in mitigating the vulnerability.

ating an alarm notification.

On the other hand, anomaly detection based NIDS easily orient itself for the new kind of attack types. Anomaly detection based systems monitor the network traffic and create a basis for them as a normal. In the event of an intrusion, changes in the features of the normal traffic drive evidence of an attack — conversely, these kind of detection systems are known with their high false positive rate. Any changes in the usage behavior can trigger an alarm.

Artificial Neural Networks (ANN) can transfer either signature-based detection methodology or anomaly based detection methodology into practice, because of the structures and natures. The ANN is a universal function approximator as proven by the universal approximation theorem [39]. However, the proof is not constructive regarding the number of the neurons needed, the network topology and the other parameters that directly affect the performance of the network. Because of that finding, a domain-specific topology or a design for the network is a challenging problem.

In this research, we compare the artificial neural network architectures and its components to find an optimal solution for the NIDS domain.

1.1 Purpose of the Study and Research Questions

This research investigates the performance of the neural network architectures and their parametric combinations in the NIDS domain. A comparative experiment was performed to figure out each neural network architecture's performance counters, their limitations, and open issues. Accordingly, in this study, the following research questions were formed:

- RQ1. Which type of Neural Network architecture has better accuracy and lowest false positive rate in the NIDS domain?
- RQ2. Which kind of Neural Network architectures has the lowest training and detection cost without sacrificing performance criteria like accuracy and false positive rate?

- RQ3. What are the effects of the activation functions, dropout values, optimization functions, the momentum of the learning and depth of the network over the performance?

1.2 Significance of the Study

Deciding on an artificial neural network architectural topology and finding the proper parameters for a dedicated topic is a time-consuming process. This study provides a baseline by comparing ANN architectures and their components using a near-real-time simulation environment and KDD99 dataset which is an acceptable benchmark dataset for the NIDS domain.

The main contributions of this thesis are summarized as follows:

- A performance comparison of artificial neural network architectures in the NIDS domain.
- A performance comparison of artificial neural networks' components such as optimization algorithms, activation functions, momentum, and dropout.
- An evaluation of the ANN topologies in a near-real-time NIDS environment.

1.3 Thesis Organization

The rest of the thesis is organized as follow: In Chapter 2, neural network architectures and components and parameters which are used in creating a neural network are explained. Later, this chapter describes the Network Intrusion Detection Systems and their types. Furthermore, usage of the neural networks in the NIDS are investigated in that chapter. In order to address the research questions, an experiment test bench and a methodology were conducted in Chapter 3. In Chapter 4, experiment results are reported; and finally, the conclusion, implications, and limitations of the study and the suggested directions for further work are discussed in Chapter 5.

CHAPTER 2

LITERATURE SURVEY

2.1 Network Intrusion Detection

A network intrusion detection system (NIDS) is a software that automates the network intrusion detection methods [13]. Consequently, NIDS is an essential part of any network architecture. They provide a layer of monitoring for the systems and produce alerts when hostile traffic or suspicious activity is detected. NIDS architecture can be analyzed from various points such as where traffic was captured, the location of the system, measurements, and others. Nevertheless, the intrusion detection problem can be seen as a classification problem that assigns the type of attack or normal for the given network event. There are two major methods for detecting intrusions, signature-based (sometimes call misuse-based) and anomaly-based intrusion detection.

Signature-based detection is comparing network traffic data with a base model which created with previously detected intrusion events. This method has high prediction accuracy and low false positive rate for known threats. However, it is very ineffective at detecting unknown attacks. Most common methods used for signature-based detection are pattern recognition and rule implication.

On the other hand, the anomaly detection method uses a profile that describes normal traffic behavior. This profile is usually generated from legitimate traffic of the network. Any traffic activity that deviates from the normal profile is considered as an anomaly. This method can recognize unfamiliar intrusion. However, the most critical drawback of this method is the high false alarm rate. Common methods used in anomaly detection are statistical analysis and machine learning.

Various research about the taxonomy for NIDS were proposed in the literature [56] [11] [27]. Commonly accepted elements are summarized as follow;

- **Detection method:** Two main methods are considered for detection: Signature-based and anomaly-based.
- **Model acquisition:** Models are based on human supervisor knowledge or simulated environment generation.
- **Usage frequency:** Detection takes place on a near-real-time data (continuous monitoring) or a defined sized batch data (periodic monitoring).
- **Architecture:** Position of the NIDS in a network architecture can be in a single point for the centralization or several junction points for distributed collection.

Lastly, measuring the performance of NIDS can be summarized in a five main counter [56]. These counters gives information about feasibility for the perception of NIDS performance in an actual deployment environment.

- **Prediction accuracy:** An indicator for the quality of detection.
- **False positive rate:** An indicator for the trustworthiness of the system.
- **Processing time:** Needed time to process the data and detect the intrusion.
- **Adaptability:** An indicator for the detecting new type of attacks or metamorphosed attacks.
- **Resource consumption:** An indicator for usage of the system resource like memory, CPU, and bandwidth.

Over the years, network intrusion detection systems have been developed. Machine learning methods have been particularly effective in detecting intrusions, including techniques such as fuzzy sets [54][29][91][1], artificial immune system [44][95][24], support vector machines (SVMs) [72] [90][20][38], or k-nearest neighbour algorithm (k-NN) [64][61][87][2].

The current state of the network intrusion detection systems has been recently surveyed for anomaly detection. Recent surveys [73][55][18][11][45] conclude that ANNs remain highly popular with researchers and have recently shown promising potentials for low false positive rate, especially when equipped with advanced techniques such as auto encoder network.

2.2 Artificial Neural Networks

The artificial neural network is a computing system that is inspired by biological neural networks. The biological neural network is an interactive system of neurons, which defines a recognizable pathway. An ANN can be described as a collection of interconnecting processing units in a topology. ANN's each processing unit is called an artificial neuron. Every neuron contains a step function as an activation function. These functions are also called the transfer functions. These functions are an abstraction of representing the action potential in the real neuron. This activation function can decide whether an input vector of the number belongs to a specific class or not. In ANN, most of the activation functions are nonlinear functions. This configuration allows the network to compute nontrivial problems using a small number of neurons. The topological structure of the neurons affects the ANN's performance and learning capabilities.

ANNs are parallel computing systems consisting of a considerable amount of simplistic processors with multiple interconnections [42]. ANN is a universal function approximator as proven by the universal approximation theorem [39]. An ANN can be describes as a weighted directed graph. Each node connects each other with weighted edges. Every node in the graph has a binary threshold unit as a computational model which is proposed by McCulloch et al. [67]. Each unit computes the weighted sum of the input edges and generates an output value between 0 and 1. This binary threshold unit calls the activation function.

Learning ability of an ANN can be described as updating graph edges weights so that a created network can efficiently execute a particular task. This process occurs in the training phase. Weighs are updated according to available patterns in the training

data. Learning methodologies can be grouped by the following three main categories: supervised, unsupervised and semi-supervised also sometimes called hybrid learning. In supervised learning, the network is provided the correct answers alongside associated training data. After each calculation, weights are updated according to the distance between the actual response and accurate answers. This distance function is called optimization function. In unsupervised learning, the network does not need any correct answers for training. It extracts the underlying structure of the data and correlations between patterns of the data. Then these patterns and structures collect into categories. Semi-supervised learning uses both methodologies. They are gathered the correlation patterns into the categories and acknowledged these categories with correct answers or a supported data.

2.2.1 Artificial Neural Network Architectures

An artificial neural network architecture or topology defines the connection way of neuron in a network [34]. These interconnections can be defined with the following four properties:

- the type of the connection,
- the order of the connection,
- whether the structure is fully connected or not,
- symmetry¹ of the connection.

Neurons cluster into layers in a neural network. Each cluster is a collection of the node that has similar behavior, function or a hierarchy. The stack of the layers creates a neural network. The connection between the layers can discriminate four types:

- interlayer connection is a relationship between adjacent layers,
- intralayer connection is a relationship between the same layers,

¹ A symmetry of an object is a transformation that leaves certain properties of that object intact.

- self-connection is a connection which has the same origin point and termination point,
- super-layer connection is a relationship between the neuron which is higher have interlayer or intralayer connection.

Common neural network architectures are listed in Table 2.1.

Table 2.1: List of common Neural Network architectures.

#	Neural Network Architecture	Abbreviation
1	Convolutional Neural Network	CNN
2	Recurrent Neural Network	RNN
3	Feed Forward Neural Network	FFNN
4	AutoEncoder Networks	AEN
5	Self-Organizing Map	SOM
6	Evolutionary Neural Network	EEN

Convolutional Neural Network features pooling layers, and each interconnected layer is a composition of convolutional cells [58]. Every segment of the network serves a different purpose. Convolutional cells proceed input data, and pooling layers simplify it with non-linear functions. Pooling layers tend to shrink when the network goes deep. This method reduces unnecessary features in the input data.

On the other hand, Recurrent Neural Networks are not a stateless network [32]. They have intralayer connections and have links between passes, relationships through time. Neurons are fed data from the previous layer and also from themselves from the previous state with self-connection. Because of the structure, the order of the data matters.

One of the most basic neural network architecture is the Feed-Forward Neural Networks. FFNN feeds data from the input layer to the output layer [80]. FFNNs are mostly designed as fully connected graphs, and they have an interconnection between adjacent layers.

Autoencoders are slightly similar to Feed-Forward Neural Networks as a topology, but fundamentally they have different architectures. Autoencoders are encoding the input data into the hidden layers and decoding the data from the hidden layers to the

output layer [17]. With this capability, Autoencoder networks can compress the high dimensional input data into the low dimensional form. Autoencoder networks' decoding layers' weights the same as encoding layers' weights. Because of this reason, they have a symmetric network structure.

A self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. In SOM, the input is presented to the network, after which the network evaluates which of its neurons most nearly match that input. Then selected neurons are adjusted according to input value for better fitting [53]. This process drags the neuron in a cluster.

Evolutionary Neural Network or in other term, Neuroevolution is a composition of neural networks and genetic algorithms [9]. The basic idea behind the neuroevolution is finding the best fitted neural network topology and parameters for the domain with the help of genetic algorithms [89]. The genetic algorithm creates a batch of a neural network and decides which of the neural network best fits the problem using various fitness functions.

2.2.2 Parameters of an Artificial Neural Network

ANN is a universal function approximator as proven by the universal approximation theorem [39]. However, the proof is not constructive regarding the number of the neurons needed, the network topology and the other parameters that directly affect the performance of the network. Because of that finding, a domain-specific topology or a design for the network is a challenging problem.

One of the parameters of an artificial neural network is the activation functions. The activation function is a binary threshold unit which computes the weighted sum of the input edges and generates an output value according to the behavior of the function. Mathematically,

$$y = \theta \left(\sum_{j=1}^n w_j x_j - u \right), \quad (2.1)$$

where $\theta(\cdot)$ is a unit step function and w_j is the edges' weight. Linear activation functions, logistic functions and Gaussian function are the most frequently used activation function families in a neural network [42]. Activation functions which are chosen for this research are listed in Table 2.2.

One of the other essential components of a neural network is the optimization algorithm. The optimization algorithm subsequently repeats propagation and weight update phases. When the input vector enters the network, computed output is compared with the expected result by using a loss function (2.2). The loss function over n training can be formulated as an average of losses over individual examples.

$$E = \frac{1}{2n} \sum_x \|(y(x) - y'(x))\|^2 \quad (2.2)$$

The resulting difference which is called an error value is calculated for each of the neuron's output value. Then these values are propagated (2.3) from the end of to network through to entering layer. This process continues until each neuron updated with the new value according to the error value.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \quad (2.3)$$

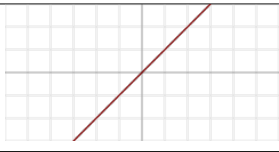
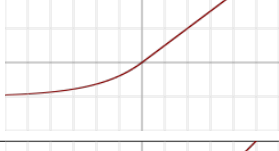
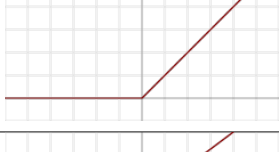
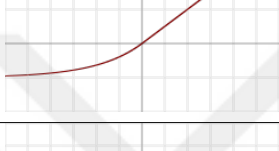
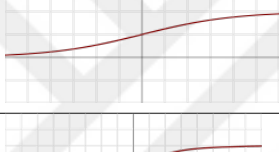
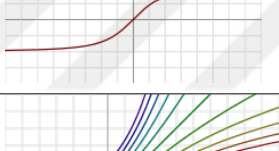
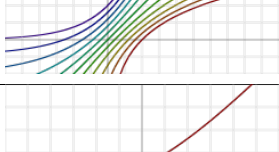
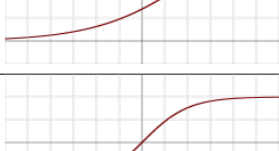
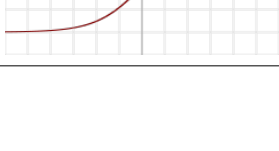
Algorithmic representation of back-propagation can be shown as Algorithm 1.

Most common optimization functions in the artificial neural network are Stochastic Gradient Descent, Adam Optimizer, and their variants.

Stochastic Gradient Descent (SGD) is an iterative stochastic approximation of gradient descent [69]. The function is called stochastic because the sampling system in the optimizer is selected randomly.

Adam is another important optimization function in artificial neural networks. Adaptive Moment Estimation is developed on the RMSProp optimizer [51]. This function

Table 2.2: Activation Functions

Name	Plot	Equation
linear		$f(x) = x$
elu		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
relu		$f(\alpha, x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
selu		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
sigmoid		$f(x) = \sigma(x) \frac{1}{1+e^{-x}}$
softsign		$f(x) = \frac{x}{1+ x }$
softmax		$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$
softplus		$f(x) = \ln(1 + e^x)$
tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$

computes running averages of both the gradients and the second moments of the gradients.

As an addition to an optimization functions, Momentum value can be added to both optimization function. Momentum value prevents oscillation which is caused by large updates in the optimization functions [81].

Algorithm 1: Back-propagation

Data: $ProblemSize, InputPatterns, iterations_{max}, learn_{rate}$

Result: Updated $Network$

$Network \leftarrow ConstructNetworkLayers();$

$Network_{weights} \leftarrow InitializeWeights(Network, ProblemSize);$

for $i = 1$ to $iterations_{max}$ **do**

$Pattern_i \leftarrow SelectInputPattern(InputPatterns);$

$Output_i \leftarrow ForwardPropagate(Pattern_i, Network);$

$BackwardPropagateError(Pattern_i, Output_i, Network);$

$UpdateWeights(Pattern_i, Output_i, Network, learn_{rate});$

end

return $Network$

Lastly, Dropout is a regularization method for decreasing overfitting in neural networks by preventing memorizing on training data [86]. Dropout applies to shoot out cells (except output layer) in an artificial neural network.

2.3 Usage of the Artificial Neural Networks in Intrusion Detection

The research on intrusion detection began with Denning's literature [28]. In [28], the author introduced a model for a real-time intrusion detection expert system that aims to detect a wide range of security violations ranging from attempted break-ins by outsiders to system penetrations and abuses by insiders, which laid the foundation of intrusion detection system. Later, many research efforts on anomaly detection have been carried out using various techniques.

Ding et al. [30] used the Adam algorithm in their model for preventing the detection model falling into the local minimum and speeding up the training phase. They performed the test and training with KDD99 dataset. Their proposed model achieved the training accuracy of 95.57%. However, there is no significant improvement in the detection of rare attack types.

Ghanbari et al. [36] proposed a model for detecting the distributed denial of service attack (DDOS). The proposed neural network model use convolutional neural net-

work architecture and is trained and tested with the Center for Applied Internet Data Analysis (CAIDA) dataset. They reach the true positive rate of 80.77%; however, they concluded that their model is based on supervised learning, which is not suitable for real-world applications.

Liu et al. [65] also applied the convolutional neural network for the intrusion detection domain. In their research, they used the KDD99 dataset for the training and test their models. They used a static convolutional neural network model to compare the various activation methods such as sigmoid, tangent hyperbolic and rectified linear unit. Study shows that a neural network model that used the sigmoid as an activation function performs better than other activation functions.

Tran et al. [96] show that a convolutional neural network with large-scale raw input data provided decent experimental results from a straightforward yet minimalistic architecture. Nevertheless, their study did not compare this result with another neural network architecture like multi-layered feed forward neural network.

Farahnakian et al. [33] evaluated the performance of the autoencoder in network intrusion detection with KDD99 dataset. Also, they investigated the effects of the hidden layer count on the proposed system performance. Their experiment resulted with 94.01% accuracy rate. However, their research does not give information about the detailed structure of the generated neural network, and they used only 10% of the KDD99 dataset.

Powers et al. [78] present a hybrid artificial immune system using SOM network for network intrusion detection. They used the KDD99 dataset in their research. In their research, they classified just a generic type attack individually rather than using and classifying the whole attack families. In addition, they train their networks with 28 features.

Wang et al. [101], created another hybrid artificial neural network for the NIDS. In this research, they compared feed-forward neural networks with recurrent neural networks and also with their system. Their comparison does not give information about the compared neural network structures and architectural differences or parameters.

One of the comprehensive comparison study was done by Kwon et al. [55] in 2017.

They compared the structure of the autoencoder network with recurrent neural networks. This research pointed out that the autoencoder detection accuracy is higher than the RNN network in a static test dataset.

Chiba et al. [21] have another important study about the comparison of neural network architecture. In this study, they generate 48 neural networks by changing the components parameters and compare them in a KDD99 environment. However, their research is limited with the feedforward neural networks.

Bontemps et al. [16] created a recurrent neural network for the NIDS domain. They also used KDD99 dataset. However, they prepared their dataset as a time series. Their research showed that the RNN with time series prediction rate is higher than the RNN with the static dataset.

In this study's literature review, we applied a review process based on the systematic literature review framework defined by Kitchenham [52]. According to this framework, our review was performed in three phases: planning, conducting and reporting, and for the planning phase, the corresponding articles published between years 2008 and 2018 were searched in the popular electronic databases. Forward and backward snowballing [43] was also used for extending the research.

At the end of our the literature review, distribution of research papers by neural network architectures are illustrated in Figure 2.1 and listed in Table 2.3.

Among neural network architectures, neural networks (18 out of 43 research papers or 42%) find to be the most widely used methods in the NIDS field. SOM and Feed Forward Neural Network methods (10 out of 43 research papers and 5 out of 43 research papers respectively) are other popular neural network architectures applied in NIDS. It is inferred that although many research papers were published, few of them (3 out of 43 research papers or approximately 7% for each) employed RNN, CNN and, Hybrid Neural Network models.

On the other hand, the distribution of articles by datasets is represented in Figure 2.2. It is apparent that the KDD99 dataset is the most commonly used one in NIDS field. It has been used in 19 (44%) out of 43 articles in total. Following are NSL-KDD and DARPA'98 datasets which have been used in 12 (28%) and 2 (5%) articles

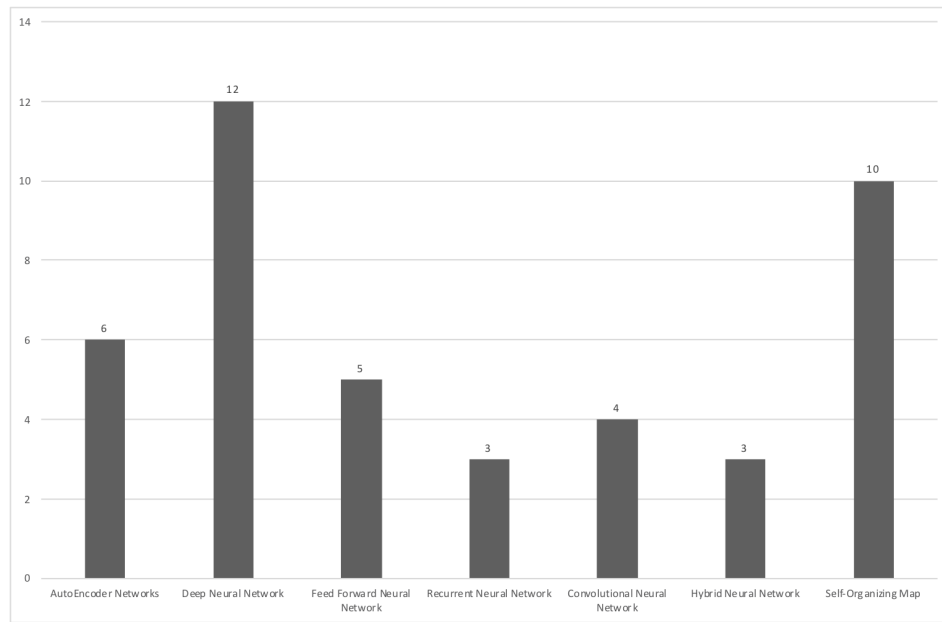


Figure 2.1: Distribution of studies based on neural network architecture.

Table 2.3: Distribution of studies based on neural network architecture.

Architecture	Studies
Convolutional Neural Network	[96] [100] [36] [65]
Recurrent Neural Network	[10] [16] [93] [79]
Feed Forward Neural Network	[83] [3] [4] [70] [66] [31]
AutoEncoder Networks	[94] [48] [12] [6] [33] [99]
Hybrid Neural Network	[101] [37] [77]
Deep Neural Network	[55] [98] [62] [50] [104] [7] [8] [82] [60] [49] [5] [82]
Self-Organizing Map	[78] [46] [40] [22] [41] [14] [75] [25] [26] [103]

respectively. Additionally, researchers of 3 articles (7%) preferred to use their own dataset to train and test the algorithms.

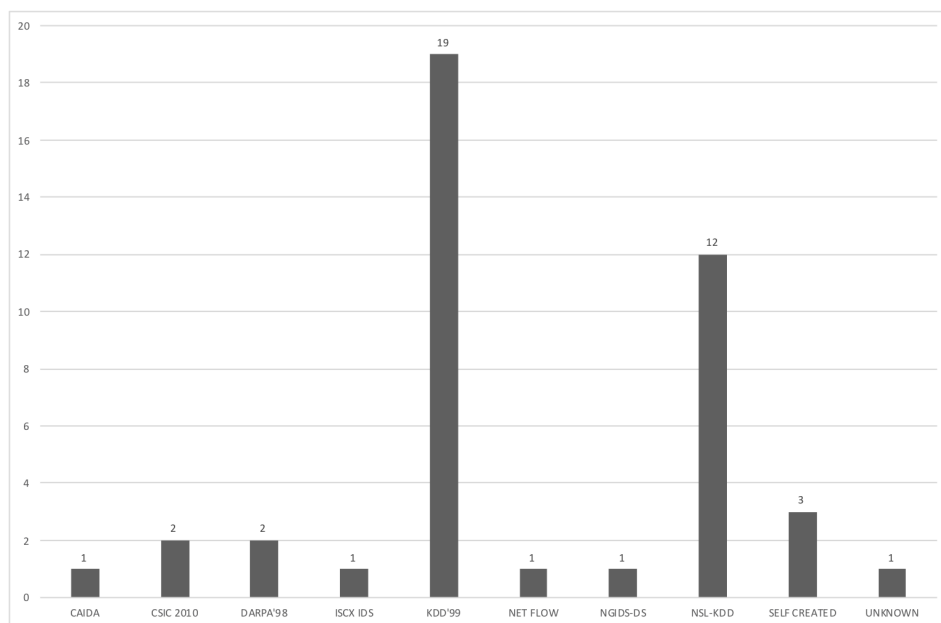


Figure 2.2: Distribution of studies based on dataset.

CHAPTER 3

METHODOLOGY

In this study, we focus on the comparison of the neural network architectures in the network intrusion detection system. We choose the most frequently used neural network architectures in the network intrusion detection system domain [73]. For this purpose, we adopted a systematic and structured methodology which consists of three main steps:

1. Pre-processing, cleaning, and transforming the relevant data,
2. Developing models using comparable analytical techniques,
3. Evaluating and assessing the validity and the utility of the models against each other and the goals of the study,

Overview presentation of our methodology has been exhibited in Figure 3.1.

First, Neural Network Generator (NNG) Preprocessor reads raw data with tracker interfaces. These interfaces specified for each source type which are big data tracker, Rest API tracker, and file tracker. KDD99 data stored in the big data environment for this study because of data size and accessibility. Each tracker reads data and transfers them to a matrix. NNG Preprocessor application was developed on C++ v17 programming language with R-Interpreter and Python-Interpreter.

In second phase, preprocessor starts to decode data. The computational graph which will be created by NNG needs numerical data and collected data contains protocol-specific values in it. As a result of that, decoding data is a crucial step for the rest of the experiment.

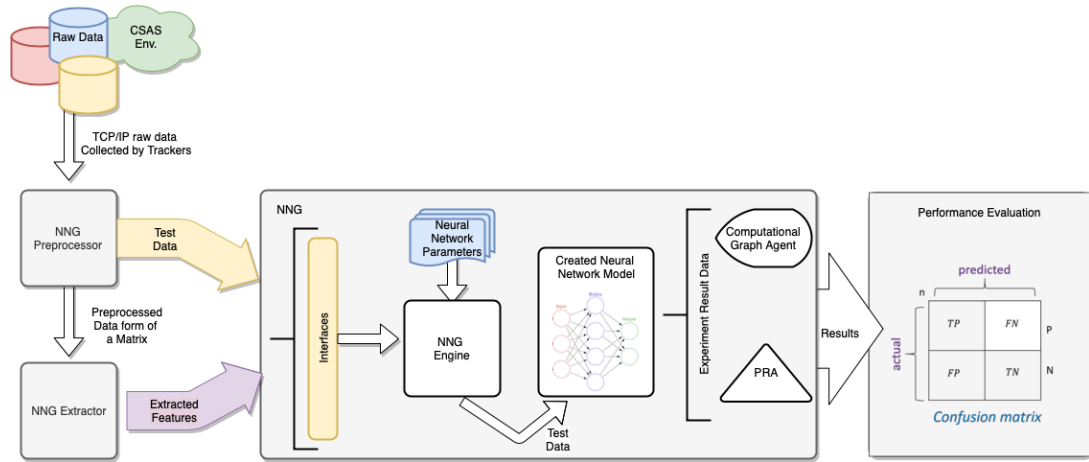


Figure 3.1: Overview of the methodology

NNG Extractor application extracts the basic features. The matrix, which is passed from NNG Preprocessor, is created from raw TCP/IP dump data. Extractor gathers each packages' duration information, protocol type, port information, connection flag, and sends/receives byte counts. Extractor also labels each feature for the type. If data feature is extracted from a time frame, extractor application marks this feature as a "Cont" otherwise data mark as "Disc". NNG Extractor application was developed on Python 3.6 programming language with Pandas v0.23.4 as a data analysis library, NumPy v1.15.4 as a scientific computing library, and SciKit v0.20.1 for regression calculation library.

After the extraction processes, data arrives to the NNG Engine. The NNG Engine creates neural network computational graphs according to given architecture and then evaluates the networks with the test data. Each generation of the network changes parameters stored in a matrix of activation functions, depth of the network, optimization algorithm, learning rate, learning momentum, and dropout rate. NNG Engine assures that all generated neural networks used the same preprocessed dataset as an input and the entire dataset was firstly split into two equal partitions. Then, the first half of the dataset was split into two parts (first 80% for training and the remaining 20% for validation). The remaining half part of the entire dataset (50%) is used for testing. NNG Engine was developed on Python v3.6 programming language with TensorFlow v1.12 as low-level computational graph API, and Keras v2.2.4 as high-level neural networks API.

Finally, NNG collects the performance and statistical data for the tested network. Each statistical data is stored in a comma separated value file for the future analysis with the name of the created network. Performance data of the network is also stored as a binary. This data contains information about each epoch of the network and the metric values.

The rest of the chapter is organized as follow: In Section 3.1, construction of the network intrusion detection system is described. In Section 3.2 and 3.3, the general properties of the KDD99 datasets, preprocessing stage and feature extraction from dataset are explained. In Section 3.4 the generation of the neural network models according to properties and given architectures are defined, and Section 3.5 gives information about the used software and libraries in the study, and finally, evaluation of the result and experiment platform are discussed in Section 3.6 and 3.7.

3.1 Network Intrusion Detection Model

To perform the comparison of the neural network in the network intrusion detection domain, we have adopted a top-down approach for constructing the neural networks for the experiment. Our method consists of five main phases; determination of the neural network architecture, determination of the parameters which are needed for the neural network architecture, generation of all possible combinations of the architectures with these parameters, building each architecture and collecting the test results, and evaluating these results.

Phase 1: Determination of the Neural Network Architectures

In this phase, we systematically reviewed the literature of the NIDS domain, and we defined the frequently used neural network architectures. At the end of the review, we determined the list of neural network architectures which are shown in Table 3.1.

Phase 2: Determination of the parameters of the Neural Networks

In this phase, we determine the components of the neural networks according to the architectures which affect the neural network performance. These components are as follows:

Table 3.1: List of Neural Network architectures in the NIDS domain.

#	Neural Network Architecture	Abbreviation
1	Convolutional Neural Network	CNN
2	Recurrent Neural Network	RNN
3	Feed Forward Neural Network	FFNN
4	AutoEncoder Networks	AEN
5	Self-Organizing Map	SOM
6	Evolutionary Neural Network	EEN

- **Optimizers** are specific algorithms which determine the distance between neural networks output and the labeled value. As an optimizer algorithm, we decide to choose the most suitable algorithms which are Stochastic Gradient Descent (SGD) and Adam optimizer in the domain. Adam and SGD optimizers are the most popular optimization algorithms and are common ways to optimize neural networks [93].
- **Activation functions** calculate the output value of a neuron. On the activation function side, we choose the most popular [84] nine functions for the NIDS domain. These functions are; elu, relu, selu, linear, sigmoid, softsign, softmax, softplus, tanh.
- **Learning rate** of the optimizer algorithms determines the size of the steps that are needed to reach the local minimum. In this research, learning rates of the networks are determined as commonly accepted range of 0.01 to 0.001[19][85].
- **Momentum** defines the oscillation of the learning rate in each epoch. We decay the network with the rate of 0.0 and 1×10^{-6} [19].
- **Dropout rate** is a regulation technique that prevents the overfitting problem in the neural network. Neurons are removed with the rate of 0.0, 0.25 and 0.50[86].
- **Depth of the network** is defined by the number of hidden layers of the network. We determined the depth of the network as 1, 2, 5, 10, 25 hidden layers depth. We limited the neural network depth with 25 layer because of the limitation of the hardware.

Phase 3: Generation of the all possible combinations of the architectures

Next, we generate all possible combinations of the neural networks according to network architecture and its parameters. According to phase 1 and phase 2, all possible combinations of the neural networks count is $6 * 2 * 9 * 2 * 2 * 3 * 5 = 6480$ combinations.

Phase 4: Building each architecture and collecting the test results In this phase, we developed an application named Neural Network Generator (NNG) which pick a neural network combination from the lookup table and create a computational graph for this architecture. After the creation of the architecture, NNG test the network with the test data and collect the performance counter of the architecture.

Phase 5: Evaluating these results

Finally, the collected performance counters are evaluated according to evaluation measurements in the Section 3.6.

3.2 Dataset

Network intrusion detection datasets were created with data packets from networks, command sequences from user input, or low-level system information, such as system call sequences, log files, and CPU/memory usage. We list some commonly used benchmarks in Table 3.2. All of these datasets have been used in either misuse detection or anomaly detection.

Table 3.2: Common NIDS Benchmark Datasets

Data source	Dataset name	Abbreviation
Network traffic	DARPA 1998 TCPDump Files	DARPA98
	DARPA 1999 TCPDump Files	DARPA99
	KDD99 Dataset	KDD99
	10% KDD99 Dataset	KDD99-10
	Internet Exploration Shootout Dataset	IES
User behavior	UNIX User Dataset	UNIXDS
	DARPA 1998 BSM Files	BSM98
System callsequences	DARPA 1999 BSM Files	BSM99
	University of New Mexico Dataset	UNM

DARPA'98 dataset has a number of issues associated with its design and execution

that remain unsettled. Some methodologies used in the evaluation are questionable and may have biased its results. One problem is that the evaluators have published relatively little concerning some of the more critical aspects of their work, such as validation of their test data [68]. Because of that, we focus on one benchmark: the KDD99 dataset.

3.2.1 KDD99 Dataset

KDD99 is a widely used and publicly available dataset for the network intrusion detection systems. The KDD99 dataset is prepared by Stolfo et al. [88]. This dataset drove from DARPA'98 NIDS evaluation program [92]. DARPA'98 is compressed binary data that is collected from 7 weeks of network traffic as a raw TCP dump. This data contains about 5 million connection records with each about 100 bytes. This 4 gigabytes of compressed data is split into two parts. First two weeks of data is prepared for the testing purpose and contains approximately 4,900,000 single connection vectors with 41 features. Each vector is labeled as either normal or a specific attack type. These attacks types can be categorized into 4 groups as following :

1. Denial of Service Attack (DoS): is an attack in which attackers try to deny legitimate users access to a service.
2. User to Root Attack (U2R): is an exploit which allows an attacker to access a normal user's or a superuser's account on the system.
3. Remote to Local Attack (R2L): is a vulnerability that an attacker can send packets to the target machine over a network in spite of attacker's privileges.
4. Probing Attack: is an information gathering attempt from network and devices on that network.
5. Normal : is an information that shows the normal network traffic flow.

Test part of the research contains 13 attack types. The names and detailed information about the attacks are listed in Table 3.3.

Table 3.3: Attacks listed in the KDD99 dataset

Category	Attack Type	10% KDD99	
		Train	Test
DOS	back	2203	1098
	land	21	9
	neptune	107201	58001
	pod	264	87
	smurf	280790	164091
	teardrop	979	12
Probe	ipsweep	1247	306
	nmap	231	84
	portsweep	1040	354
	satan	1589	1633
R2L	ftp_write	8	3
	guest_password	53	4367
	imap	12	1
	multihop	7	18
	phf	4	2
	spy	2	0
	warezclient	1020	0
	warezmaster	20	1602
U2R	loadmodule	9	2
	buffer_overflow	30	22
	rootkit	10	13
	perl	3	2
Normal		97278	60593
Total		494021	292300

3.3 Data Preprocessing

Data preprocessing is required in every knowledge discovery system, including NIDS, which tries to classify network traffic as normal or attack. Standard preprocessing steps for this study include data cleaning, integration, feature extraction, data normalization, and information decoding.

In this study, the data preprocessing stage is executed with an application named NNG Preprocessor, developed in Python, to feed the feature extraction unit of the test bench. Preprocessor works on raw data. This raw data retrieved from a dataset KDD99 or are collected a simulation environment. Either way, read data is captured

package data form of TCP/IP dump. NNG Preprocessor reads raw data with tracker interfaces. These interfaces are specified for each source type which are big data tracker, Rest API tracker, and file tracker. KDD99 data is stored in the cloud environment for this study because of data size and accessibility. Each tracker reads data and converts the data to a matrix. After the matrix conversion process, preprocessor starts to decode data. The computational graph which will be created by NNG needs numerical data and collected data contains protocol-specific values in it. As a result, decoding data is a crucial step for the rest of the experiment.

Finally, NNG Preprocessor passes the prepared data to an application named NNG Extractor for the feature extraction.

3.3.1 Feature Extraction

Collected dataset's features can be categorized into three following groups:

1. Basic features contain all the attributes that can be extracted from TCP/IP connections.
2. Traffic features contain computed attributes in the network traffic with respect to a window interval of 2 seconds. However, some of the slow probing attacks like low rate port scanning use much larger time interval. For this reason, host, and service attributes re-calculate according to connection windows of 100 connection rather than time window of 2 seconds.
3. Content features R2L and U2R attacks do not contain any sequential pattern to follow. This kind of attack usually uses a single connection to gather information from the network. Because of this, features extract from the suspicious behaviors in the data portions.

Kayacik et al.[47] investigate the relevance of each feature in KDD99 intrusion detection datasets. They create a feature set that contains 41 distinct feature about the data. Six of these features are extracted from TCP/IP connection data and the rest of them are extracted from traffic and content information. NNG Extractor application

determines and starts to extract 41 predefined features from the data matrix based on the entropy of the feature.

Entropy comes from information theory. Shows the amount of information obtained. In other words, the higher the entropy, the more information content it has. Information indicates the importance of property or quality, and determines which attribute is most important to distinguish between classes to be knowledgeable. This piece of information is also calculated by the training data. Information can help in choosing the best distinction; If the value is high, then this section is good, otherwise the section is not good enough.

NNG Extractor application firstly extracts the basic features. The matrix is which passed from NNG Preprocessor is created from raw TCP/IP dump data. Extractor gathers each packages' duration information, protocol type, port information, connection flag, and send/receive byte counts. Extractor also labels each feature for the type. If the data feature is extracted from a time frame, extractor application mark this feature as a "Cont" otherwise data is marked as "Disc". Full features are listed in the Table 3.4.

3.4 Neural Network Generator

The experiments were performed with an application named Neural Network Generator (NNG), developed in Python with TensorFlow and Keras libraries. These libraries are explained in Section 3.5. The application creates neural network computational graphs according to given architecture and then evaluates the networks with the test data. Each generation of the network changes parameters stored in a matrix of activation functions, depth of the network, optimization algorithm, learning rate, learning momentum, and dropout rate.

Firstly, NNG gets the definition of the neural network architecture from the lookup map. This lookup map holds the description of the neural network architecture with the appropriate parameters which is needed to create the computational graph. Neural network architectures which are used by the NNG are listed in Table 3.5.

Table 3.4: Extracted features.

Feature	Description	Type
duration	Duration of the connection.	Cont.
protocol type	Connection protocol (e.g. tcp, udp)	Disc.
service	Destination service (e.g. telnet, ftp)	Disc.
flag	Status flag of the connection	Disc.
source bytes	Bytes sent from source to destination	Cont.
destination bytes	Bytes sent from destination to source	Cont.
land	1 if connection is from/to the same host/port; 0 otherwise	Disc.
wrong fragment	number of wrong fragments	Cont.
urgent	number of urgent packets	Cont.
hot	number of "hot" indicators	Cont.
failed logins	number of failed logins	Cont.
logged in	1 if successfully logged in; 0 otherwise	Disc.
# compromised	number of "compromised" conditions	Cont.
root shell	1 if root shell is obtained; 0 otherwise	Cont.
su attempted	1 if "su root" command attempted; 0 otherwise	Cont.
# root	number of "root" accesses	Cont.
# file creations	number of file creation operations	Cont.
# shells	number of shell prompts	Cont.
# access files	number of operations on access control files	Cont.
# outbound cmds	number of outbound commands in an ftp session	Cont.
is hot login	1 if the login belongs to the "hot" list; 0 otherwise	Disc.
is guest login	1 if the login is a "guest" login; 0 otherwise	Disc.
Count	number of connections to the same host as the current connection in the past two seconds	Cont.
srv count	number of connections to the same service as the current connection in the past two seconds	Cont.
error rate	% of connections that have "SYN" errors	Cont.
srv error rate	% of connections that have "SYN" errors	Cont.
error rate	% of connections that have "REJ" errors	Cont.
srv error rate	% of connections that have "REJ" errors	Cont.
same srv rate	%of connections to the same service	Cont.
diff srv rate	% of connections to different services	Cont.
srv diff host rate	% of connections to different hosts	Cont.
dst host count	count of connections having the same destination host	Cont.
dst host srv count	count of connections having the same destination host and using the same service	Cont.
dst host same srv rate	% of connections having the same destination host and using the same service	Cont.
dst host diff srv rate	% of different services on the current host	Cont.
dst host same src port rate	% of connections to the current host having the same src port	Cont.
dst host srv diff host rate	% of connections to the same service coming from different hosts	Cont.
dst host error rate	% of connections to the current host that have an S0 error	Cont.
dst host srv error rate	% of connections to the current host and specified service that have an S0 error	Cont.
dst host error rate	% of connections to the current host that have an RST error	Cont.
dst host srv error rate	% of connections to the current host and specified service that have an RST error	Cont.

As an optimizer parameter for the computational graph, we chose Adam optimizer and Stochastic Gradient Descent (SGD) optimizer. Adam and SGD optimizers are the most popular optimization algorithms and are a common way to optimize neural networks [93]. The Stochastic Gradient Descent algorithm minimizes the cost function by updating the parameters in the reverse direction. On the other hand, Adam optimizer computed derivative of the mini-batch of parameters and took the weighted average of this calculation. Adam optimizer drove from RMSprop optimizer algorithm. Adam optimizer has many variations of itself like Nadam, Adamax.

The learning rate of the optimizer algorithms determines the size of the steps that we

Table 3.5: Neural Network Architectures

Neural Network Architecture	Short Description
Convolutional Neural Network (CNN)	A Convolutional Neural Network is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network.
Recurrent Neural Network (RNN)	A recurrent neural network is a class of artificial neural network where connections between nodes form a directed graph along a sequence.
Feed Forward Neural Network (FFNN)	A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.
AutoEncoder Networks (AEN)	An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner.
Self-Organizing Map (SOM)	A self-organizing map or self-organizing feature map is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples, called a map, and is, therefore, a method to perform dimensionality reduction.
Evolutionary Neural Network (ENN)	Neuroevolution, or neuro-evolution, is a form of artificial intelligence that uses evolutionary algorithms to generate artificial neural networks(ANN), parameters, topology, and rules.

needed to reach the local minimum. Besides the fix learning rate¹, computing the adaptive learning rate for each parameter is also a regularly used method in the neural networks. In this method, a learning momentum variable is used. This variable stores an exponential decay average of the past squared gradient. This way, oscillation of the learning rate in each step will decrease with respect to the momentum variable.

Another parameter of the network are the activation functions. An activation function calculates the output value of a neuron. Details of the activation functions that are chosen for the network creation explained in Section 2.2.1.

The depth of the network parameter determines the count of the hidden layers. NNG creates neural networks. This parameter has a significant effect on the training time

¹ Fix step size is hyperparameter which determines to what extent newly acquired information overrides old information.

and detection time of the network.

The last used parameter in the network is the dropout rate. Dropout is a regulation technique that prevents the overfitting problem in the neural network [86].

Moreover, NNG determines the CPU or GPU specs of the test platform and decides the consecutive thread count for the platform. This determined consecutive thread count will be an important part of the computational graph. If the number of threads is higher than the hardware can lift, the current platform will start to experience problems with performance. Correspondingly, NNG generates computational graph according to selected neural network parameters and add this created neural network to the appropriate thread. This created computational graph organizes the neural network in terms of a forward propagation step. This step computes the output of the neural network. Process followed by a backward pass or back propagation step which is used to compute the gradients or compute the derivatives.

At the same time, NNG arranges the input interface for the dataset. For each thread, NNG uses this input interface to construct a dataset for the prepared the computational graph. NNG can get input from either the recorded dataset or simulation environment. Each input interface opens a channel for the relevant thread. Then, preprocessed data redirected to neural networks' input channel. Finally, NNG collects the performance data and statistical data for the tested network. Each statistical data are stored in a comma separated value file for future analysis with the name of the created network. Performance data of the network is stored in binary. This data contains information about each epoch of the network and the metric values. Performance data contains variables such as time for each epoch, training accuracy, moving mean for each step, Poisson Distribution result and cross entropy changes according to time.

3.4.1 Network Attack Simulator

The most challenging aspect of simulation of network-based intrusion detection research is proving the reliability and dependability of simulated datasets in comparison to real-life datasets. On the other hand, well designed simulation environment offers repeatability, programmability and extensibility of the validation instrument [15].

A Virtual Lab named Cyber Security Attack Simulator (CSAS) have been implemented in order to create an isolated platform to simulate, test and analyze different types of security threats. The infrastructure was built by means of a VMware virtualization software (or, for repeatability of the study, any virtualization platform can be used for this stage) on one physical machine. In order to connect the virtual machine to the network, we mapped the external internet connection of our host machine to the internal VM network. The CSAS allows us to configure different network topologies for defining different attack scenarios. The virtual testbed is an isolated environment to mainly fabricate and collect simulated attack data. As, network technologies are growing rapidly, we primarily employed open platforms to include different efforts and different packages whenever there is a need [76]. Moreover, using open source tools and applications facilitate the repeatability of the study. We used open source systems and applications, e.g., GNS3, Ubuntu, Pfsense firewall in this study.

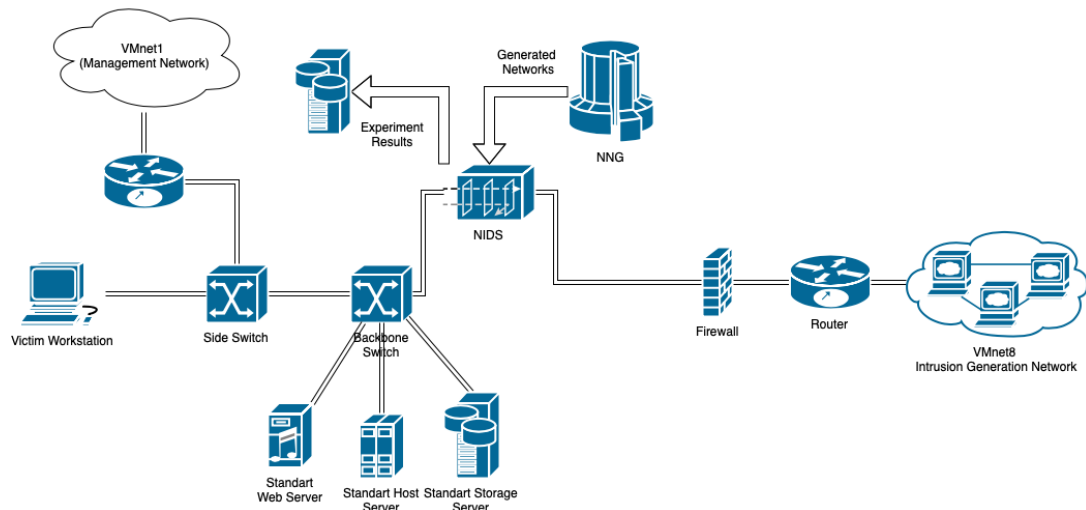


Figure 3.2: Simulator network topology

We initially defined the network topology as shown in Figure 3.2 using GNS3. Attacker and target machines are Ubuntu Docker appliance for GNS3. Pfsense is an appliance of the GNS3. VMnet8 is our exit point to the internet. We disabled all incoming and outgoing traffic from the VMnet8 using firewall rules during the experimental phase.

In this experiment, CSAS uses the remaining half of the segmented dataset. First, this data is converted into a time series to be used by the created ANN. In this way, they

can be tested in a real environment. It moves in the form of traffic time series in real-life network structures. They are also available in non-clean data in this traffic. The topologies tested with static data are transferred over CSAS. They continue testing in a structure close to the actual environment. This data is generated from data that the ANNs have never seen before. A system contained in the VNet8 network in the test environment has been established to provide data to the environment. This data can be accessed from a firewall to access the NIDS location. The data generated by the data provider in the VNet8 environment is such that it will provide attack data to the servers and clients behind the backbone.

As a result of these tests, the data is collected and added to the results of the tests carried out with static data.

3.5 Software Used

In this study, Neural Network Generator (NNG) application was developed on Python v3.6 programming language with TensorFlow v1.12 as low-level computational graph API, and Keras v2.2.4 as high-level neural networks API. NNG also uses the TensorBoard for the collect the statistical data about the computational graph. NNG Extractor application was developed on Python 3.6 programming language with Pandas v0.23.4 as a data analysis library, NumPy v1.15.4 as a scientific computing library, and SciKit v0.20.1 for regression calculation library. NNG Preprocessor application was developed on C++ v17 programming language with R-Interpreter and Python-Interpreter. Submodules, calculate and normalize the data, of the NNG Preprocessor, were developed on R programming language. Finally, R programming language also was used in Platform Result Analyzer (PRA) to calculate the evaluation results.

In the simulation part of the study, Virtual Lab named Cyber Security Attack Simulator (CSAS) was implemented on VMWare v14 virtualization environment. On this environment, Docker was used as a container manager, GNS3 was used as network simulation manager and Cisco 3500 switch image, and PfSense firewall images were used as active network components on GNS3 simulator. As an attacker and victim platform, Ubuntu Xenial OS was chosen.

The experiments were performed on the Intel x64 based i7 3.5Ghz CPU, 32GB of memory, two NVIDIA 1080x graphic card connected with SLI and 5 TB of disk space.

3.6 Evaluation Measures

Network Intrusion Detection Systems (NIDS) are designed for detecting anomalies and attacks in the network. Thus, the effectiveness of a NIDS can be by its capacity to make accurate predictions[74]. Therefore, there are four possible outcomes for a given event. These outcomes are shown in Table 3.6, also known as the confusion matrix. True negatives, as well as true positives, correspond to correct operation of the NIDS; that is, the classification of the NIDS was correct, and events marked as "normal" or "attack". False positives indicate to "normal" events being predicted as "attack"; false negatives are "attack" events incorrectly predicted as "normal" events.

Table 3.6: Confusion Matrix

		Predicted class	
		Negative class (Normal)	Positive class (Attack)
Actual class	Negative class (Normal)	True negative (TN)	False positive (FP)
	Positive class (Attack)	False negative (FN)	True positive (TP)

3.6.1 True Negative Rate

True negative rate (TNR) 3.1, also known as Specificity, is the ratio of the number of legitimate records detected as normal instances divided by the total number of normal (legitimate) instances included in the test set.

$$\frac{TN}{TN + FP} \tag{3.1}$$

3.6.2 True Positive Rate

True positive rate (TPR) 3.2, also known as Detection Rate (DR) or Sensitivity, is defined as the number of intrusion records detected as attacks by the NIDS divided

by the total number of intrusion records present in the test set.

$$\frac{TP}{TP + FN} \quad (3.2)$$

3.6.3 False Positive Rate

In the network intrusion detection system, a positive result of an event considered as an "attack". False positive rate (FPR) 3.3, also known as false alarm rate (FAR), is the ratio of the number of legitimate instances detected as attack instances divided by total normal (legitimate) instances included in the test set.

$$\frac{FP}{TN + FP} = 1 - Specificity \quad (3.3)$$

If the false positive rate is high in a NIDS, it can cause the administrator to intentionally ignore the system warnings, which makes the system enter into an uncertain status. Thus, a NIDS should have a high DR and a low FAR [102].

3.6.4 False Negative Rate

False negative rate (FNR) 3.4 is the ratio of the number of attack instances detected as normal instances divided by the total attack instances included in the test set. This term is used to describe a network intrusion device's inability to detect the true attack. The FNR value can be calculated by one minus TPR.

$$\frac{FN}{TP + FN} = 1 - Sensitivity \quad (3.4)$$

3.6.5 Precision, Recall, and F-Measure

The Precision, Recall, and F-Measure metrics ignore the normal data that has been correctly classified by the NIDS (TN), and focus on both the intrusion data and FAR generated by NIDS.

Accuracy (3.5) can be defined as the proportion of the total number of the correct predictions to the actual test set size.

$$\frac{TN + TP}{TN + TP + FN + FP} \quad (3.5)$$

Precision (3.6), which is another information retrieval term, is often is paired with “Recall”. It indicates the percentage of intrusions that have occurred. It is calculated by the number of correctly classified positive (intrusion) examples divided by the number of examples labeled by the system as positive. A NIDS aims to obtain a high Precision, meaning that the number of false alarms is minimized.

$$\frac{TP}{TP + FP} \quad (3.6)$$

Recall measures the missing part from the Precision; namely, the percentage from the real intrusions covered by the classifier. Consequently, it is desired for a classifier to have high recall value. Recall does not take into consideration the number of False Alarms. Thus, a classifier can have at the same time both good recall and high false alarm rate.

F-score or F-measure is considered as the harmonic mean of recall and precision. The higher value of F-score indicates that the NIDS is performing better on recall (true positive rate) and precision. F-Measure is preferred when only one accuracy metric is desired as an evaluation criterion. Note that when Precision and Recall reaches 100%, the F-Measure is maximum, meaning that the classifier has 0% false alarms and detects 100% of the attacks. Thus, the F-Measure of a classifier is desired to be as high as possible.

3.7 Experiment

NNG assures that all generated neural networks used the same preprocessed dataset as input and the dataset were split into two. The first half of the dataset is split into 80% for training and 20% for validation and the remaining part is used for testing.

In the training phase, every network is trained with constant epoch value. After the training phase is complete, each network is tested with an unknown data which is the remaining half of the main dataset and simulated attack data.

The time performance of an intrusion-detection system corresponds to the total time that the NIDS needs to detect an intrusion. This time includes the processing time and the propagation time. The processing time depends on the processing speed of the NIDS, which is the rate at which the NIDS processes audit events. Time performance is a critical metric for the NIDS. For this reason, each generated the network's training time data and detection time data stored.

Eventually, stored results and metrics are analyzed with the help of an application named Platform Result Analyzer (PRA) which is developed on R programming language.

CHAPTER 4

RESULTS

We performed 6480 comparative experiments as explained in Section 3.1 by using the KDD99 dataset with extracted 41 features. After the evaluation of the experiments, we selected the top ten successful networks according to the false positive rate and accuracy metrics. We evaluate these results of the experiments in three main section. First, we focus on the comparison of neural network architectures' performance on network intrusion detection domain. Second, we elaborate on how neural network parameters affect the evaluation counters like false positive rate and accuracy of the network in the data and, lastly, we detail the result of the top selected networks' behavior in the Cyber Security Attack Simulator (CSAS).

4.1 Results of the Architectural Comparison

In our experiments, all Convolutional Neural Networks (CNN) architectures have resulted in accuracy below 10%, as shown in Figure 4.4 . This is because of the structure of the CNN. CNN architectures convolute the input data. However, in the NIDS domain, convoluting reduces the number of features in the input data. Thus, every created CNN architecture resulted with a low accuracy rate.

Regarding the false positive rate and accuracy, lowest detection time was reached by CNN with a time of 0.23s, shown as in Figure 4.2. However, CNN architecture lost significant performance because of its architecture. Convoluting the input data caused the loss of features. Despite the detection time, lowest accuracy and highest false positive rates belong to the CNN architecture in our experiment. Also because of

the architecture, the Convolutional Neural Network has the lowest training time with 248s. Overall node count in a CNN is lower than the other network architectures. This effect suggests that the time cost of backpropagating the network for each epoch is less than the other architectures for the NIDS domain. For the rest of the architectures, Autoencoder, Feed Forward Neural Network, and Recurrent Neural Network, the average training times were in the range between 343s to 934s as shown in Figure 4.1.

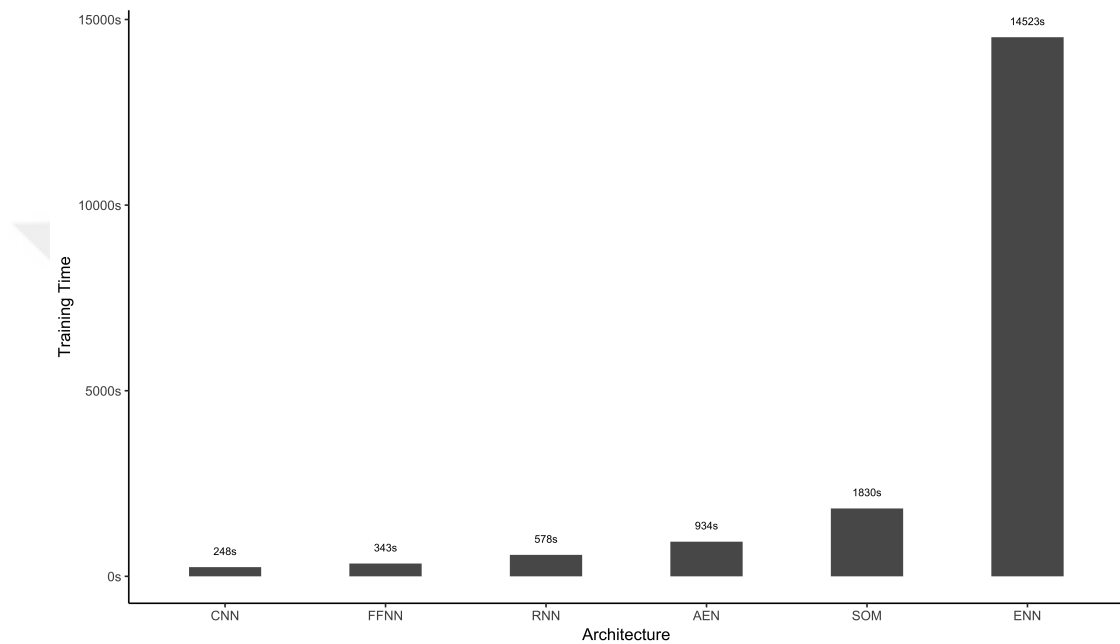


Figure 4.1: Architecture average training time.

Furthermore, in Figure 4.2, the training time of the Evolutionary Neural Network architecture was over 4 hours (14523s) on average. Neuroevolution technique creates a batch of neural networks and tests them with a fitness function with the given data. Each batch contains more than one neural network. In this case, in our experiment neuro-evaluator creates 20 neural networks for a batch. Thus, testing and backpropagating each networks' time cost becomes higher. Meanwhile, ENN architecture created complex networks in the NIDS domain with the selected feature. Because of that, ENN has the highest detection time. Figure 4.2 shows the average detection time of the neural network architectures.

On the other hand, in Figure 4.3 and Figure 4.4, the average accuracy of the Auto Encoder Networks reached 94.71%, and the average false alarm rate was 0.35%. Auto

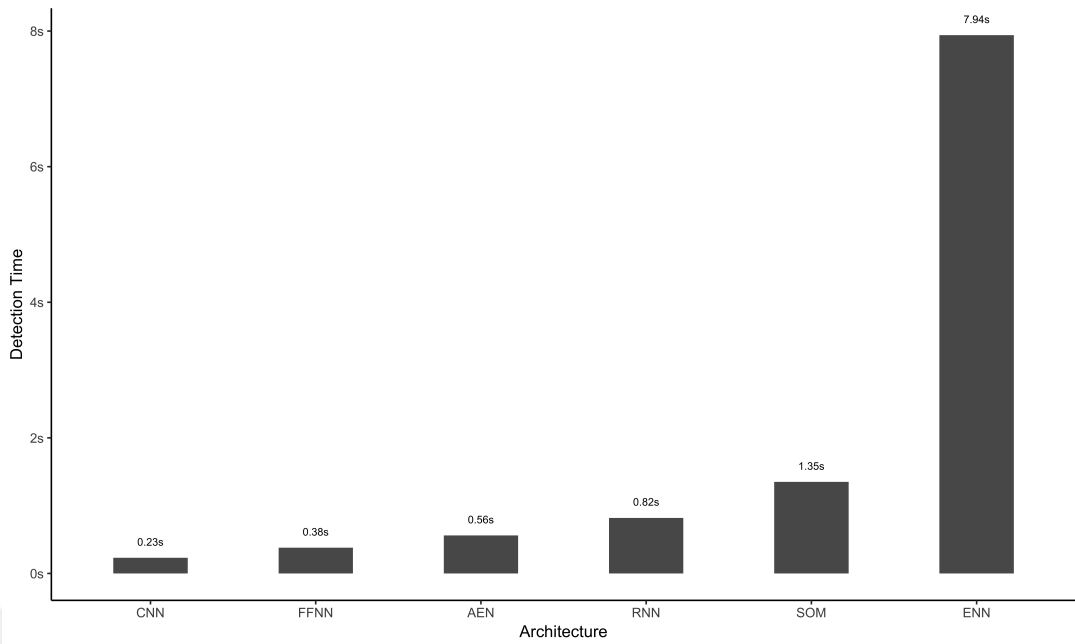


Figure 4.2: Architecture average detection time.

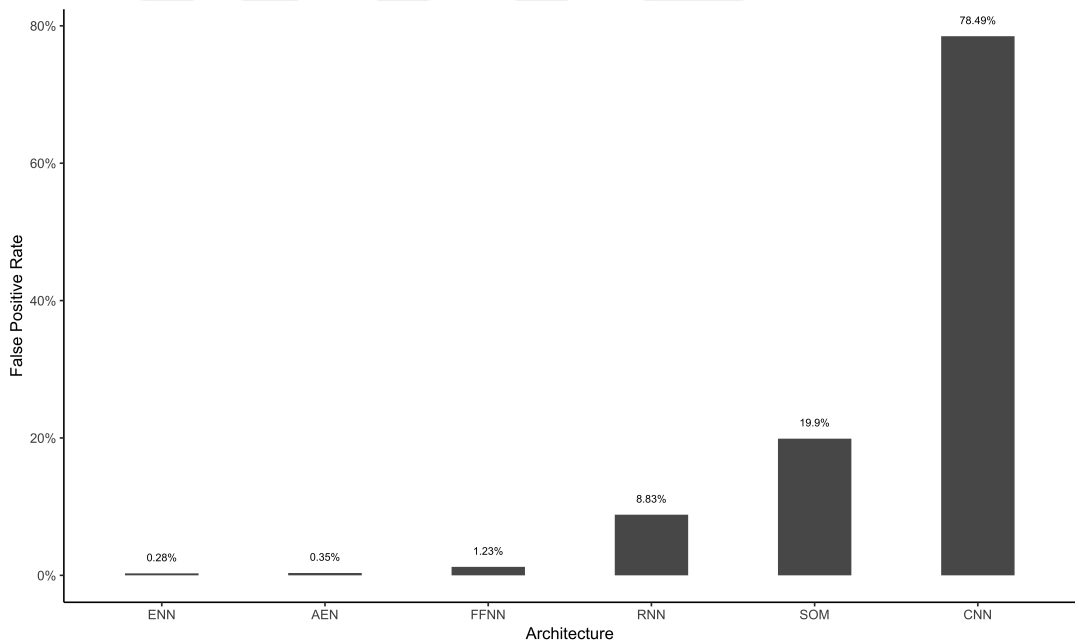


Figure 4.3: Architecture average False Positive Rate.

Encoder networks can compress the high dimensional input data into the low dimensional form, and this kind of network architectures can decode low dimensional data to original data. According to an architectural structure, basic working principle of an autoencoder network is encoding the input data into the hidden layers and decod-

ing the data from the hidden layers to the output layer. For this reason, autoencoder networks' training times and detection times are less than the other complicated, deep neural networks.

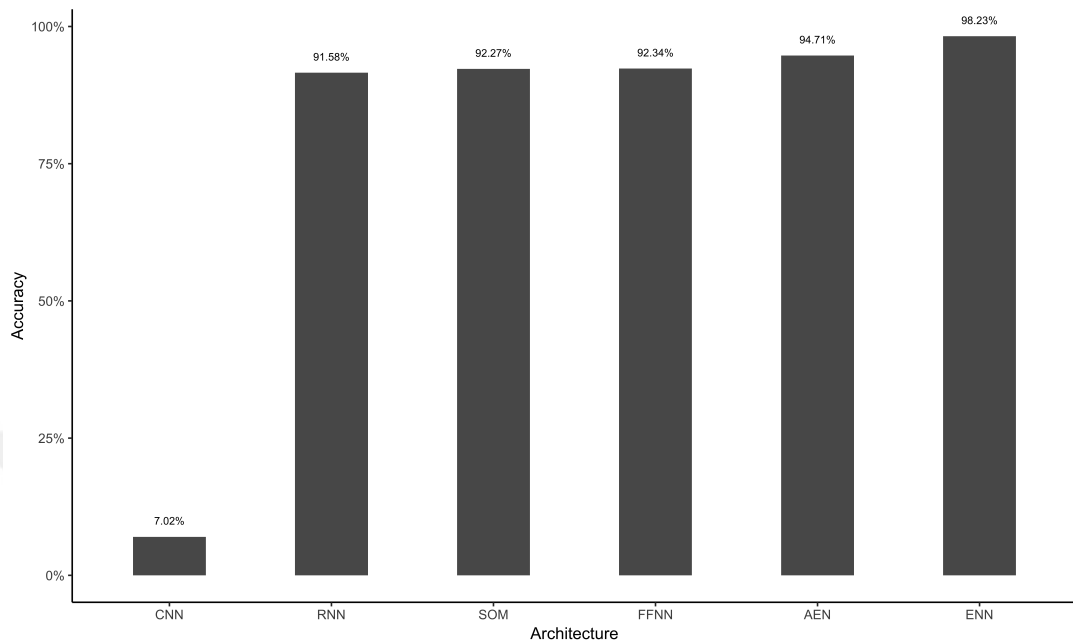


Figure 4.4: Architecture average Accuracy.

Lastly, in Figure 4.3 and Figure 4.4, for Feed Forward Neural Network architectures, the average accuracy was 92.34%, and the false positive rate was 1.23%. Moreover, Recurrent Neural Networks reached 91.58% for accuracy and average of 8.83% of the false positive rate. RNNs perform better than the other networks if the input is in stream form. However, this performance may vary according to the determination of the time frame in the preprocessing stage. On the other hand, Feed Forward Neural Networks gave optimal performance curve with the proper choice of activation function and optimization method.

4.2 Top 10 of the Neural Networks

For defined evaluation criteria and methodology, we sorted results according to lowest false positive rate and highest accuracy. In this section, we present the results of the chosen top 10 networks, listed in Table 4.1.

Table 4.1: Top 10 rated network.

Type	Architecture	Hidden Layer	Output Layer	Optimizer	Learning Decay	Dropout
Type 0	FFNN	softmax	softmax	Adam	decay	0.5
Type 1	AEN	sigmoid	sigmoid	Adam	fix	0
Type 2	AEN	sigmoid	sigmoid	Adam	fix	0.5
Type 3	AEN	sigmoid	sigmoid	Adam	decay	0
Type 4	FFNN	elu	softmax	Adam	fix	0.5
Type 5	RNN	relu	softmax	Adam	fix	0.5
Type 6	RNN	relu	softsign	Adam	decay	0.5
Type 7	FFNN	softplus	relu	Adam	fix	0
Type 8	RNN	relu	softsign	Adam	fix	0.5
Type 9	FFNN	relu	elu	Adam	decay	0.5

In Figure 4.5, all the top 10 networks accuracy distributed between 94.074% and 99.012%. The Type 7 network has the highest accuracy rate with 99.012% and the Type 3 lowest accuracy rate with 94.074%. Type 7 network used Feed Forward Neural Network architecture and Type 7 network was created with Softplus activation function in the hidden layers. 9 of the top 10 networks employed an activation function which is a member of the logistics in their hidden layers function family. Due to the nature of the sigmoid activation function, it converges close to its saturation values of 0 and 1.

Also, this network employed the Adam optimizer. Adam optimizer has the lowest training error and is computationally faster than the SGD. Adam optimizer also reached a steady state in fewer epochs than the SGD. Later epochs' oscillation created large updates in the SGD and needed a second order momentum for reducing the update rate.

Moreover, 7 of the top 10 networks employed a dropout method with the rate of 0.5. This statistic indicates that these networks have an issue with overfitting. Overfitting is an error when the network starts to memorize the result. The reason for the overfitting is that input data is not sufficient for the network or the created network has a complex structure. In our case, the problem which caused the overfitting is the complexity. Networks that use dropout values usually have a deeper hidden layer structure in this situation.

In the perspective of false positive rate, the Type 0 has the minimum rate with 0.085%. This network used the Adam optimizer with Softmax activation function in the hidden

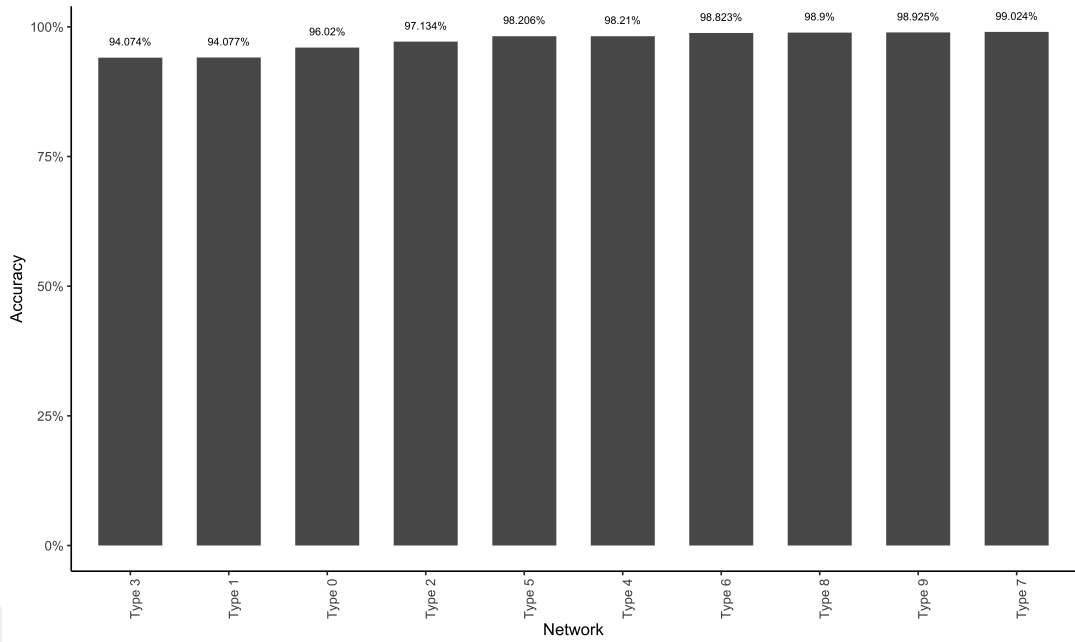


Figure 4.5: Top 10 Neural Network's accuracy.

layers. Moreover, this network used a drop-out value with 0.5. The Type 0 network is created with FFNN architecture. The Type 9 network has the maximum value with 0.819%. The distribution of the results can be seen in Figure 4.6

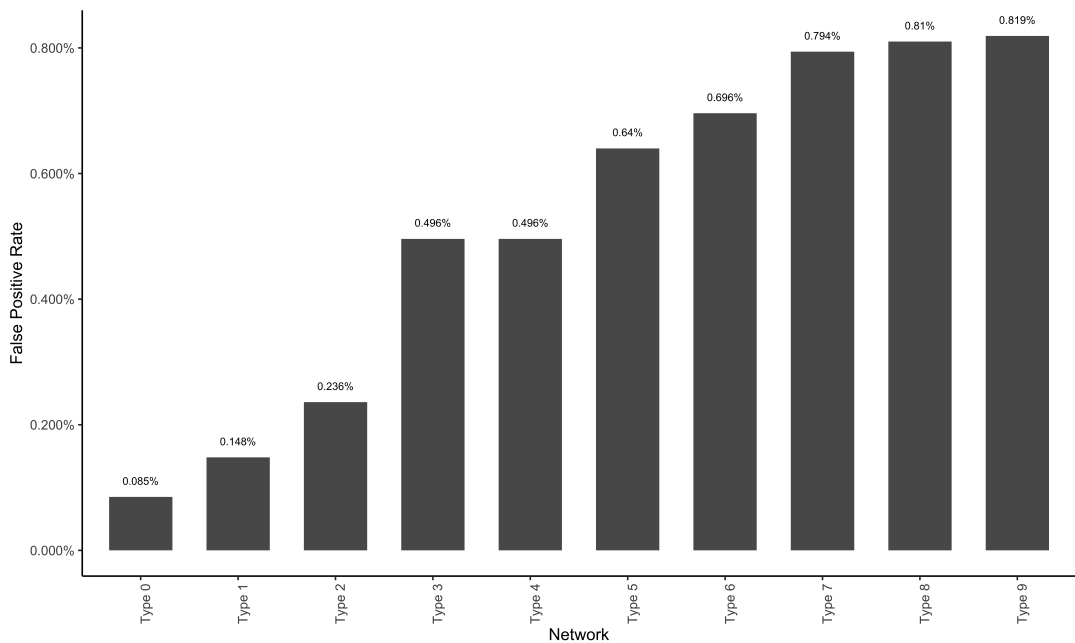


Figure 4.6: Top 10 Neural Network's false positive rate.

4.3 Effects of the Parameters

Figure 4.7 describes the performance comparison of the optimizer functions used in all ANNs tested in the experiment. In the pivoting of the optimizer algorithm during the benchmarking, we see that the Adam optimizer algorithm achieves a higher success than the SGD optimizer algorithm in all experiments. This trend continues in the topologies of ANN. As is illustrated by Figure 4.7, Adam optimizer has the lowest false positive rate against SGD optimizer. Furthermore, Adam optimizer has the highest accuracy and precision value overall. Adam optimizer targets high dimensional or large dataset space [51]. Also, because of the memory efficiency and ability to work with the higher dimensional data, the Adam optimizer provides the lower training time and detection time. However, SGD optimizers fitted FFNN had a better performance rate over Adam optimizer. Finally, deeper network with Adam optimizers resulted with higher accuracy and lowest false positive rate against SGD optimizer.

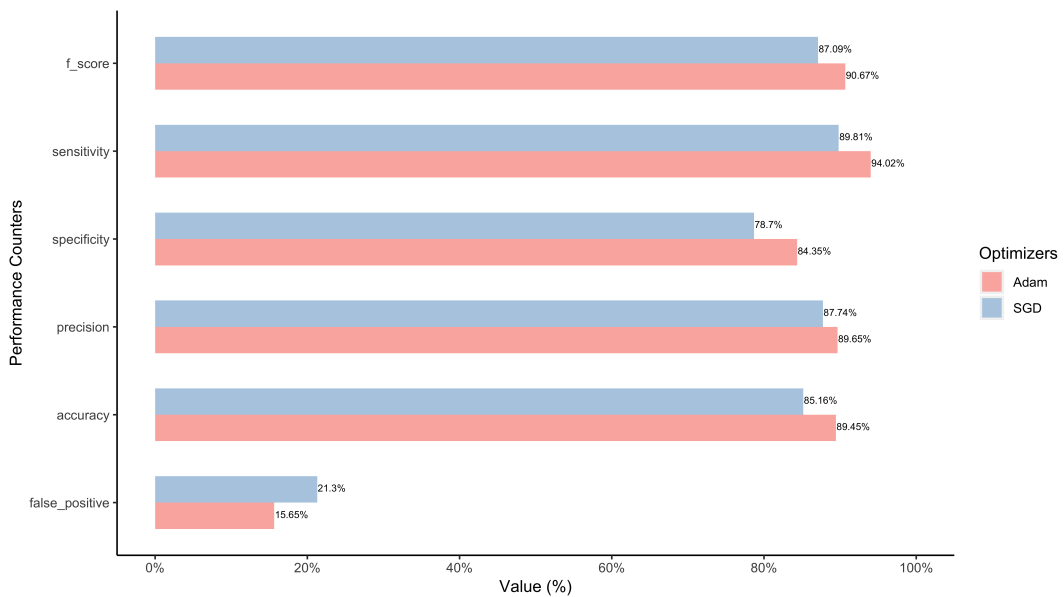


Figure 4.7: Comparison of optimizers.

As shown by Figure 4.8, Architectures which employed a decay momentum value for their learning rate reach slightly better performance rate against fix learning rate. This result is also the same for the accuracy and precision value. Network with Adam

optimizers, perform better with the momentum parameters because of the nature of the optimizer. On the contrary, momentum value had a significant effect on the training cost of the networks. Networks which employed a degree of momentum resulted with 13.43% lower training time and 23.46% lower resource consumption.

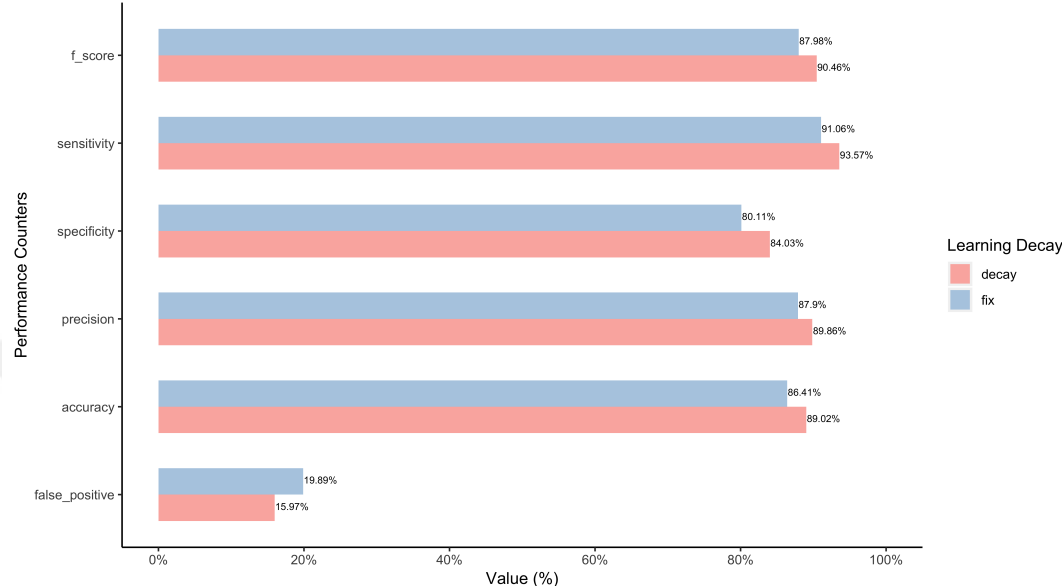


Figure 4.8: Compression of learning decay rates.

Dropout applied networks resulted with a high false positive rate against the non-applied network. Moreover, this result reflected on accuracy and precision value of the non-applied networks as shown in Figure 4.9. The effectiveness of the dropout was higher with deeper neural networks. However, this conclusion is heavily dependent on the input dataset size.

In the hidden layer, Softplus activation function has the lowest false positive rate on average with 10.8% as shown in Figure 4.10. However, in terms of the accuracy rate Softmax, Softplus and Sigmoid activation function share the leading place. These three activation functions have the same characteristics and all the three activation function belong to the logistic function family. Due to the nature of the logistic function, it converges close to its saturation values of 0 and 1. In addition, as a result of the experiments, the ReLu activation function showed a highest performance than the other functions in RNN topologies.

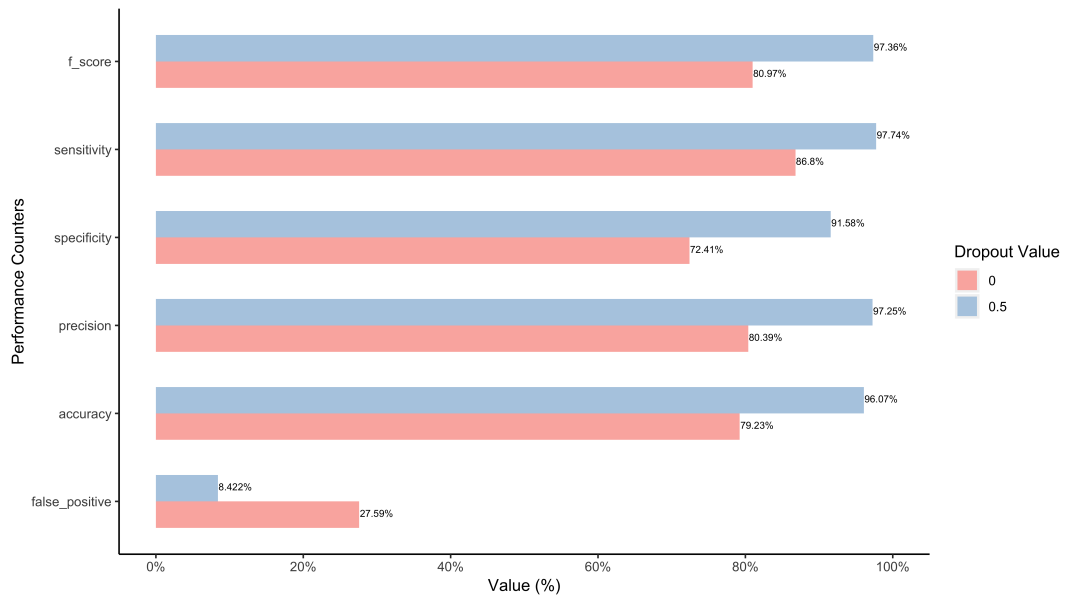


Figure 4.9: Compression of dropout rates.

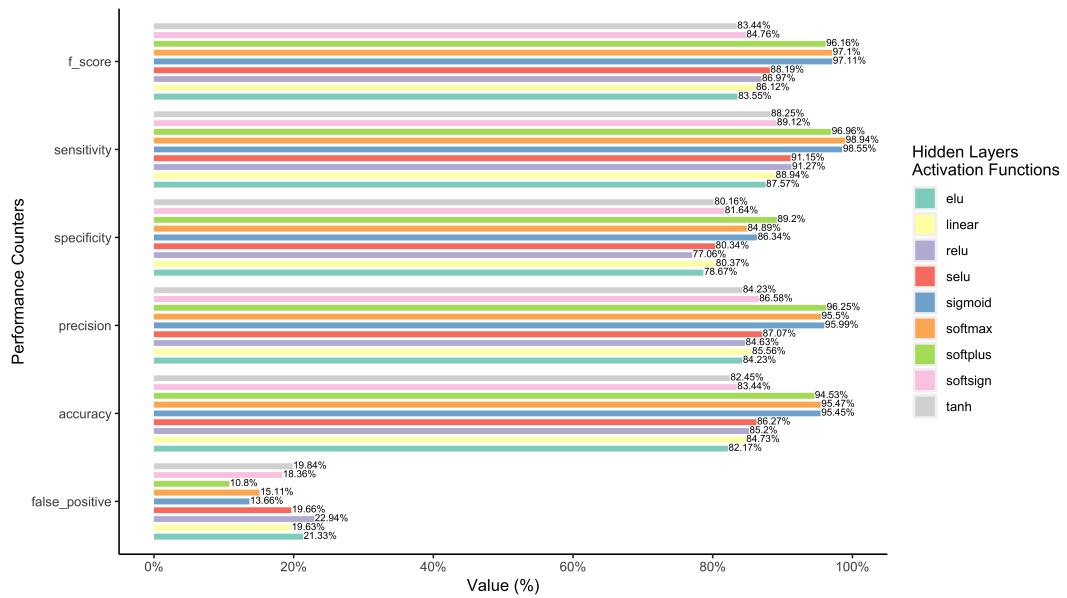


Figure 4.10: Hidden layer activation function performance counters.

CHAPTER 5

DISCUSSION AND CONCLUSION

Since the purpose of this research was to compare the efficiency of neural network architectures in the NIDS domain, for this purpose, we created an experiment bench for generating samples of the various neural network architectures with customizable parameters and tested these neural networks using well-known benchmark dataset KDD99. Also, we developed a simulation platform which acts like a live network, to examine the created neural network. This simulation gives us an opinion about their performance about how trained network behaves in the near-real-time environment.

Many review research [97] [63] [102] [73], have shown that the usage of the artificial neural networks in network intrusion detection domain is increasing. However, despite the popularity of the ANNs and the capability of the general approximation, constructing or determining a domain-specific topology remains a time consuming and a challenging topic. In this research, we aim to propose a comparison guideline for the artificial neural network topologies and components in the NIDS domain. The results and findings of this research will help the researchers and practitioners.

Further, we collected the test results from the experiment bench according to defined evaluation measurement criteria in Section 3.6. Accumulated outcomes are examined and investigated according to research questions in Section 1.1.

At the end of the analysis, the neural networks which used the auto-encoder network architecture have outperformed the other evaluated architectures with 94.71% of accuracy and 0.35% false positive rate. On the other hand, Evolutionary Neural Networks architecture reached the highest mark in the test; however, training time

and detection time was the highest of them all. Moreover, the feed forward neural network architectures also had considerable results in terms of false positive rate of 1.23% and accuracy of 92.34%. Lastly, in the simulation environment, recurrent neural network architecture and auto-encoder neural network architecture resulted with the highest percentages of accuracy and false positive rate.

Subsequently, training costs have the major value for the evaluation of the neural networks. ENN has the worst training and prediction time in all of the tested neural networks, because of its architectural structure. Despite the best training and detection time that were recorded in tests, worst detection accuracy and highest false positive rate belong to CNN network architecture.

In our experiment, we observed a correlation between network size and dropout value with the used KDD99 dataset. When the depth of the network increases, neural network more tend to face with overfitting. However, decreasing the depth of the architecture cause the miss detections. Moreover, the Adam optimizer performed better than the SGD optimizer. Adam optimizer produced lower false positive of 5.65% and higher accuracy result of 4.29%.

Just as crucial as the optimizer; activation functions also have a significant effect on the performance. Neural networks which employed logistic function as an activation function tend to give a better result than the other tested activation functions.

5.1 Discussion of Results

Comparative studies [21] [23] [49] in this field focus on one type of network architecture and does not mention about the preliminary parameters of the created neural networks. Many other studies [57] [35] [71] focus on comparing artificial neural network performances with different techniques used in NIDS. These research gives limited information about the employed neural network architectures. Because of these reasons, finding the effects of the components which play the crucial role in the performance of an ANN, cannot be determined.

Chiba et al. [21] compared parameters of the neural networks to find an architecture

for higher detection rate and lower false positive rate. In their research, they generated and examined 48 different feed-forward neural networks in KDD99 dataset. After the examination, logistic activation functions yields with the higher accuracy and lowest false rate. Our findings support the results of this study, however, the scope of the experiments performed by this study only cover the feed-forward neural network architecture. Thus, comparison of the other type of architectures was missing in this research.

Shone et al. [82] compared the deep learning architectures with the shallow network architectures and, they proposed an autoencoder based deep learning model for the NIDS. Their results demonstrate that the proposed architecture has the highest detection rate. Shone et al.'s results concluded that the autoencoder based architectures have the highest accuracy value. However, in this study, they used one activation function and one optimization function. Because of this reason, the effect of the used activation functions and optimization functions and other parameters cannot be determined with their research.

Furthermore, Dao et al. [23] also examined the performance of feed-forward neural networks. In their experiments, they chose only an SGD optimization method and changed the parameters around the selected optimization method. Conversely, this study just contains a comparison of the limited number of activation function with only one optimization function on the feed-forward neural network.

On the other hand, Kim et al. [49] completed a comparison of the depth of the neural networks. This study was also performed on the SGD optimization method, and as an activation function, they used the hyperbolic sigmoid function. They showed that incrementing the hidden layer count decreases the reliability of the network and increases the computation time. Our results show that the deeper feed-forward neural network has the same behavior, but we found complex and more profound auto-encoder neural network resulted in a better performance ratio.

The analysis was carried out according to the determined research questions and the results gave the following important results to the practitioners and researchers:

- The auto-encoder network topology gave the highest performance among the

topologies tested. RNN topologies on the flow data provided a noteworthy performance. In addition, it was observed that CNN topologies failed in the NIDS area.

- In spite of excellent accuracy rates, the detection times of ENN topologies were higher than the other topologies. In order to improve the detection time performance of ENN structures, it is observed that more powerful machines are needed in the current test environment. In the same way, auto-encoder networks have been the most successful topology in experiments with high accuracy and low FAR values in combination with short detection time.
- The activation functions of the logistic function family have reached higher values than the other functions in the NIDS field. The softplus, which is a logistic function, has the highest performance values. Although the ReLu activation function gave higher results in RNN architectures, it was observed that it gave the lowest results when used in auto-encoder topology. In the same way, when the optimizer algorithms are determined as pivot, Adam optimizer algorithm has reached the highest performance values in all topologies used according to SGD optimizer algorithm. In addition, the ENN topology selected Adam as the optimal optimizer algorithm according to the NIDS domain data. Another important effect on FAR is the dropout value. Using the dropout value, ANN topologies achieved a better performance than those who did not use dropout.

5.2 Limitations and Further Research

Testing all variations of the neural network was time expensive and was a slow process. For this reason, increasing the variance of the testing parameters like optimizer function and depth of the networks could be considered as a future work with a more capable, modern GPU supported experimentation environment. Of equal importance, a simulation environment which has a higher bandwidth and throughput, which is also capable of creating a higher ratio of data will show more detailed results.

Bibliography

- [1] M. Saniee Abadeh, J. Habibi, and C. Lucas. “Intrusion Detection Using a Fuzzy Genetics-Based Learning Algorithm”. In: *Journal of Network and Computer Applications* 30.1 (Jan. 2007). 00200, pp. 414–428. DOI: 10.1016/j.jnca.2005.05.002.
- [2] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. “A Novel SVM-kNN-PSO Ensemble Method for Intrusion Detection System”. In: *Applied Soft Computing* 38 (Jan. 2016). 00121, pp. 360–372. DOI: 10.1016/j.asoc.2015.10.011.
- [3] Iftikhar Ahmad, Azween B Abdullah, and Abdullah S Alghamdi. “Application of Artificial Neural Network in Detection of DOS Attacks”. In: *Proceedings of the 2nd International Conference on Security of Information and Networks*. ACM, 2009, pp. 229–234. ISBN: 1-60558-412-6.
- [4] Iftikhar Ahmad, Azween B Abdullah, and Abdullah S Alghamdi. “Application of Artificial Neural Network in Detection of Probing Attacks”. In: *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium On*. Vol. 2. IEEE, 2009, pp. 557–562. ISBN: 1-4244-4681-3.
- [5] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. “Intrusion Detection Using Deep Belief Networks”. In: *Aerospace and Electronics Conference (NAECON), 2015 National*. IEEE, 2015, pp. 339–344. ISBN: 1-4673-7565-9.
- [6] Md Zahangir Alom and Tarek M Taha. “Network Intrusion Detection for Cyber Security on Neuromorphic Computing System”. In: *Neural Networks (IJCNN), 2017 International Joint Conference On*. IEEE, 2017, pp. 3830–3837. ISBN: 1-5090-6182-7.
- [7] Md Zahangir Alom and Tarek M Taha. “Network Intrusion Detection for Cyber Security Using Unsupervised Deep Learning Approaches”. In: *Aerospace*

and Electronics Conference (NAECON), 2017 IEEE National. IEEE, 2017, pp. 63–69. ISBN: 1-5386-3200-4.

- [8] Khaled Alrawashdeh and Carla Purdy. “Reducing Calculation Requirements in FPGA Implementation of Deep Learning Algorithms for Online Anomaly Intrusion Detection”. In: *Aerospace and Electronics Conference (NAECON), 2017 IEEE National*. IEEE, 2017, pp. 57–62. ISBN: 1-5386-3200-4.
- [9] Christine M Anderson-Cook. “Practical Genetic Algorithms”. In: *Journal of the American Statistical Association* 100.471 (Sept. 2005). 04437, pp. 1099–1099. DOI: 10.1198/jasa.2005.s45.
- [10] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. “An Evolutionary Algorithm That Constructs Recurrent Neural Networks”. In: *IEEE transactions on Neural Networks* 5.1 (1994), pp. 54–65.
- [11] Stefan Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy*. 2000, p. 27.
- [12] R Can Aygun and A Gokhan Yavuz. “Network Anomaly Detection with Stochastically Improved Autoencoder Based Models”. In: *Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference On*. IEEE, 2017, pp. 193–198. ISBN: 1-5090-6644-6.
- [13] Rebecca Bace and Peter Mell. *NIST Special Publication on Intrusion Detection Systems*. 00696. USA: National Institute of Standards and Technology, 2001, p. 53.
- [14] Rachid Beghdad. “Critical Study of Neural Networks in Detecting Intrusions”. In: *Computers & security* 27.5-6 (2008), pp. 168–175.
- [15] Sunny Behal and Krishan Kumar. “Trends in Validation of DDoS Research”. In: *Procedia Computer Science* 85 (2016). 00018, pp. 7–15. DOI: 10.1016/j.procs.2016.05.170.
- [16] Loïc Bontemps, James McDermott, and Nhien-An Le-Khac. “Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks”. In: *International Conference on Future Data and Security Engineering*. Springer, 2016, pp. 141–152.

- [17] H. Bourlard and Y. Kamp. “Auto-Association by Multilayer Perceptrons and Singular Value Decomposition”. In: *Biological Cybernetics* 59.4-5 (Sept. 1988). 00874, pp. 291–294. DOI: 10.1007/BF00332918.
- [18] Carlos A. Catania and Carlos García Garino. “Automatic Network Intrusion Detection: Current Techniques and Open Issues”. In: *Computers & Electrical Engineering*. Special Issue on Recent Advances in Security and Privacy in Distributed Communications and Image Processing 38.5 (Sept. 1, 2012), pp. 1062–1072. DOI: 10.1016/j.compeleceng.2012.05.013.
- [19] Pravin Chandra and Yogesh Singh. “An Activation Function Adapting Training Algorithm for Sigmoidal Feedforward Networks”. In: *Neurocomputing* 61 (Oct. 2004). 00070, pp. 429–437. DOI: 10.1016/j.neucom.2004.04.001.
- [20] Wun-Hwa Chen, Sheng-Hsun Hsu, and Hwang-Pin Shen. “Application of SVM and ANN for Intrusion Detection”. In: *Computers & Operations Research* 32.10 (Oct. 2005). 00308, pp. 2617–2634. DOI: 10.1016/j.cor.2004.03.019.
- [21] Zouhair Chiba et al. “A Novel Architecture Combined with Optimal Parameters for Back Propagation Neural Networks Applied to Anomaly Network Intrusion Detection”. In: *Computers & Security* 75 (June 2018). 00001, pp. 36–58. DOI: 10.1016/j.cose.2018.01.023.
- [22] Emilio Corchado and Álvaro Herrero. “Neural Visualization of Network Traffic Data for Intrusion Detection”. In: *Applied Soft Computing* 11.2 (2011), pp. 2042–2056.
- [23] Vu N P Dao and Rao Vemuri. “A Performance Comparison of Different Back Propagation Neural Networks Methods in Computer Network Intrusion Detection”. In: (). 00054, p. 7.
- [24] D. Dasgupta and F. Gonzalez. “An Immunity-Based Technique to Characterize Intrusions in Computer Networks”. In: *IEEE Transactions on Evolutionary Computation* 6.3 (June 2002). 00505, pp. 281–291. DOI: 10.1109/TEVC.2002.1011541.

- [25] Eduardo De la Hoz et al. “PCA Filtering and Probabilistic SOM for Network Intrusion Detection”. In: *Neurocomputing* 164 (2015), pp. 71–81.
- [26] Emiro De la Hoz et al. “Feature Selection by Multi-Objective Optimisation: Application to Network Anomaly Detection by Hierarchical Self-Organising Maps”. In: *Knowledge-Based Systems* 71 (2014), pp. 322–338.
- [27] Hervé Debar, Marc Dacier, and Andreas Wespi. “A Revised Taxonomy for Intrusion-Detection Systems”. In: *Annales Des Télécommunications* 55 (Issue 7–8 2000). 01343, pp. 361–378.
- [28] Dorothy E Denning. “An Intrusion-Detection Model”. In: *IEEE Transactions on software engineering* 2 (1987). 04895, pp. 222–232. DOI: 10.1109/TSE.1987.232894.
- [29] J.E. Dickerson and J.A. Dickerson. “Fuzzy Network Profiling for Intrusion Detection”. In: *PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.00TH8500)*. PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society - NAFIPS. 00330. Atlanta, GA, USA: IEEE, 2000, pp. 301–306. ISBN: 978-0-7803-6274-1. DOI: 10.1109/NAFIPS.2000.877441.
- [30] Shan Ding and Genying Wang. “Research on Intrusion Detection Technology Based on Deep Learning”. In: *Computer and Communications (ICCC), 2017 3rd IEEE International Conference On*. IEEE, 2017, pp. 1474–1478. ISBN: 1-5090-6352-8.
- [31] M Dondo and J Treurniet. *Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System*. DEFENCE RESEARCH AND DEVELOPMENT CANADAOTTAWA (ONTARIO), 2004.
- [32] Jeffrey L. Elman. “Finding Structure in Time”. In: *Cognitive Science* 14.2 (Mar. 1990). 09789, pp. 179–211. DOI: 10.1207/s15516709cog1402_1.
- [33] Fahimeh Farahnakian and Jukka Heikkonen. “A Deep Auto-Encoder Based Approach for Intrusion Detection System”. In: *Advanced Communication*

Technology (ICACT), 2018 20th International Conference On. IEEE, 2018, pp. 178–183. ISBN: 979-11-88428-01-4.

- [34] Emile Fiesler. “Neural Network Classification and Formalization”. In: *Computer Standards and Interfaces* 16.3 (1994). 00000, pp. 231–240.
- [35] P. García-Teodoro et al. “Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges”. In: *Computers & Security* 28.1 (Feb. 1, 2009). 00001, pp. 18–28. DOI: 10.1016/j.cose.2008.08.003.
- [36] Maryam Ghanbari, Witold Kinsner, and Ken Ferens. “Detecting a Distributed Denial of Service Attack Using a Pre-Processed Convolutional Neural Network”. In: *Electrical Power and Energy Conference (EPEC), 2017 IEEE.* IEEE, 2017, pp. 1–6. ISBN: 1-5386-0817-0.
- [37] M Govindarajan and R M Chandrasekaran. “Intrusion Detection Using Neural Based Hybrid Classification Methods”. In: *Computer networks* 55.8 (2011), pp. 1662–1671.
- [38] Hongmei Deng, Qing-An Zeng, and D.P. Agrawal. “SVM-Based Intrusion Detection System for Wireless Ad Hoc Networks”. In: *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484).* 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484). 00111. Orlando, FL, USA: IEEE, 2003, 2147–2151 Vol.3. ISBN: 978-0-7803-7954-1. DOI: 10.1109/VETEFC.2003.1285404.
- [39] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.
- [40] Laheeb M Ibrahim, Dujan T Basheer, and Mahmud S Mahmud. “A Comparison Study for Intrusion Database (Kdd99, Nsl-Kdd) Based on Self Organization Map (SOM) Artificial Neural Network”. In: *Journal of Engineering Science and Technology* 8.1 (2013), pp. 107–119.
- [41] Dennis Ippoliti and Xiaobo Zhou. “A-GHSOM: An Adaptive Growing Hierarchical Self Organizing Map for Network Anomaly Detection”. In: *Journal of Parallel and Distributed Computing* 72.12 (2012), pp. 1576–1590.

- [42] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. “Artificial Neural Networks: A Tutorial”. In: *Computer* 29.3 (Mar. 1996). 02276, pp. 31–44. DOI: 10 . 1109/2 . 485891.
- [43] Samireh Jalali and Claes Wohlin. “Systematic Literature Studies: Database Searches vs. Backward Snowballing”. In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2012, pp. 29–38. ISBN: 1-4503-1056-7.
- [44] Jungwon Kim and P.J. Bentley. “Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with a Negative Selection Operator”. In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 2001 Congress on Evolutionary Computation. Vol. 2. 00361. Seoul, South Korea: IEEE, 2001, pp. 1244–1252. ISBN: 978-0-7803-6657-2. DOI: 10 . 1109/CEC . 2001 . 934333.
- [45] Peyman Kabiri and Ali A Ghorbani. “Research on Intrusion Detection and Response: A Survey.” In: *IJ Network Security* 1.2 (2005). 00277, pp. 84–102.
- [46] Amin Karami. “An Anomaly-Based Intrusion Detection System in Presence of Benign Outliers with Visualization Capabilities”. In: *Expert Systems with Applications* (2018).
- [47] H Günes Kayacık, A Nur Zincir-Heywood, and Malcolm I Heywood. “Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets”. In: *Proceedings of the Third Annual Conference on Privacy, Security and Trust*. 00004. 2005, p. 6.
- [48] Oğuz Kaynar et al. “Intrusion Detection with Autoencoder Based Deep Learning Machine”. In: *Signal Processing and Communications Applications Conference (SIU), 2017 25th*. IEEE, 2017, pp. 1–4. ISBN: 1-5090-6494-X.
- [49] Daniel E Kim and Mikhail Gofman. “Comparison of Shallow and Deep Neural Networks for Network Intrusion Detection”. In: *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*. IEEE, 2018, pp. 204–208. ISBN: 1-5386-4649-8.
- [50] Kwangjo Kim and Muhamad Erza Aminanto. “Deep Learning in Intrusion Detection Perspective: Overview and Further Challenges”. In: *Big Data and*

Information Security (IWBIS), 2017 International Workshop On. IEEE, 2017, pp. 5–10. ISBN: 1-5386-2038-3.

- [51] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 22, 2014). 15852. arXiv: 1412.6980 [cs].
- [52] Barbara Kitchenham. *Procedures for Performing Systematic Reviews*. Technical Report TR/SE-0401. 03120. Keele University, 2004, p. 33.
- [53] Teuvo Kohonen. “Self-Organized Formation of Topologically Correct Feature Maps”. In: *Biological Cybernetics* 43.1 (1982). 09950, pp. 59–69. DOI: 10.1007/BF00337288.
- [54] Chan Man Kuok, Ada Fu, and Man Hon Wong. “Mining Fuzzy Association Rules in Databases”. In: *ACM SIGMOD Record* 27.1 (Mar. 1, 1998). 00653, pp. 41–46. DOI: 10.1145/273244.273257.
- [55] Donghwoon Kwon et al. “A Survey of Deep Learning-Based Network Anomaly Detection”. In: *Cluster Computing* (2017), pp. 1–13.
- [56] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. “Intrusion Detection: A Survey”. In: *Managing Cyber Threats: Issues, Approaches, and Challenges*. Ed. by Vipin Kumar, Jaideep Srivastava, and Aleksandar Lazarevic. Massive Computing. 00192. Boston, MA: Springer US, 2005, pp. 19–78. ISBN: 978-0-387-24230-9. DOI: 10.1007/0-387-24230-9_2.
- [57] Aleksandar Lazarevic et al. “A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection”. In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003, pp. 25–36.
- [58] Y. Lecun et al. “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11 (Nov./1998). 15840, pp. 2278–2324. DOI: 10.1109/5.726791.
- [59] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *nature* 521.7553 (2015), p. 436.
- [60] Brian Lee et al. “Comparative Study of Deep Learning Models for Network Intrusion Detection”. In: *SMU Data Science Review* 1.1 (2018), p. 8.

- [61] Yang Li and Li Guo. “An Active Learning Based TCM-KNN Algorithm for Supervised Network Intrusion Detection”. In: *Computers & Security* 26.7-8 (Dec. 2007). 00123, pp. 459–467. DOI: 10.1016/j.cose.2007.10.002.
- [62] Jingxi Liang, Wen Zhao, and Wei Ye. “Anomaly-Based Web Attack Detection: A Deep Learning Approach”. In: *Proceedings of the 2017 VI International Conference on Network, Communication and Computing*. ACM, 2017, pp. 80–85. ISBN: 1-4503-5366-5.
- [63] Hung-Jen Liao et al. “Intrusion Detection System: A Comprehensive Review”. In: *Journal of Network and Computer Applications* 36.1 (Jan. 2013). 00507, pp. 16–24. DOI: 10.1016/j.jnca.2012.09.004.
- [64] Yihua Liao and V.Rao Vemuri. “Use of K-Nearest Neighbor Classifier for Intrusion Detection”. In: *Computers & Security* 21.5 (Oct. 2002). 00000, pp. 439–448. DOI: 10.1016/S0167-4048(02)00514-X.
- [65] Yuchen Liu, Shengli Liu, and Xing Zhao. “Intrusion Detection Algorithm Based on Convolutional Neural Network”. In: *DEStech Transactions on Engineering and Technology Research (iceta 2017)*.
- [66] Parisa Lotfallahtabrizi and Yasser Morgan. “A Novel Host Intrusion Detection System Using Neural Network”. In: *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*. IEEE, 2018, pp. 124–130. ISBN: 1-5386-4649-8.
- [67] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943). 16645, pp. 115–133. DOI: 10.1007/BF02478259.
- [68] John McHugh. “Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory”. In: *ACM Transactions on Information and System Security* 3.4 (Nov. 1, 2000). 01170, pp. 262–294. DOI: 10.1145/382912.382923.
- [69] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. “A Mean Field View of the Landscape of Two-Layer Neural Networks”. In: *Proceedings of the*

National Academy of Sciences 115.33 (Aug. 14, 2018). 00000, E7665–E7671.
DOI: 10.1073/pnas.1806579115.

- [70] Ionita Mihai-Gabriel and Patriciu Victor-Valeriu. “Achieving DDoS Resiliency in a Software Defined Network by Intelligent Risk Assessment Based on Neural Networks and Danger Theory”. In: *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium On*. IEEE, 2014, pp. 319–324. ISBN: 1-4799-5338-5.
- [71] Rodrigo Moraes, João Francisco Valiati, and Wilson P. Gavião Neto. “Document-Level Sentiment Classification: An Empirical Comparison between SVM and ANN”. In: *Expert Systems with Applications* 40.2 (Feb. 2013). 00337, pp. 621–633. DOI: 10.1016/j.eswa.2012.07.059.
- [72] S. Mukkamala, G. Janoski, and A. Sung. “Intrusion Detection Using Neural Networks and Support Vector Machines”. In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No.02CH37290)*. 2002 International Joint Conference on Neural Networks (IJCNN). 00802. Honolulu, HI, USA: IEEE, 2002, pp. 1702–1707. ISBN: 978-0-7803-7278-8. DOI: 10.1109/IJCNN.2002.1007774.
- [73] Mehmet Uğur ÖNEY and Serhat PEKER. “The Use of Artificial Neural Networks in Network Intrusion Detection: A Systematic Review”. In: *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. 00000. IEEE. 2018, pp. 1–6.
- [74] José A. Onieva, Javier Lopez, and Jianying Zhou. *Secure Multi-Party Non-Repudiation Protocols and Applications*. Vol. 43. Advances in Information Security. 00023. Boston, MA: Springer US, 2009. ISBN: 978-0-387-75629-5 978-0-387-75630-1. DOI: 10.1007/978-0-387-75630-1.
- [75] V K Pachghare, Parag Kulkarni, and Deven M Nikam. “Intrusion Detection System Using Self Organizing Maps”. In: *Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. International Conference On*. IEEE, 2009, pp. 1–5. ISBN: 1-4244-4710-0.
- [76] Jianli Pan. *A Survey of Network Simulation Tools: Current Status and Future Development*. 00000.

- [77] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. “A Hybrid Intelligent Approach for Network Intrusion Detection”. In: *Procedia Engineering* 30 (2012), pp. 1–9.
- [78] Simon T Powers and Jun He. “A Hybrid Artificial Immune System and Self Organising Map for Network Intrusion Detection”. In: *Information Sciences* 178.15 (2008), pp. 3024–3042.
- [79] Benjamin J Radford et al. “Network Traffic Anomaly Detection Using Recurrent Neural Networks”. In: *arXiv preprint arXiv:1803.10769* (2018).
- [80] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” In: *Psychological Review* 65.6 (1958). 08443, pp. 386–408. DOI: 10.1037/h0042519.
- [81] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors”. In: *Nature* 323.6088 (Oct. 1986). 16419, pp. 533–536. DOI: 10.1038/323533a0.
- [82] Nathan Shone et al. “A Deep Learning Approach to Network Intrusion Detection”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.1 (2018), pp. 41–50.
- [83] Jimmy Shun and Heidar A Malki. “Network Intrusion Detection System Using Neural Networks”. In: *Natural Computation, 2008. ICNC’08. Fourth International Conference On*. Vol. 5. IEEE, 2008, pp. 242–246. ISBN: 0-7695-3304-3.
- [84] P Sibi, S Allwyn Jones, and P Siddarth. “ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS”. In: . Vol. 47 (2005). 00105, p. 5.
- [85] Yogesh Singh and Pravin Chandra. “A Class +1 Sigmoidal Activation Functions for FFANNs”. In: *Journal of Economic Dynamics and Control* 28.1 (Oct. 2003). 00000, pp. 183–187. DOI: 10.1016/S0165-1889(02)00157-4.
- [86] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *The Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

- [87] Gary Stein et al. “Decision Tree Classifier for Network Intrusion Detection with GA-Based Feature Selection”. In: *Proceedings of the 43rd Annual Southeast Regional Conference on - ACM-SE 43*. The 43rd Annual Southeast Regional Conference. Vol. 2. 00232. Kennesaw, Georgia: ACM Press, 2005, p. 136. ISBN: 978-1-59593-059-0. DOI: 10.1145/1167253.1167288.
- [88] S.J. Stolfo et al. “Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project”. In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. DARPA Information Survivability Conference and Exposition. DISCEX'00. Vol. 2. 00532. Hilton Head, SC, USA: IEEE Comput. Soc, 1999, pp. 130–144. ISBN: 978-0-7695-0490-2. DOI: 10.1109/DISCEX.2000.821515.
- [89] Felipe Petroski Such et al. “Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning”. In: (Dec. 18, 2017). 00083. arXiv: 1712.06567 [cs].
- [90] A.H. Sung and S. Mukkamala. “Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks”. In: *2003 Symposium on Applications and the Internet, 2003. Proceedings*. 2003 Symposium on Applications and the Internet (SAINT 2003). 00483. Orlando, FL, USA: IEEE Comput. Soc, 2003, pp. 209–216. ISBN: 978-0-7695-1872-5. DOI: 10.1109/SAINT.2003.1183050.
- [91] Arman Tajbakhsh, Mohammad Rahmati, and Abdolreza Mirzaei. “Intrusion Detection Using Fuzzy Association Rules”. In: *Applied Soft Computing 9.2* (Mar. 2009). 00238, pp. 462–469. DOI: 10.1016/j.asoc.2008.06.001.
- [92] Mahbod Tavallaee et al. “A Detailed Analysis of the KDD CUP 99 Data Set”. In: *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium On*. IEEE, 2009, pp. 1–6. ISBN: 1-4244-3763-6.
- [93] Jihyun Kim Thi-Thu-Huong Le and Howon Kim. “An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization”. In: *International Conference On Platform Technology And Service* (2017).

- [94] Jingjing Tian and Ping' An Li. "An Intrusion Detection Algorithm of Dynamic Recursive Deep Belief Networks". In: *Proceedings of the 2017 International Conference on Information Technology*. ACM, 2017, pp. 180–183. ISBN: 1-4503-6351-2.
- [95] Adel Nadjaran Toosi and Mohsen Kahani. "A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers". In: *Computer Communications* 30.10 (July 2007). 00251, pp. 2201–2212. DOI: 10.1016/j.comcom.2007.05.002.
- [96] Nam Nhat Tran, Ruhul Sarker, and Jiankun Hu. "An Approach for Host-Based Intrusion Detection System Design Using Convolutional Neural Network". In: *International Conference on Mobile Networks and Management*. Springer, 2017, pp. 116–126.
- [97] Chih-Fong Tsai et al. "Intrusion Detection by Machine Learning: A Review". In: *Expert Systems with Applications* 36.10 (2009), pp. 11994–12000.
- [98] Nguyen Thanh Van, Tran Ngoc Thinh, and Le Thanh Sach. "An Anomaly-Based Network Intrusion Detection System Using Deep Learning". In: *System Science and Engineering (ICSSE), 2017 International Conference On*. IEEE, 2017, pp. 210–214. ISBN: 1-5386-3422-8.
- [99] Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. "An Anomaly Detection Method to Detect Web Attacks Using Stacked Auto-Encoder". In: *Fuzzy and Intelligent Systems (CFIS), 2018 6th Iranian Joint Congress On*. IEEE, 2018, pp. 131–134. ISBN: 1-5386-2836-8.
- [100] R Vinayakumar, K P Soman, and P Poornachandran. "Applying Convolutional Neural Network for Network Intrusion Detection". In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2017)*, pp. 1222–1228.
- [101] Gang Wang et al. "A New Approach to Intrusion Detection Using Artificial Neural Networks and Fuzzy Clustering". In: *Expert systems with applications* 37.9 (2010), pp. 6225–6232.
- [102] Shelly Xiaonan Wu and Wolfgang Banzhaf. "The Use of Computational Intelligence in Intrusion Detection Systems: A Review". In: *Applied Soft Com-*

puting 10.1 (Jan. 2010). 00633, pp. 1–35. DOI: 10.1016/j.asoc.2009.06.019.

- [103] Yaping Zhang et al. “Intrusion Detection Method Based on Improved Growing Hierarchical Self-Organizing Map”. In: *Transactions of Tianjin University* 22.4 (2016), pp. 334–338.
- [104] Guangzhen Zhao, Cuixiao Zhang, and Lijuan Zheng. “Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network”. In: *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference On*. Vol. 1. IEEE, 2017, pp. 639–642. ISBN: 1-5386-3221-7.



CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Öney, Mehmet Uğur

Nationality: Turkish (TC)

Date and Place of Birth: 11.06.1984, Pazar/Rize **Phone:** 0 535 8851217

EDUCATION

Degree	Institution	Year of Graduation
B.S.	Atılım University, Dept. of Computer Engineering	2008

PROFESSIONAL EXPERIENCE

Enrollment	Place	Year
QA Manager	May Cyber Technology, Ankara, Turkey	2012-Cont

PUBLICATIONS

1. Öney, M.U., Peker, S., "The Use of Artificial Neural Networks in Network Intrusion Detection: A Systematic Review", IDAP, 2018
2. Daneshgadeh, S., Öney M.U., Kemmerich, T., Baykal, N., "A Simulation Environment for Cyber-Security Attack Analysis Based on Network Traffic Logs", Chapter 10, Modeling and Simulation of Complex Networks, IET, 2018
3. Öney, M.U., Çevik, A., Çağıltay, N. and Kılıç, Ö., "Topluluk Zekâsı Yönetimi ve Optimizasyonu". Akademik Bilisim Kutahya, 2007