T.C.
ALTINBAŞ UNIVERSITY
Institute of Graduate Studies
Electrical and Computer Engineering

DEEP LEARNING BASED FULLY AUTONOMOUS NETWORK INTRUSION DETECTION SYSTEM

Zaid Farhan Dakhil DAKHIL

Master Thesis

Supervisor

Prof. Dr. Osman Nuri UÇAN

Istanbul, 2020

DEEP LEARNING BASED FULLY AUTONOMOUS NETWORK INTRUSION DETECTION SYSTEM

by

Zaid Farhan Dakhil DAKHIL

Electrical and Computer Engineering

Submitted to the Institute of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2020

The thesis titled “DEEP LEARNING BASED FULLY AUTONOMOUS NETWORK INTRUSION DETECTION SYSTEM” prepared and presented by “Zaid Farhan Dakhil DAKHIL” was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

Prof. Dr. Osman Nuri UÇAN

Supervisor

Thesis Defense Jury Members:

Prof. Dr. Osman Nuri UÇAN	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Izzet Parug DURU	Vocational School, Istanbul Gedik University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies:
____/____/____

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.

Zaid Farhan Dakhil DAKHIL

DEDICATION

I devote and pledge this research work to my supervisor who is salient for guiding me through whole research work as well as my family for always assisting me in my hard time.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor Prof. Dr. Osman Nuri Uçan for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Prof. Dr. Osman Nuri Uçan a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work. He helped me by looking outside of all the deep learning based information, and considering the deep learning based knowledge from a more than just from the productivity point of view. He patiently explained to me a great number of technical details with regards to deep convolutional neural network algorithm, for example how to deal with deep convolutional neural network algorithm and to provide insight into the reliability of those results. Without his time and help none of this research would have been possible, and I'm grateful to him for giving me the opportunity to work on this interesting domain.

ABSTRACT

DEEP LEARNING BASED FULLY AUTONOMOUS NETWORK INTRUSION DETECTION SYSTEM

DAKHIL, Zaid Farhan Dakhil,

M.S, Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Osman Nuri UÇAN

Date: 02 / 2020

Pages: 63

The aim of this thesis is to use develop and evaluate a deep learning based fully autonomous network intrusion detection system. The growing adoption of deep learning brings with it the increased participation of network intrusion in dataset, and as such novel method of deep convolutional neural network attack classifications are needed. Evaluation and results of experiments are provided from using a deep feed forward fully connected neural network classifier on the NID Kaggle dataset. This dataset is splitted in 80% for training and 20% for testing with 10-cross fold for validation. In this work, network intrusion of a network was presented that evaluated the use of deep fully connected deep convolutional neural networks for classifying network flows as benign or malignant. The deep convolutional neural network performs well in its ability to classify network flows as malignant and benign, achieving an accuracy for malignant at 90% training as high as 228 percentile out of 250 however for benign it achieved a score of 69 percentile. To perform this work, we used the MATLAB R2019b with

deep learning toolbox and recent benchmark dataset was used. The experiments showed results that outperformed other leading techniques when using deep convolutional neural networks for classification of network flows. Further, it was shown that the use of flow statistics generated by the DCNN were more accurate in detecting the intrusion, in addition to a technique of dimensionality reduction of the categorical variables using the deep convolutional neural network algorithm, enabled the application to achieve an extra-ordinary accuracy, a detection rate of 0.9993, and a false positive rate of only 0.0003. After the algorithm was tested the result column predicts whether the packet is an attacking packet or not and shows 96.77% accuracy in just 0.57s. As can been seen by the results, the deep convolutional neural network performs well in its ability to classify network flows as malicious and benign, achieving a low false positive rate of only 0.000127 and achieving an F1-score of 0.9990. The trained network achieved 0.9987 for precision, 0.9993 for recall and 0.9990 for F1-score in case of DCNN with IPs while 0.9784 for precision, 0.9677 for recall and 0.9730 for F1-score in case of DCNN without IPs.

Keywords: Deep Learning, Network Security, Intrusion Detection, Evaluation, Deep Convolutional Neural Network, Autonomous System, Optimization

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
ANN	:	Artificial Neural Network
DCNN	:	Deep Convolutional Neural Network
DL	:	Deep Learning
GPU	:	Graphic Processing Unit
IDS	:	Intrusion Detection System
ICS	:	Intrusion Classification System
LR	:	Logistic Regression
IoT	:	Internet Of Things
ML	:	Machine Learning
RF	:	Random Forest
SVM	:	Support Vector Machine

1. INTRODUCTION

1.1 INTRODUCTION

This work will allow for assessment of deep convolutional neural network method accuracy on smaller feature subsets and for making a fairer comparison on network intrusion traffic detection and classification for attacks with more advanced deep learning method. These adversaries primarily consist of three groups: Nation-states, cybercriminals, and activist groups (e.g. Anonymous). Their motivations include espionage, political and ideological interests, and financial gain [1]. While their motivations may be varied, their aims are the same: leverage the connectivity of society through the Internet to carry out a malicious goal. These goals can vary from theft of intellectual property, denial of service, disruption of business, theft of personally identifiable information (PII) or payment card information (PCI), financial fraud, demanding a ransom (i.e. ransom ware), destruction of physical property (e.g. Conflicker Warm) [2], and other nefarious purposes.

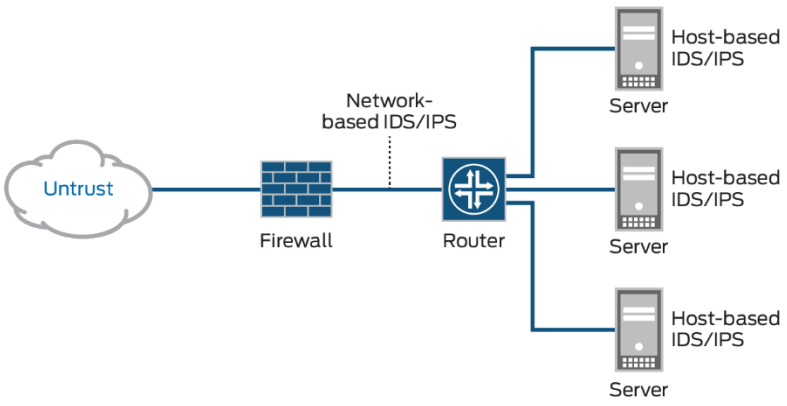

Figure 1.1: A type of network intrusion detection and prevention system [3].

Due to the opportunity existing for bad actors to conduct this malicious activity, it is imperative that all cyber infrastructure be secured and protected from misuse. Among the many cyber infrastructure systems that exist (e.g. critical infrastructure, cyber-physical systems, SCADA systems, Internet of Things, etc.), this thesis focuses on the protection of networks maintained and intrusion detection using deep learning operated by enterprises, both large and small, from being exploited by bad actors [3]. Often, bad actors seek to gain access to enterprise networks for

a variety of reasons including but not limited to theft of intellectual property, access to trade secrets, insider information for illegal stock trading, disruption of business (e.g. Sony attacks in 2016) [4], theft of financial information (e.g. Target breach in 2016).

1.2 PROBLEM STATEMENT

The challenge inherent in information systems and networks from being compromised is the fact that they are built upon complex layers of software. Due to its growing complexity, software often contains vulnerabilities that can be found and exploited by an attacker. Even if the software of a given security tool uses proven algorithms and standards for security, it can still suffer from a bad implementation that leaves a security hole. These risks can be mitigated by putting in place strategies and best practices, such as continuous patching, bug bounty programs, red/blue team exercises, threat hunting, honeypots, honeynets, moving target defense, and vulnerability management programs, to name a few. However, attackers continually try new avenues to compromise defenses by altering their attack strategies and using never before seen techniques. Commonly referred to as a zero-day attack, these types of attacks can be very damaging and frustrating to the defender, as the attacker has developed a new exploit that bypasses the defenses of a given system or software. Therefore, as mentioned previously, the best strategy is to implement a defense-in-depth strategy and expect the inevitable result that with enough time and resources, an attacker will inevitably gain access to the network somewhere along the way. It is paramount that when this occurs, the attack is discovered promptly and quarantined or eliminated before any material harm is done.

1.3 DEFENSE-IN-DEPTH STRATEGY

In order to combat bad actors, a wide array of approaches have been developed in order to stay one step ahead of the adversary [4]. The best overall approach for tackling this problem consists of a defense-in-depth strategy, whereby various security tools, techniques, and mechanisms are employed throughout an organization's ecosystem, both horizontally and vertically at different levels [5]. It is commonly understood that there is no such thing as 100% security [6]. The aim instead is towards managing risk and reducing the surface area available for attack. Security is an intractable problem, as it is impossible to think of all the possible ways an attacker may break through the defenses. A preferred strategy, as suggested by MIT Computer Science and Artificial

Intelligence Laboratory (CSAIL) [7, 8], is to minimize the attack surface, and manage risk by employing a defense-in-depth strategy. Various techniques can be used to reduce surface area vulnerable to attack (e.g., by enforcing access control, multi-factor authentication, network segmentation, and continuous patching) as well as reducing risk by deploying tools at various stages, from the exterior-facing network to the interior network, and down to the individual host-level workstations on the network.

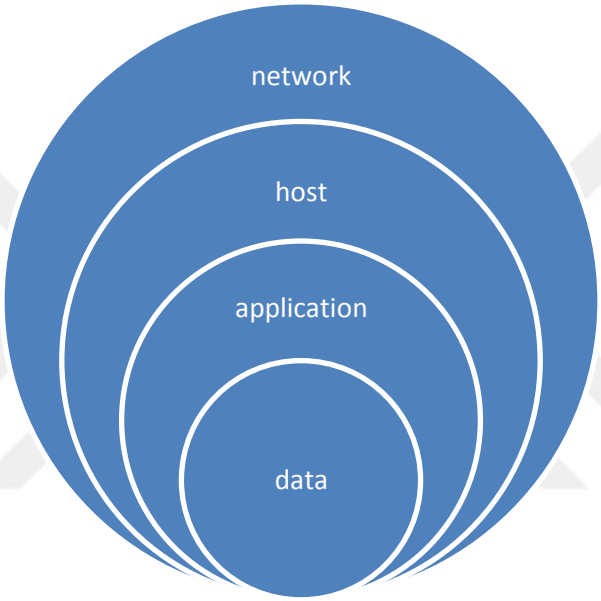

Figure 1.2: An example of defense-in-depth strategy with the data as a source [8].

The challenge inherent in information systems and networks from being compromised is the fact that they are built upon complex layers of software [9]. Due to its growing complexity, software often contains vulnerabilities that can be found and exploited by an attacker. Even if the software of a given security tool uses proven algorithms and standards for security, it can still suffer from a bad implementation that leaves a security hole. However, attackers continually try new avenues to compromise defenses by altering their attack strategies and using never before seen techniques. Commonly referred to as a zero-day attack, these types of attacks can be very damaging and frustrating to the defender, as the attacker has developed a new exploit that bypasses the defenses of a given system or software [10]. Therefore, as mentioned previously, the best strategy is to implement a defense-in-depth strategy and expect the inevitable result that with enough time and resources, an attacker will inevitably gain access to the network

somewhere along the way. It is paramount that when this occurs, the attack is discovered promptly and quarantined or eliminated before any material harm is done.

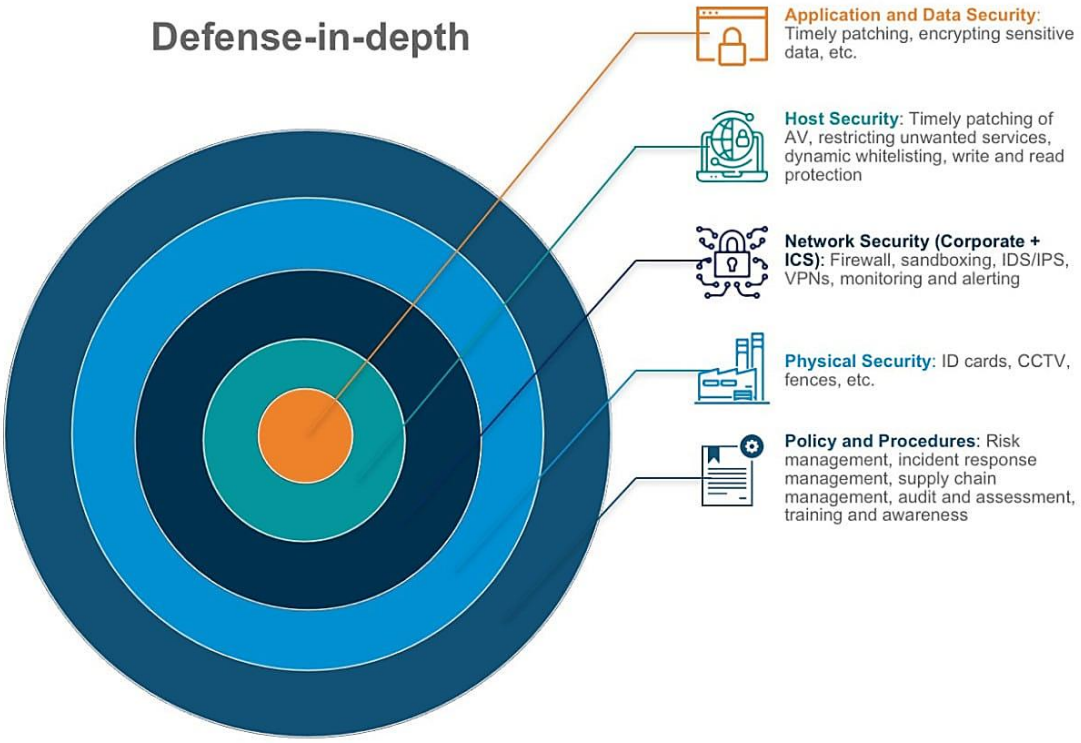

Figure 1.3: A more detailed aspects of defense-in-depth strategy layers with the operation predicted on different layers [10].

1.4 NETWORK INTRUSION DETECTION SYSTEM (NIDS)

One of the most effective ways to protect the confidentiality, integrity, and availability of information and enterprise systems once an attacker has compromised its defenses is to deploy Intrusion Detection Systems (IDS). Intrusion Detection Systems are defined by the National Institute of Technology (NIST) as "software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems" [11]. Intrusion Detection is the art and science of finding attackers that have bypassed preventive defense mechanisms such as firewalls, access control, and other protection mechanisms further up or down the stack. More formally, Intrusion Detection is defined by NIST as the "process of monitoring the events occurring in a computer system or network and

analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices [12]. Host-based intrusion detection systems monitor and control data coming from an individual workstation using tools and techniques such as host-based firewalls, anti-virus/anti-malware agents, data-loss prevention agents, and via monitoring system call trees [13]. Network-based defenses monitor and control network traffic flows via firewalls, anti-virus, proxies, and network intrusion detection techniques [14]. Network Intrusion Detection Systems (NIDS) are essential security tools that help increase the security posture of a computer network. NIDS have become necessary, along with firewalls, anti-virus, access control, and other common defense-in-depth strategies towards helping cyber threat operations teams become aware of attacks, security incidents, and potential breaches occurring on their networks [15].

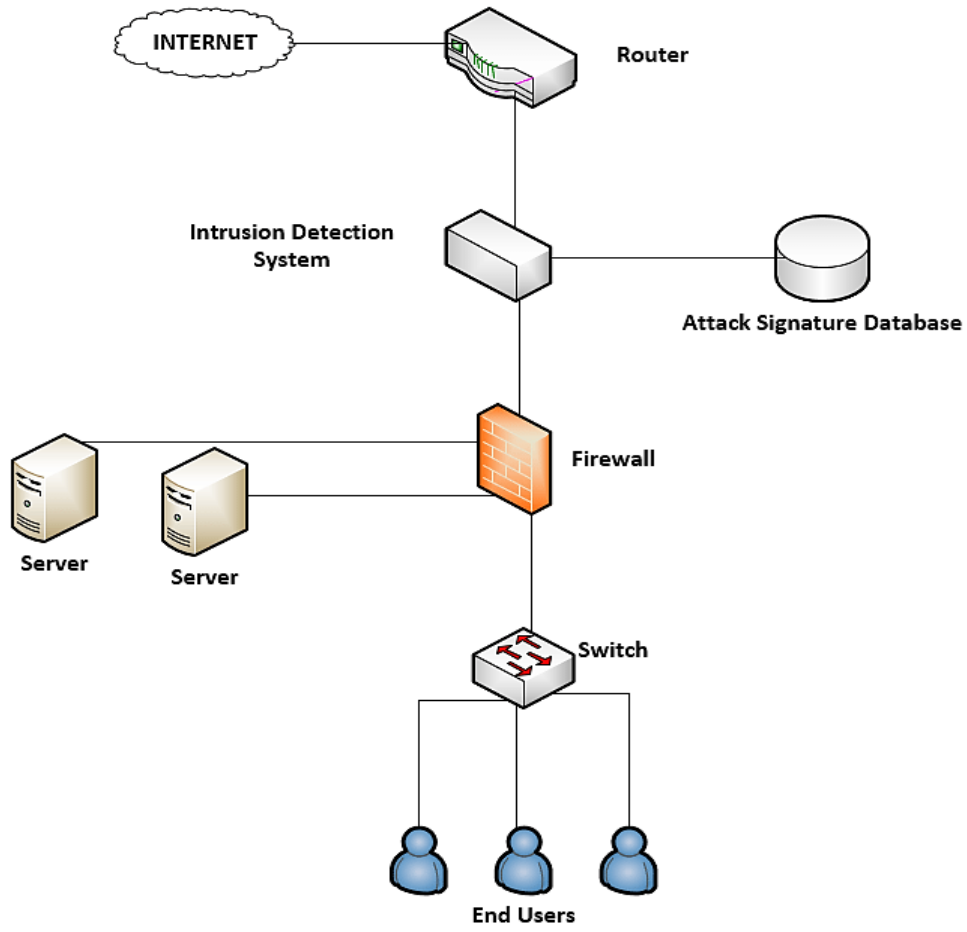

Figure 1.4: The working of intrusion detection system in the presence of firewall and connection of database [15].

1.5 TYPES OF INTRUSION DETECTION SYSTEM

Signature based systems generate alarms when a known misuse or bad activity occurs. These systems use techniques to measure the difference between input events and signatures of known bad intrusions. If the input event shares patterns of similarity with known bad intrusions, then the systems flags these events as malicious. These systems are effective in finding known bad attacks and can flag them with a low false positive rate. The downside to these systems is that they are not able to detect novel attacks [16]. Anomaly based systems trigger alarms when observed events are behaving substantially differently from previously defined known good patterns. The advantage of these systems is that unlike signature-based systems, they are able to detect novel and evolving attacks. An anomaly by definition is anything that deviates from what is considered standard, normal, or expected. Anomalies are rare in occurrence, and deviate from the normal, expected behavior [17]. The goal of an anomaly detection system is to identify any event, or series of events, that fall outside a predefined set of normal behaviors. It is important to note that not all anomalies are necessarily malicious. By definition, anomalies are just deviations from expected normal behavior. Once an event or pattern is deemed to be an anomaly, it can be further labeled as either benign or malicious. Therefore, one of the main challenges in anomaly-based systems is the problem of generating a high rate of false positives, as well as a high rate of false negatives.

1.6 OBJECTIVES OF THESIS

A deep learning approach, known as Deep convolutional neural network (DCNN), is used to detect and classify attack traffic in the case where there isn't any labeled malicious training data. This approach is important because in practice it may be difficult to obtain labeled training data in order to train a supervised deep learning algorithm on malicious and benign traffic. In addition, the nature of the adversary is that they are constantly evolving and attempting new attacks, for which a pattern-based system may not be effective since new attacks may have patterns that are vastly different than what has been seen historically. This approach is considered supervised intrusion based approach, as the learning algorithm will put the traffic into clusters, whereby anomalous activity (e.g. outliers) will stand out from the normal network

traffic. The focus of this research is on advancing NIDSs, by leveraging recent advances in deep learning.

1.7 AIMS OF RESEARCH

This research thesis approaches the challenge of detecting attacks using network intrusion detection in a two-fold manner. First, a fully connected Deep convolutional neural network (DCNN) is used to train a NIDS with supervised learning using labeled benign and malicious network traffic data. Newer benchmark datasets are used which are more representative of modern day network traffic and attacks and do not have drawbacks of previous datasets commonly used in the field. After learning these patterns of malicious and benign by training a fully connected neural network, the system can reliably and effectively detect and classify modern attack traffic with a high degree of accuracy, high rate of recall, and a low rate of false positive rate. This is considered to be a form of pattern-based detection because the system is trained on known good and known bad patterns and taught to detect these patterns in future, unseen network flows using deep learning.

1.8 NOVELTY OF RESEARCH

Firstly, this work will fill the gap of research where a separate model for each network device in the network has been created. The motivation is that training, deploying and maintaining separate models for each device could quickly become burdensome in big networks especially for deep learning systems. The NID Kaggle datasets will be implemented using deep convolutional neural network on Matlab R2019b. This deep convolutional neural network will have similar size and number of layers and their formula for the threshold will be used, albeit it will be modified – a parameter will be added to it, and an optimization is made for this parameter. Some details, such as activation function choice and optimization algorithm choice are this work's original contributions, as they were not described in detail in the referred work. Additionally, this work will employ a feature selection mechanism and will predict the accuracy of feature selection parameters efficiently using deep learning. This work will allow for assessment of deep convolutional neural network method accuracy on smaller feature subsets and for making a fairer comparison on network intrusion traffic detection and classification for attacks with more advanced deep learning method.

2. RELATED WORK

2.1 LITERATURE REVIEW

There are many sub-fields in cybersecurity data analytics. The present paper falls into prescriptive cybersecurity data analytics aim to detect network attacks using autonomous systems, which are then treated properly by the defender; therefore, this sub-field of cybersecurity data analytics may be called reactive cybersecurity data analytics [19].

Intrusion detection can be host-based or network-based.

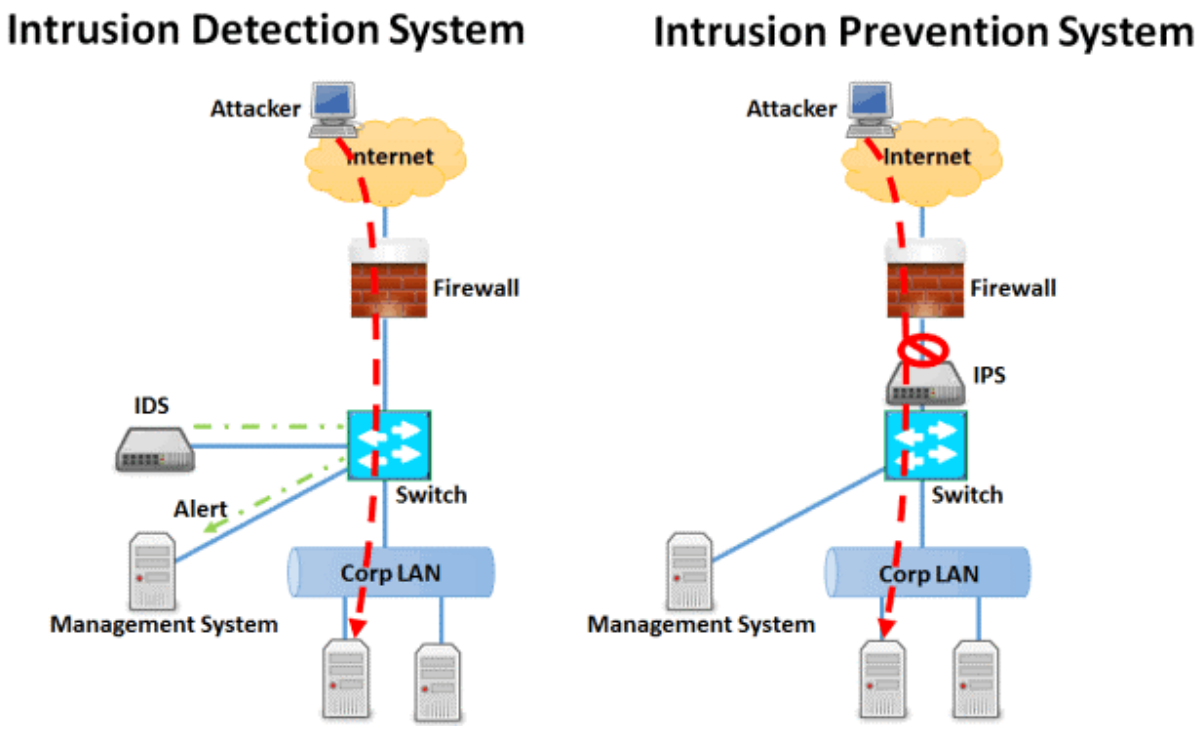

Figure 2.1: A comparison between the intrusion detection system and intrusion prevention system [23].

The study of intrusion detection for cybersecurity has roots in a paper published by Denning in 2017 [20]. They trained their classifier on "normal" traffic only, with the goal of being able to detect anomalous behaviors that may evolve and change over time. They argue that it is often difficult to train a model with anomalous training samples a priori. For their experiments, they used the classic KDD'99 dataset. While this is a common benchmark dataset that has been used in many research papers in the field, it has also been criticized for its lack of relevancy with

current day attacks. Sommer et. al suggest that this dataset may be better suited for only providing additional validation and cross-checking of a novel technique [22].

Other sub-field of cybersecurity data analytics aim to understand, characterize, and predict cybersecurity behaviors and events exhibited by cybersecurity data, including: generating datasets, measuring attack-defense structures in complex networks; measuring the susceptibility of software to attacks measuring capabilities of defense tools, quantifying capabilities of attacks, malware detection and adversarial malware detection, predicting or forecasting the cybersecurity situation based on the monitored cybersecurity data streams.

2.2 RELEVENT ALGORITHMS

2.2.1 Artificial Neural Network

While deep learning has gained popularity in recent years, it has been around for a long time and its origin dates back to the 1940s [24]. Through its history, it has gone by different names such as "cybernetics" in the 1940-1960 timeframe, and "connectionism" in the 1980-1990s, to what it is known by today as "deep learning" with renewed interested starting back up in 2006. Some of the early algorithms in deep learning were biologically inspired by computational models of the human brain, thereby popularizing algorithms with names such as artificial neural networks.

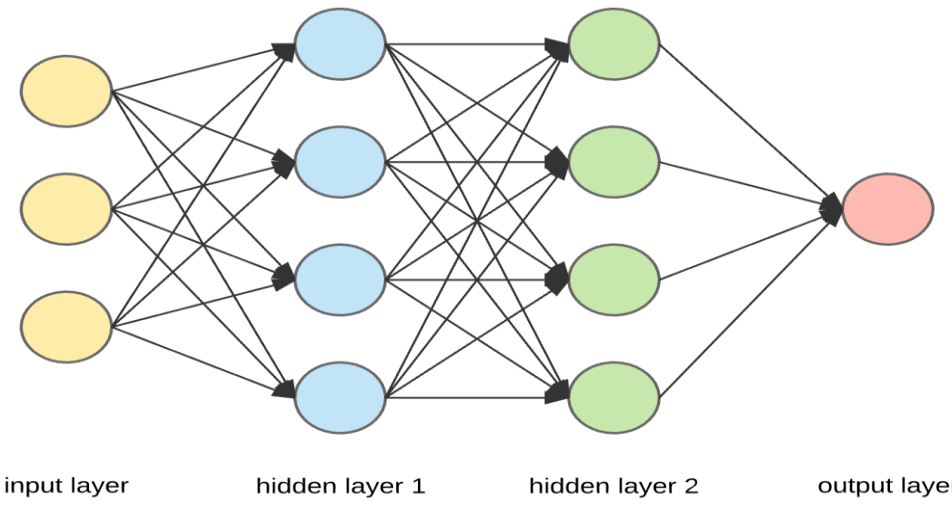

Figure 2.2: Artificial neural network algorithms based on classification, regression and clustering containing different layers [26].

(ANNs) and by describing computational nodes as neurons. While the neuroscientific perspective is considered an important source of inspiration for deep learning, it is no longer the primary basis for the field — there simply does not yet exist a full understanding of the inner workings and algorithms run by the brain. This is an active and ongoing area of research being conducted within the field of "computational neuroscience." While models of the brain such as the perceptron and neuron have influenced the architecture and direction of deep learning over the years, it is by no means a rigid guide. Modern deep learning instead is based more on the principle of multiple levels of composition [25].

2.2.2 Autoencoder

Autoencoders and Restricted Boltzmann Machines (RBMs) are two types of neural network architectures that are considered building blocks of larger deep networks [27]. Often, these types of networks are used in a pretraining phase used to extract features and pretrain weight parameters for a follow-on network(s). They are considered unsupervised because they do not use labels (ground-truth) as part of their training. A common use case for using unsupervised pretraining is when there exists a lot of unlabeled data, along with a relatively smaller set of labeled training data [28]. This is a common scenario for enterprise network security use cases, as often times there is a subset of labeled training data that has been reviewed, processed, and labeled by a human analyst, yet there is still an enormous amount of unlabeled data for which there is not enough manpower to review. The downside to having this pretraining step is the extra amount of overhead in terms of network tuning and added training time [29]. For this reason, Autoencoders can be powerful when used in anomaly detection systems.

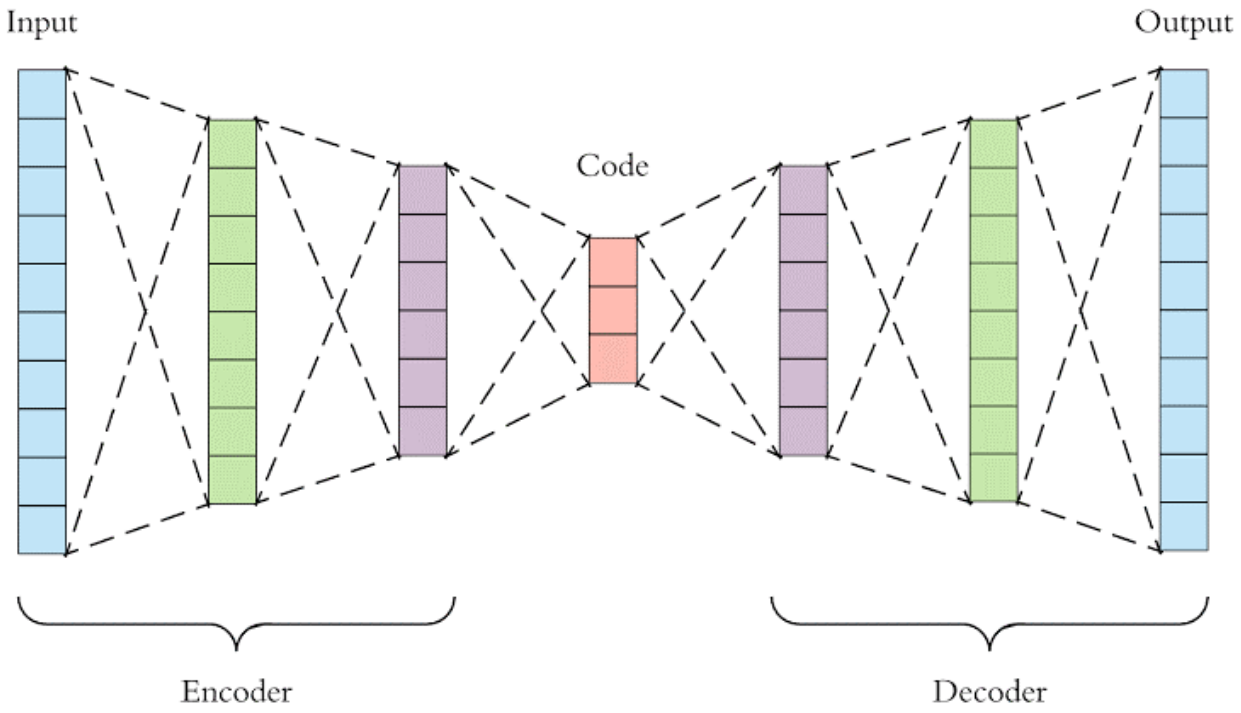

Figure 2.3: Autoencoder algorithms based on classification, regression and clustering for encoding and decoding [30].

2.3 MACHINE LEARNING

American pioneer in the field of Artificial Intelligence, Arthur Samuel, is responsible for introducing the term machine learning in 1959 [33]. Machine learning works on prediction using computer and data and also associated with computational statistics. ML is a sub-category of AI. ML contains different theories, methods and application domains as it has a strong connection with mathematical optimization.

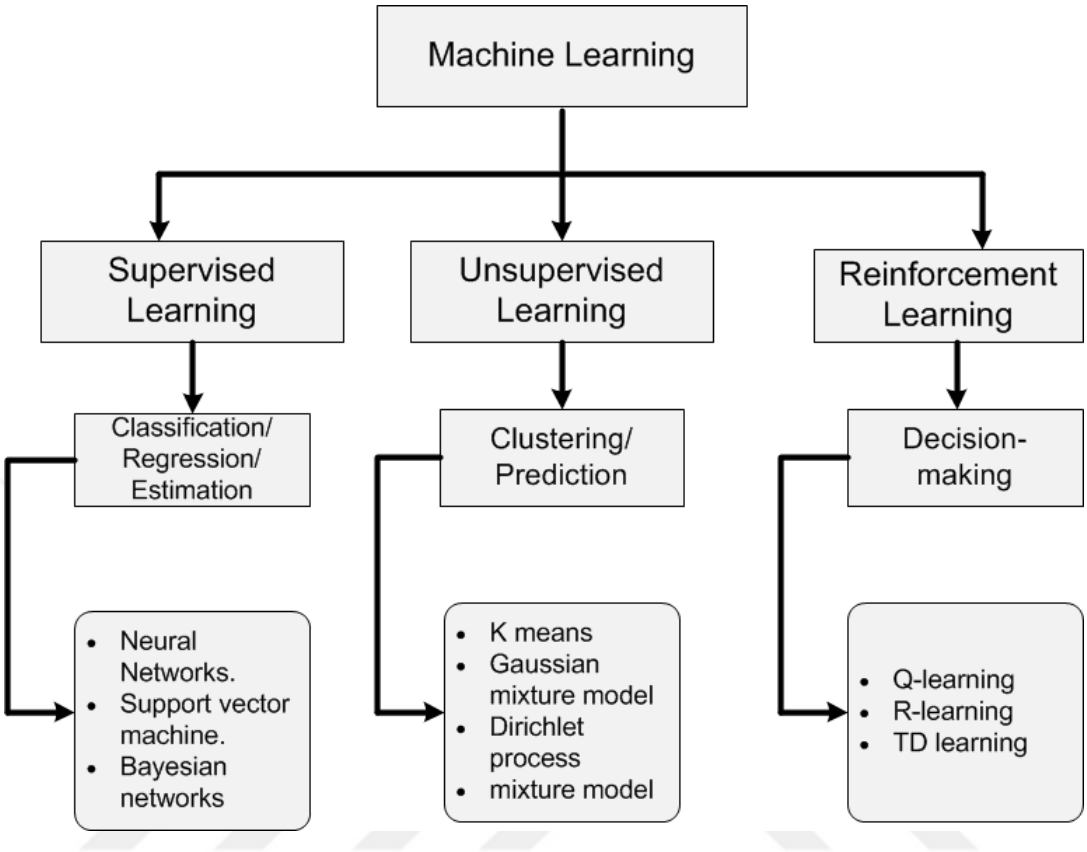

Figure 2.4: Taxonomy of machine learning with three approaches of machine learning [33].

Machine Learning primarily includes the following four steps:

Choose the basis attributes or features or values for prediction. Choose the appropriate machine-learning algorithm according to the dataset. For example, among classification algorithm or regression algorithm, which one to choose, high complexity or faster one.

- Training and evaluating model performance based on different algorithms.
- Classify or predict the unknown data, training model is used.

2.3.1 Supervised Learning

Algorithms for supervised learning are used to build a model from a set of data that has both the desirable inputs and outputs. The data used is training data and contains training examples that has single or multiple inputs and an output [26, 28]. The method used is known as supervisory signal.

When dealing with problems regarding supervised signal, first we begin with a set of training examples with correct labels associated with it. An example is when we train a supervised learning algorithm which takes thousands of pictures of hand-written digits with the correct labels containing the correct number for each image it represents [28, 29]. This enables the algorithm to classify handwritten digits and also learn the relationship of the images and their respective numbers and also use this relationship to classify new images without labels that has not been shown to the machine before.

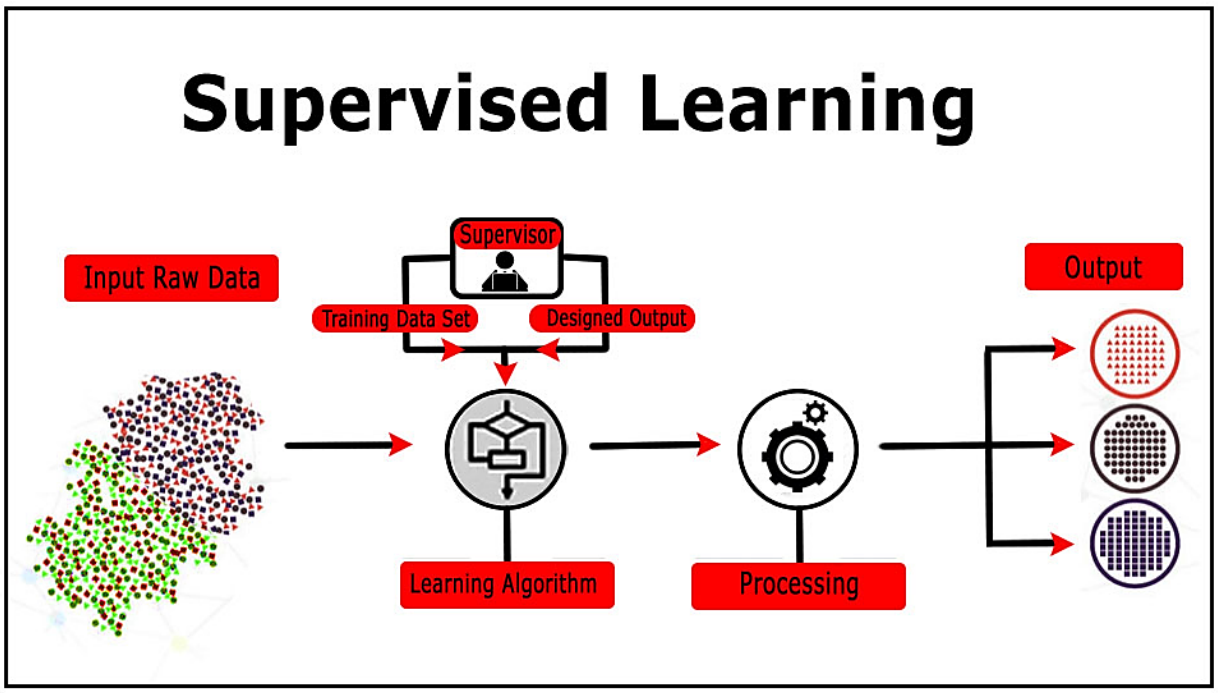

Figure 2.5: The training of supervised based algorithm from input to output [27].

Classification and regression is a part of supervised learning [20]. Algorithms for classifications are used in cases where outputs are restricted to sets of values which are limited. Algorithms for regression are used in cases when outputs may have values which are numerical within a given range.

2.3.2 Unsupervised Learning

Algorithms for unsupervised learning are given sets of data that contain inputs only and then the algorithm finds clustering of data points or finds structure in data [29]. Therefore, the algorithms

learns from non-labeled, non-categorized or non-classified test data. Unsupervised learning algorithms detect commonalities in the data which is based on the absence or presence of the commonalities in each new piece of data instead of a feedback approach.

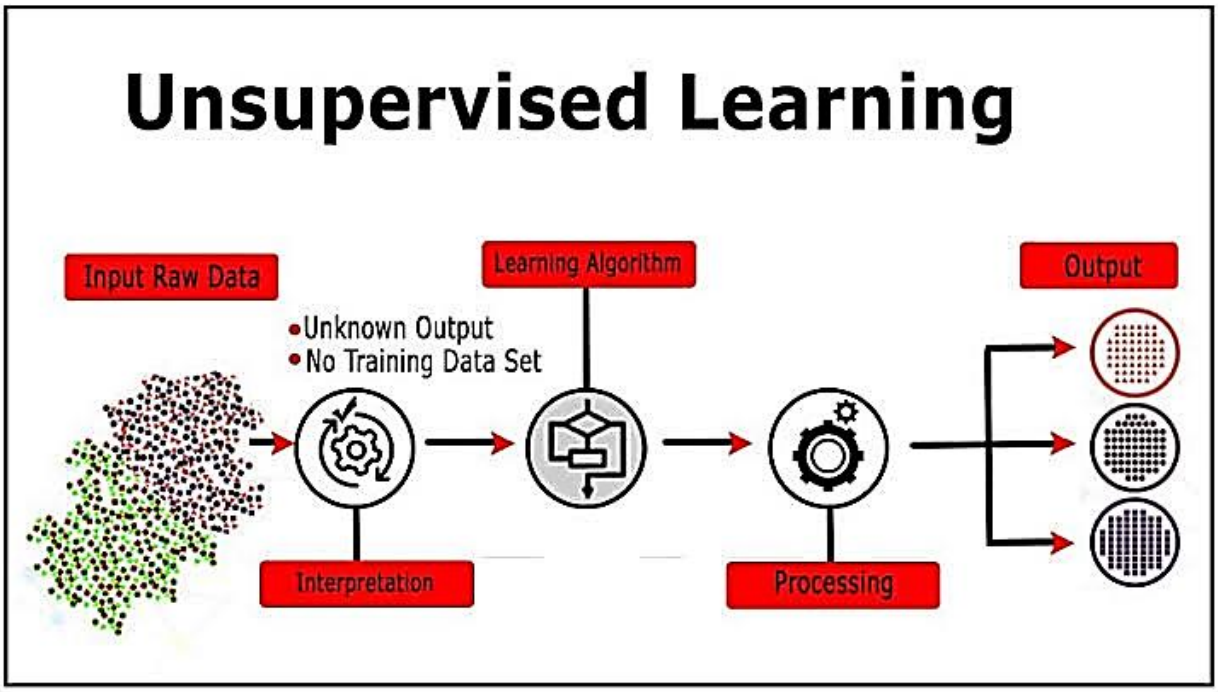

Figure 2.6: The training of unsupervised based algorithm from input to output[27].

Grouping, summarizing and finding underlying structure of unlabeled data in a useful manner is the function of unsupervised learning.

2.3.3 Reinforcement Learning

Reinforcement learning techniques can be leveraged in order to help separate the signal from the noise in a given dataset. The hypothesis is that by reducing the dimensions and removing noise from the signal, a reinforcement learning classifier can perform better, as it will mainly be learning from the signal, without additional noise getting in the way. It explores the use of reinforcement learning techniques, namely autoencoders, to perform dimensionality reduction and train a neural network to reconstruct its inputs, instead of predicting a class label as in reinforcement learning. By learning the representation of the input data for normal network flows, a reconstruction error is calculated on never-before-seen test inputs, whereby higher

reconstruction errors above a set threshold are flagged as anomalous. In this way, reinforcement learning, and specifically autoencoders, can be a powerful engine for an intusion detection system.

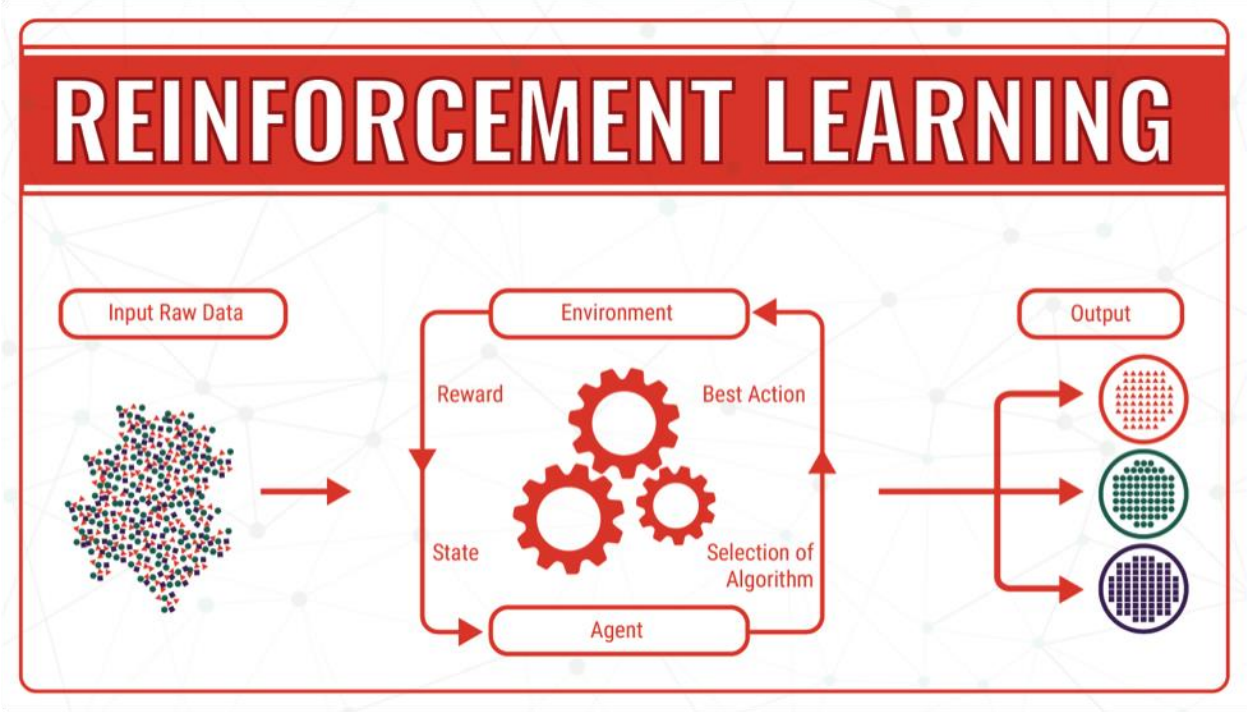

Figure 2.7: The training of reinforcement based algorithm from input to output [27].

2.4 NETWORK TRAFFIC AND PACKET CAPTURING

Network traffic data is often collected at network switches or routers in the form of raw packet capture (PCAP). PCAP data contains the full TCP/IP packet data transmitted or received on a given network device. While full packet capture information can be helpful in certain scenarios, it does bring with it a high cost in terms of space. An alternative (or complement) to PCAP data is Net-Flow, which serves to summarize the PCAP data in terms of higher-level network flows. NetFlow data offers the advantages of more breadth and lower disk space requirements, whereas PCAP data has more depth, but is more expensive in terms of disk space. NetFlow records typically consist of source/destination IP, source/destination port, source interface, protocol, and number of bytes transmitted. It can have many more fields extracted from the PCAP based on the configuration of software used for converting PCAP to NetFlow records.

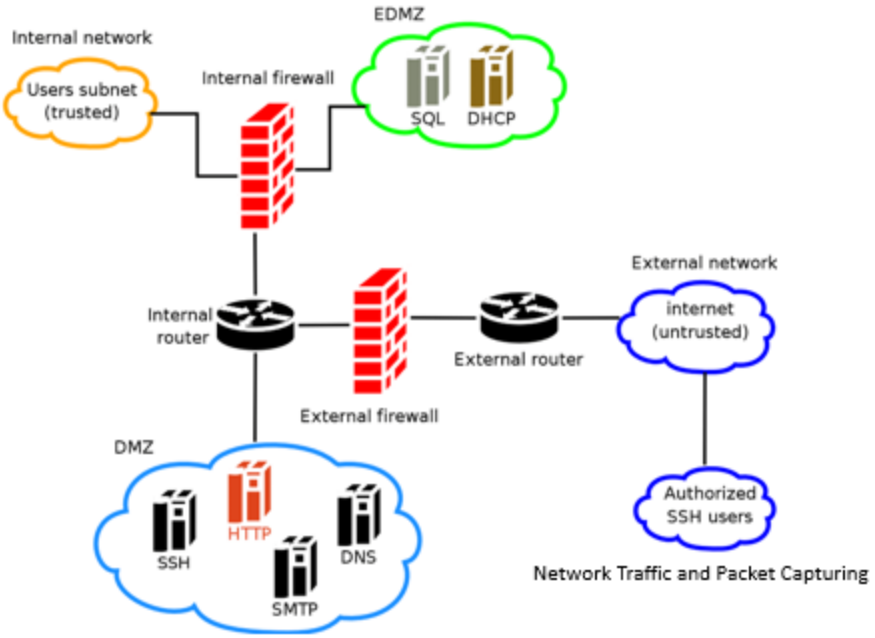

Figure 2.8: The network traffic and packet capturing including the DMZ zones for intrusion detection [29].

The machine learning algorithms used in this work utilize NetFlow records as input. Therefore, if there is any PCAP input data, it must first be converted to NetFlow. In practice, running an algorithm that takes NetFlow as input is ideal — the NetFlow protocol was created by Cisco, and is commonly used by many network switches, routers, and firewalls to export raw traffic data into flow data for further analysis [35]. An emerging IETF standard called IPFIX, short for IP Flow Information Export, is similar to the NetFlow standard, but allows for additional information that is normally sent to syslog, as well as variable length fields for collecting information such as URLs, messages, HTTP hosts, and more [32, 36]. For the purposes of this work, the summarized traffic records will simply be referred to as 'flow data'.

In order to convert raw PCAP data to flow data, there are a number of tools available. One tool is called YAF, short for Yet Another Flowmeter. YAF was created by Computer Emergency Response Taskforce (CERT) and is a part of their Network Situational Awareness (NetSA) Security Suite of tools used for monitoring large-scale networks. YAF processes packet data from PCAP into bidirectional flows, then exports the flows to IPFIX-based file format.

2.5 FLAWS IN NETWORK DEVICES

Flaws in software is found very frequently because of the fact that network devices lack the capability to be patched or fixed. This makes them prone to security risks like DDoS attacks, one of the most common issues, which happens by setting the device's password to its default password which are easily crackable by hackers [37]. Ransomware and Malware both relies on encryption to lock out the users devices completely and steals the user's data. Privacy remains among the largest issues of network as data is being transmitted, stored and processed and being harassed by large companies [38]. This very data is then being sold to various other companies which is violating our rights for privacy and data security. Home Invasion, one of the scariest threats that network can possess as most of the homes and offices rely on automatic processes that leads to the use of IoT devices in these places. IP address of these devices are exposed to hackers who can use it to pinpoint the location and gain access to our personal data just to sell them to underground sites which are the hives of criminal activities.

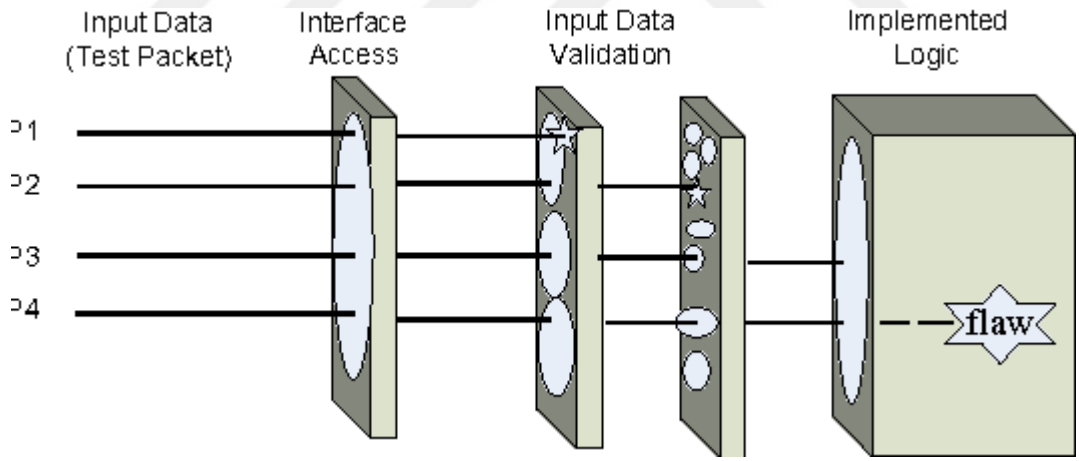

Figure 2.9: An example of packet flaw in a host based network [38].

3. METHODOLOGY

In this chapter a series of steps will be performed using a deep convolutional neural network which is fully connected feedforward neural network to perform classification on two different network intrusion detection benchmark datasets.

In this chapter, the following contributions are provided:

1. Application of a fully connected deep convolutional neural network to network intrusion detection. Through experimentation, it is determined the set of hyperparameters and network configuration that produces optimal results in terms of evaluation metrics. In addition, comparison is performed between including and excluding IP address as features.

2. Validation of the approach by evaluating results using two benchmark intrusion detection datasets. The two benchmark datasets used in this research are recent and contain modern-day attacks. Using these datasets with the deep convolutional neural network approach in this chapter shows that this technique can be applied to modern-day enterprise networks.

3. Evaluation and comparison of this deep learning approach for network intrusion against other previous techniques. There are a number of studies that have now been performed using the two intrusion detection benchmark datasets used in this thesis. Therefore, the results achieved in this work are compared with other approaches in the field.

3.1 PRE-PROCESSING STEPS

For this research study, experiments will be performed with NID KAGGLE benchmark dataset [40]. Each of these datasets contain ground truth labels which is necessary for carrying out deep supervised learning. While the version of the NID KAGGLE dataset used in this work has binary class labels for benign and malicious and the NID KAGGLE dataset also contains multinomial class labels for each of the attack types carried out in the flow records. Dataset is used within a common deep learning workflow to apply the deep convolutional neural network approach to network intrusion detection. The results of each of the experiments will be analyzed and evaluated, then the effectiveness of this approach against previous approaches in the field will be compared.

3.1.1 Splitting the Dataset

Following best practices, the dataset is separated into three separate training, validation, and test sets. First, the dataset is separated out with 33% of the data put into a test set, and 67% into a training set. In addition, since this dataset is highly imbalanced, it is important to ensure there is the same proportion of malicious flows in each training and test sample as there are in the dataset as a whole. In order to accomplish this, the dataset is stratified to ensure the distribution of benign and malicious traffic is equivalent in both training and test data sets. In addition to the standard training and test split, a separate hold out validation set is used during the training iterations. This is a standard methodology used when there is plenty of data available. Therefore, the training split is further separated out into a validation set according to the same parameters as the original training and test split, including stratification. This results in 67% of the training dataset set aside for training, and 33% of the training dataset used for validation during the training iterations.

3.1.2 Data Cleaning

Once the data is in the form of split data, a process is performed to further clean, transform, and prepare the data for use in the deep learning algorithm which is deep convolutional neural network in our study. This step includes common tasks such as making sure there are no erroneous characters in the data, removing fields that contain all zero or null values, and removing or modifying NaN (not a number) values. Often, a large amount of time can be spent in this stage to clean and prepare the data. This is due not only to the various formats and locations of target data, but also to ensure the accurateness and effectiveness of the model that is being trained on this data.

During this stage, it is common practice to also normalize or scale the continuous values amongst all the features, so that the deep learning algorithm trains on data that is all within the same feature space. There are two main types of normalization that are commonly used within the machine learning pipeline: Standardization (or Z-score normalization) and Min-Max scaling.

3.1.3 Feature Selection

The NID KAGGLE dataset, the high cardinality features are reduced in dimension using entity embedding's, following the first rule of thumb. In effect, this borrows from the technique used in word embedding, and performs IP embedding and Port embedding to utilize these features within a neural network [41]. Specifically, the high cardinality categorical variables are each mapped to an integer index.

This process works by creating a new variable for every unique value possible for the original categorical feature [42]. An example of how this works is if there is a categorical feature named 'attack' with three different possible values, such as 'worms', 'trojans', 'spyware'. Since the deep learning algorithm only works with continuous floating point values, feature encoding can encode this 'attack' feature.

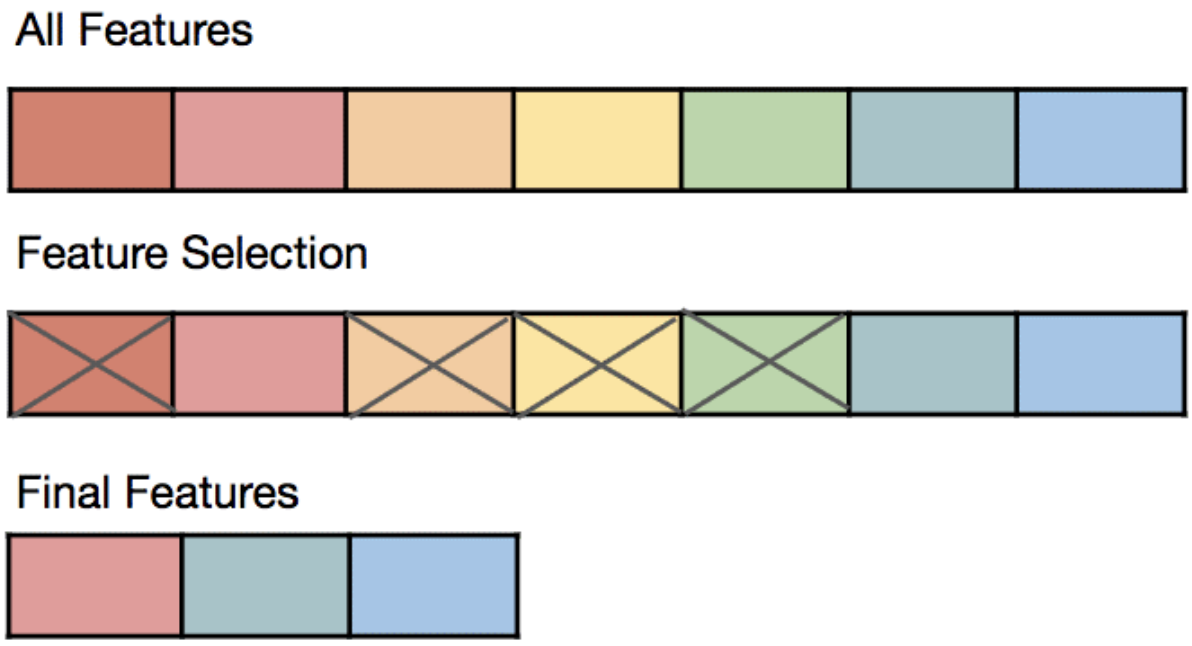

Figure 3.1: The process of feature selection from given features to refining till the final features are attained [43].

Using feature selection encoding in the context of deep convolutional neural networks has solved two main problems:

- Feature selection encoding of high cardinality features can be efficient and does not require a large amount of computational power to complete.
- Feature selection encoding ignores the relationships between categorical features, and treats them completely independent of one another.

3.2 DATASET DESCRIPTION

A primary and ongoing challenge in the field of network intrusion detection is the lack of publicly available, labeled datasets that can be used for effective testing, evaluation, and comparison of techniques. Often times, the most useful datasets for network intrusion detection are those containing captures of real network environments [44]. These datasets are not easily shared with the public, as they contain details of an organization's network topology, and more importantly sensitive information about the traffic activity of the users on the respective network. Furthermore, the effort required to create a labeled dataset from the raw network traces is an immense undertaking. As a consequence, researchers often resort to suboptimal datasets, or datasets that cannot be shared amongst the research community. Granted, publicly labeled datasets are available, such as CAIDA [45], DARPA/Lincoln Labs packet traces [46], KDD '99 Dataset [47]; however, these datasets are mostly anonymized and do not contain valuable payload information, making them less useful for research purposes. While these datasets have proven useful, there are some arguments as to the validity of using them in present day research — they may be better suited for the purposes of providing additional validation and cross-checking of a novel technique.

3.3 FULLY-CONNECTED DEEP CONVOLUTIONAL NEURAL NETWORK

A neural network is considered deep if it contains more than three layers, including input and output layers. Therefore, any network with at least two hidden layers is considered a deep convolutional neural network. An example of standard deep learning representations. The former shows a deep, fully connected neural network, as each of the neurons in the input layer are connected to every other neuron at each successive layer.

The basic technical approach of deep learning for neural networks has been around for decades, so why has this area been gaining so much attention in recent years? The main reason for this is due to an increase in scale of both amounts of data and computational power available. A larger amount of available data, combined with larger neural networks has led to an increase in performance of deep convolutional neural network learning algorithms, specifically in the context of supervised learning [51].

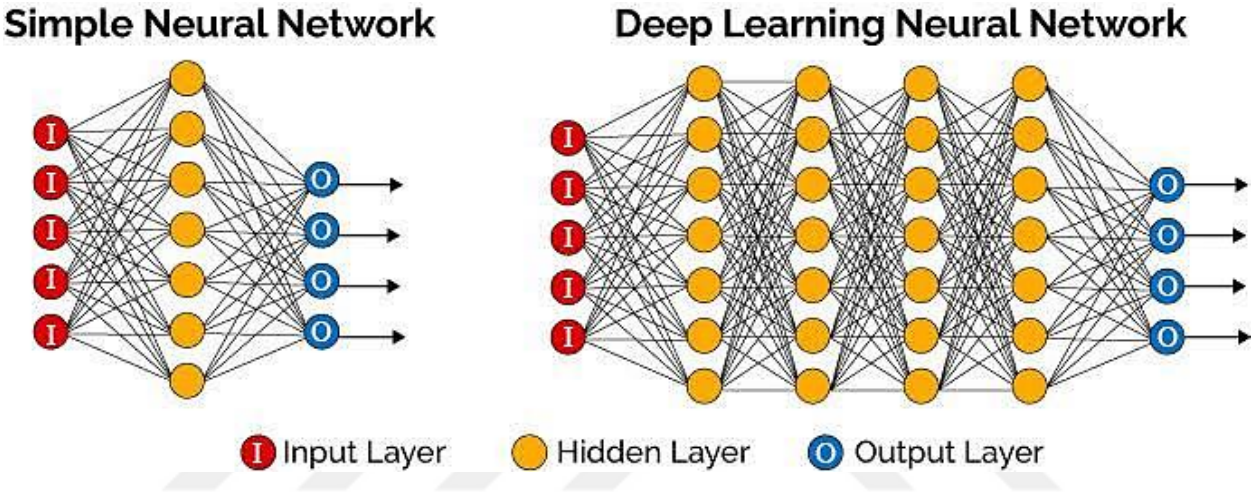

Figure 3.2: A comparison between the simple neural network and the novel deep convolutional neural network with number of hidden layers.

3.4 DEEP CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

Deep convolutional neural network architecture and hyperparameters refer to the parameters that are set before the learning process begins. These DCNN hyperparameters are tuned based on a feedback loop of the model's performance on the validation set. Hyperparameters fall into the following main categories:

- Layer size, number of units per layer
- Magnitude (momentum, learning rate)
- Regularization (dropout, L1, L2)
- Activation functions (sigmoid, relu, tanh, etc.)
- Weight initialization

- Loss functions
- Number of epochs to train, and batch size per epoch
- Data normalization scheme

Throughout the training and validation cycle, these hyperparameters are tuned in order to achieve highest and best performance for the machine learning model. The most effective neural network architecture and hyperparameters used in this work.

The configuration for the deep convolutional neural network as shown in Figure 3.2 gave best results in the experiments. The first hidden layer for the continuous input features is much larger in this case at 128 neurons, than the number of inputs at 7. Smaller numbers of neurons were also tested and found to achieve comparable results. Therefore, to keep architectures consistent, this same general configuration is also used in the study with NID KAGGLE, which has many more continuous input features.

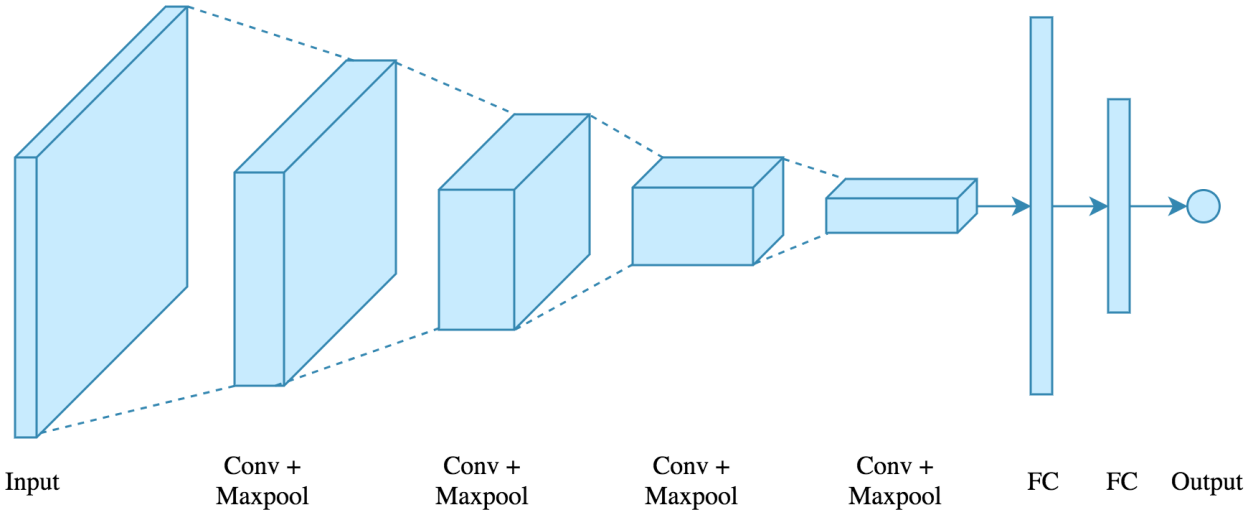

Figure 3.3: An architectural diagram of deep convolutional neural network from input leading to output different fully-connected layers.

It should also be noted that it is recommended future work to perform an in-depth study that performs further hyperparameter optimization to determine the neural network architecture and combination of features that performs best for each of the given dataset.

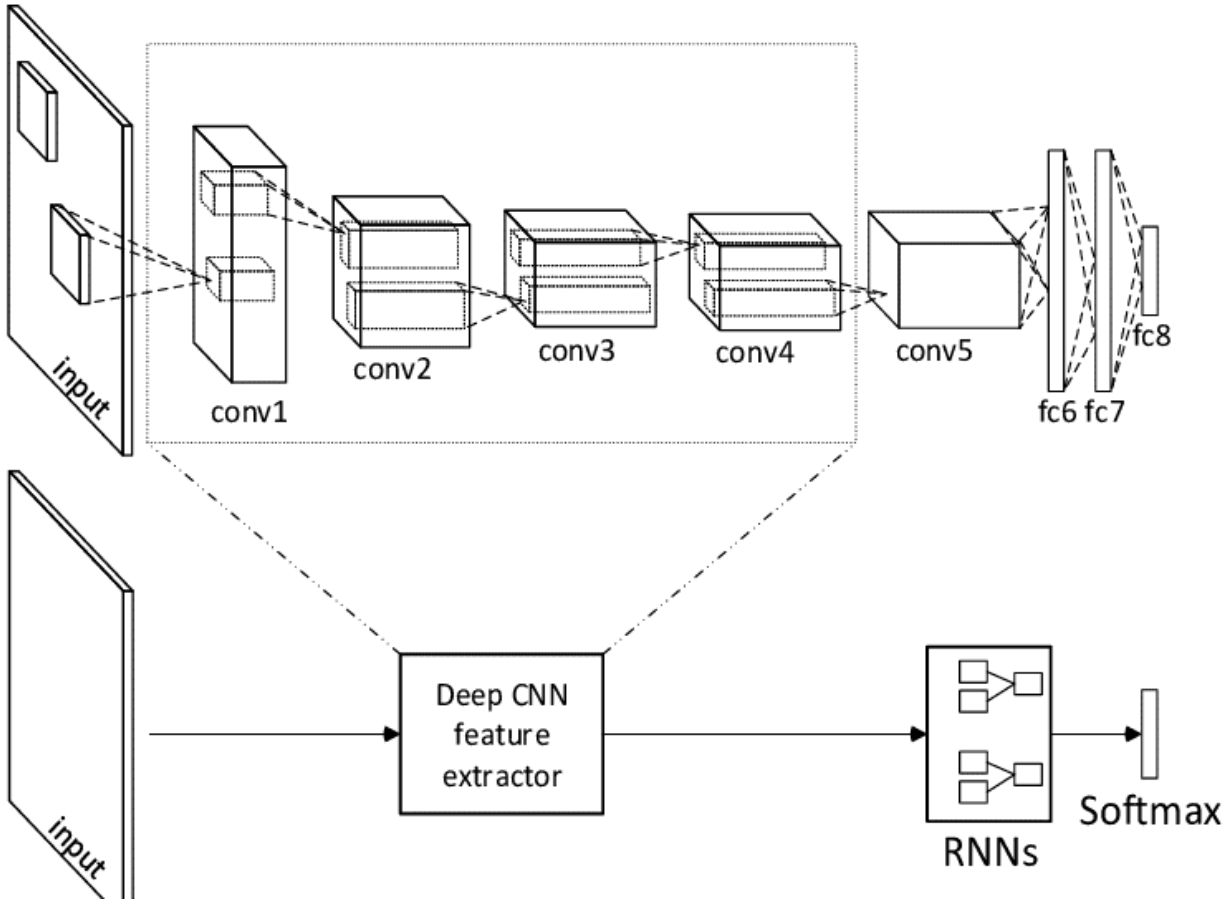

Figure 3.4: An detailed architectural diagram of deep convolutional neural network from input leading to output different fully-connected layers with deep feature extractor and SoftMax.

3.5 TYPES OF ATTACKS

An important component of this dataset is the degree to which it represents real network traffic that would be naturally generated by a live network, commonly referred to as background traffic. In order to accomplish this, a profile-based agent on their previously proposed profile system in [52]. Similar to the background traffic generation in NID KAGGLE, this system profiles normal human interactions on the network in order to be used for benign background traffic generation at a later time. Using the DCNN system for this dataset, 25 users' behavior was profiled based on their use of HTTP, HTTPS, FTP, SSH, and email (SMTP). Deep learning and statistical analysis techniques to obtain distributions of packets, packet sizes, protocol use, etc. These generated β-

profiles are then used by an agent written in Java in order to create realistic background traffic on the Victim-Network based on the real 25 users' prior behavior.

The NID KAGGLE dataset contains over 2.8 Million flows. This dataset also aims to cover an up-to-date and diverse set of attacks that are seen in modern day networks. Therefore, this dataset contains the following seven types of attack profiles and scenarios:

3.5.1 Brute Force Attack

A common attack where an attacker repeatedly attempts to guess a password by 'brute-forcing' a large number of attempts, one after another until they succeed with a correct username/password combination. Not only is this technique used for credentials, but it can also be used to 'brute-force' a web application or server, trying to find hidden pages or directories.

3.5.2 Heartbleed Attack

This attack exploits a vulnerability in the OpenSSL implementation of the TLS protocol. It allows the attacker to send a heartbeat payload (intended to be used to check that a server or service is still active and 'alive'), which causes the OpenSSL library to return more data to the requestor (attacker) than was designed. This enables an attacker to steal sensitive information such as private key material, which could later be used to decrypt confidential information.

3.5.3 Botnet

A botnet consists of a large number of 'zombie' hosts that have been infected with a piece of malware, whereby a Command and Control (C&C) server can send instructions to the bots to perform a specific command or series of commands.

3.5.4 DoS Attack

A Denial of Service attack is one in which the attacker intends to hinder the availability piece of the CIA (confidentiality, integrity, availability) triangle, causing the service to go down often by flooding the system with an abundance of requests beyond which the system can respond to.

3.5.5 DDoS Attack

A Distributed Denial of Service attack is similar to a DoS attack, with the only difference being that the attack is now carried out by multiple, distributed hosts (often facilitated via a botnet). These are more difficult to contain, as the source of the attack is not concentrated.

3.5.6 Web Attack

Web attacks can take a variety forms, and they are always evolving. In this dataset, some of the most popular forms of web attacks are performed, including SQL Injection, Cross-Site Scripting (XSS), and Brute-force password guessing. SQL Injection is a type of fuzzing attack where the attacker injects (or appends) additional string values into a form field that if not properly checked for by the web application, would trigger the database to perform commands it was not intended to perform. This is commonly a scenario in which many web applications leak sensitive data inadvertently. Cross Site Scripting (XSS) occurs when a web application contains form fields that don't properly sanitize the input, allowing an attacker to run malicious scripts on the server. Brute-force password guessing is similar to Brute-force SSH attacks; however they are run over the HTTP/S protocol against a web application/server.

3.5.7 Infiltration Attack

This is a dangerous attack where an external bad actor is able to gain unauthorized access to the internal network. This is often accomplished via a social engineering attack where the attacker will send a phishing email to a victim, and convince the victim to click a link that leads to a malicious website that launches an exploit, or has the victim open a malicious attachment that contains a zero-day attack that allows the bad actor to compromise the victim's computer via establishing a back door. Through this backdoor, the attacker can run commands remotely, which can be anything from performing reconnaissance on the topology of the network, looking for vulnerable services to perform lateral movement, or anything else they desire.

1. Application Layer (Host based)
2. Network Layer (Network Based)

3.6 TARINING WITH DEEP CONVOLUTIONAL NEURAL NETWORK

Determining the value of training plays a significant role in determining the efficiency of the model. Value of training determines how well the data can be segregated to generalize the result of the DCNN algorithm. For our dataset, we first found the value of training was 80% which is best suited our case. We initially classified with DCNN. However, this had no real logic behind it. As a result, we decided to find the optimal value that will give better and correct accuracy. It works by training our model on a range parameter and then finds out the best result based on comparison. In our case, we provided a range of values of neighbors from 1 to 25 and checked which value of neighbor provided the most accurate and reliable output. It is seen that best suited our data set.

Table 3.1: Training network based feature selection and embedding categorical variables using DCNN for - NID KAGGLE Dataset

Feature	Possible Values	Embedded Dimensions
Source IP	2,478	7
Destination IP	34,552	13
Source Port	64,482	16
Destination Port	24,238	12

After determining the value percentage for training, we load the data set. Duplicate rows with same values for all the four features are dropped to avoid training the same data over and over again. It also prevents testing and training with same dataset, avoiding accuracy levels that are unrealistic. Next, we split the data into train and test. 80% of the sample is train set and 20%of the data is test set.

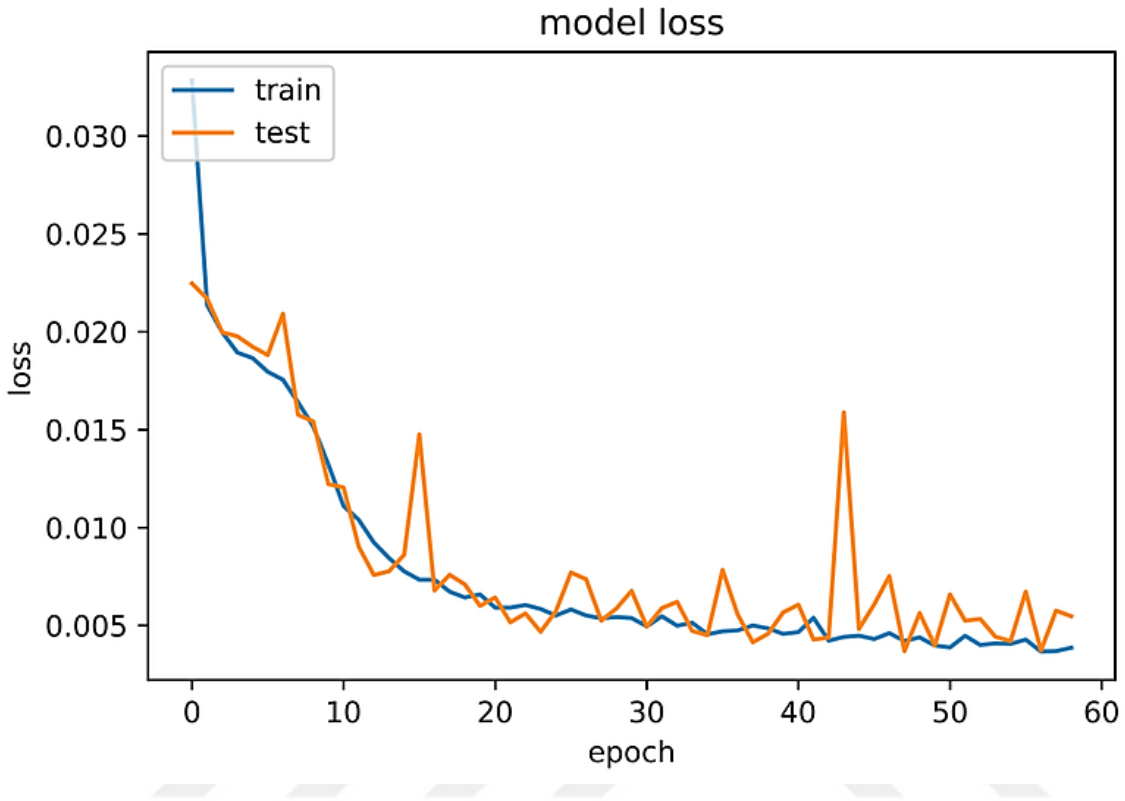

Figure 3.5: The training and testing parameters for deep convolutional neural network with respect to model loss and 60-epochs training.

The 80% data is trained with DCNN classifier. We then test the 20% data. It is done by calculating the epoch instances and model loss of the clusters or classes formed. The distances are then sorted in ascending order. The top rows are then selected and the most frequent class of these rows are then returned. Finally, we calculated the accuracy and it was very high.

Table 3.2: Feature selection and embedded categorical features - source IP address example

Index	**Latent Factors**							
1	−0.041	0.028	−0.013	0.038	−0.037	−0.035	−0.026	−0.033
2	0.004	−0.024	−0.050	−0.032	0.005	0.021	0.009	−0.044
3	−0.017	0.015	0.042	0.008	0.023	0.047	−0.009	0.036
4	0.041	0.007	0.029	−0.010	−0.013	0.025	0.035	0.015
…	…	…	…	…	…	…	…	…
n	0.043	−0.049	−0.009	0.024	0.035	0.003	−0.026	−0.041

3.7 NETWORK INTRUSION DETECTION (NIDS) WORKFLOW

Network intrusion detection in this work use the given deep convolutional neural network architecture using flow records with labels, thus having a total of 14 primary features available for use in learning/training. All of the 14 primary features in the NID KAGGLE dataset. Five of the features are categorical, and the remaining 78 features are continuous. The five categorical features are 'SourceIP', 'DestinationIP', 'SourcePort', 'DestinationPort', and 'Protocol'. The NID KAGGLE dataset as expected, as they are common and necessary elements in network flow records.

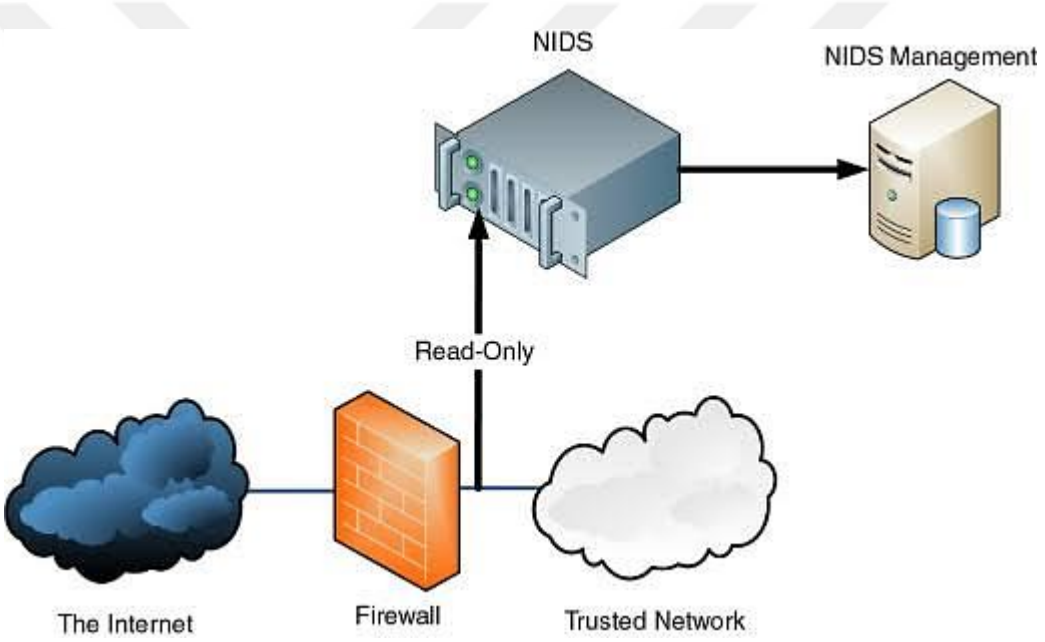

Figure 3.6: The pictorial representation for the detection of attacks in the network using deep convolutional neural network.

There are 17, 0002 unique values for 'SourceIP', and 19, 112 unique values for 'DestinationIP'. For 'SourcePort', there are a total of 64, 638 unique values, and for 'DestinationPort' there are a total of 53, 791 unique values. For the last categorical feature 'Protocol' there are three unique values of '6.0', '0.0', and '17.0' which map to internet protocol types. A value of '6.0' indicates TCP traffic, a value of '17.0' indicates UDP traffic, and while '0.0' translates to IPv6 'Hop-by-Hop Option', it is assumed that a value of '0.0' is undefined since this dataset is dealing with IPv4 traffic. In addition, the number of flows that are of UDP type is 99, 476, and the number of flows for TCP traffic is 1, 826, 704, leaving only 1, 696 flows with protocol label '0.0'.

Therefore, with these five categorical features, the NID KAGGLE dataset can be expanded to a maximum of 17, 002 + 19, 112 + 64, 638 + 53, 791 + 3 = 154,546 features. The training data and feature selection is used to build DCNN architecture and for predicting the intrusion in network. To implement the algorithms and test the system on MATLAB R2019b.

3.8 EVALUATION MATRICS

The primary goal of a classification algorithm in the context of network intrusion detection is to achieve the highest level of accuracy with the lowest number of false positives. In addition, the True Positive Rate (TPR) (also referred to as Detection Rate, Recall, or Sensitivity) is an important metric for network intrusion detection as it indicates the number of malicious examples that are correctly identified. For the task of intrusion detection, we can apply a two-class confusion matrix to evaluate our results as in Table 3.4. The results and evaluation of experiments with the ISCX IDS 2017 dataset are described. The experiments show that by using deep fully connected feedforward neural networks for classification of network traffic flows as either benign or malicious, and using the embedding technique for the high cardinality categorical features, high performance can be achieved in terms of evaluation metrics. A variety of hyperparameters were iterated with along with different combinations of features in order to arrive at the best results.

Table 3.3: Confusion matrix classes for predicting normal or predicting attack [15]

	Predicted normal	**Predicted attack**
Actual normal	True Negative (TN)	False Positive (FP)
Actual attack	False Negative (FN)	True Positive (TP)

Accuracy can then be defined as (1).

$$ACC = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{3.1}$$

And precision as (2).

$$PR = \frac{TP}{(TP+FP)} \tag{3.2}$$

False positive rate will be defined as (3).

$$FPR = \frac{FP}{(FP+TN)} \tag{3.3}$$

Similarly, accuracy can be computed for more than 2 classes. Also, for our machine learning models, we can also measure how long does the training take in seconds to achieve good accuracy and how complex the model is, meaning how many parameters does it contain and how much does it weight when saved to disk.

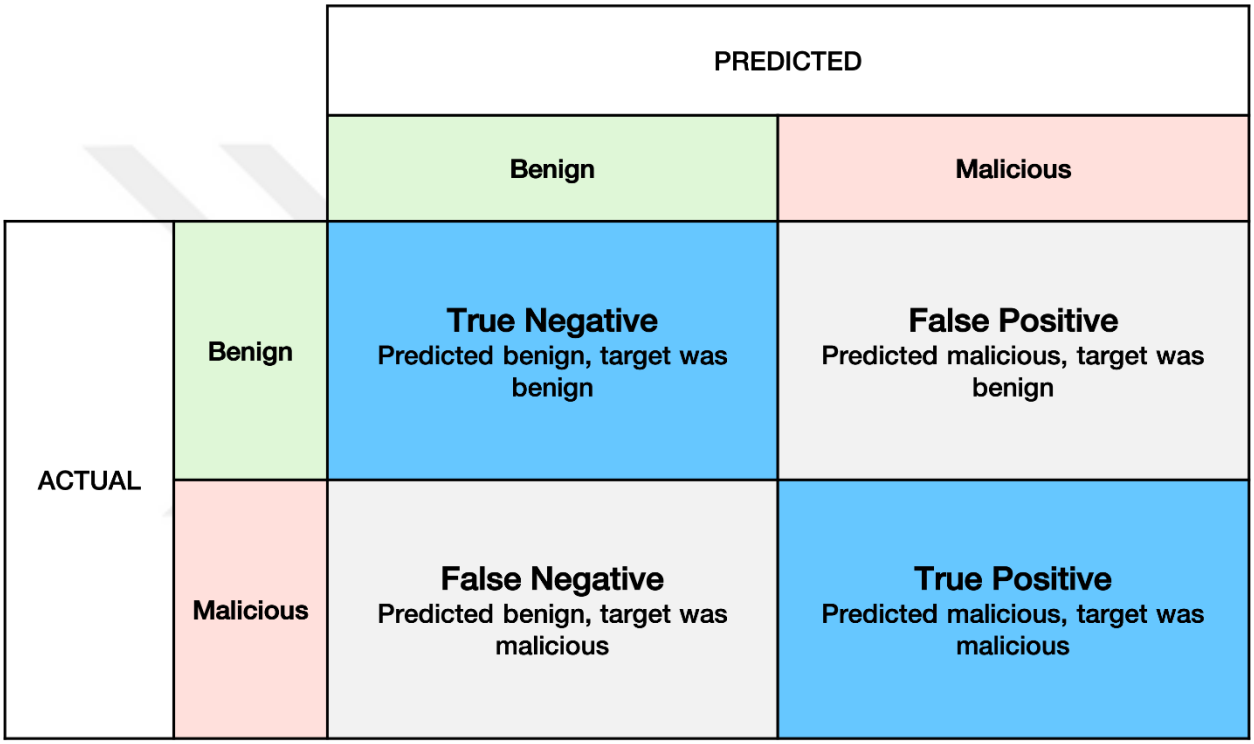

Figure 3.7: The confusion matrix and evaluation of different classes [51].

A number of common metrics are used to evaluate the effectiveness of the deep learning approaches in this work. The basic terminology will be described first.

- True Positives (TP) are the number of samples that are correctly predicted as positive (e.g. ground truth is ‘malicious’ and the prediction is also ‘malicious’).
- True Negatives (TN) are the number of samples that are correctly predicted as negative (e.g. ground truth is ‘benign’ and the prediction is also ‘benign’).
- False Positives (FP) are the number of samples that are negative but predicted as positive (e.g. ground truth is ‘benign’ and prediction is ‘malicious’).

- False Negatives (FN) are the number of samples that are positive but are predicted as negative (e.g. ground truth is ‘malicious’ and prediction is ‘benign’).

The performance of a supervised learning classification algorithm can be depicted via a confusion matrix, an example of which is shown in Figure 3.6. The rows indicate the ground truth and the columns indicate predicted class.

3.9 SUMMERY

Deep convolutional neural networks are effective when working with tabular data that consists of lots of examples, and with categorical variables of high cardinality, which are present in the domain of Cybersecurity and Network Intrusion Detection. The technique for embedding high cardinality categorical variables, which are common in cybersecurity data, leverages the power of deep convolutional neural networks to achieve better results than other leading techniques without needing to perform much in the way of hand-engineered features, especially when the amount of training data is larger. Furthermore, it was found that using the IP Address as an embedded categorical feature enabled the best performance for the deep convolutional neural network. It is theorized that by using the embedding technique and updating the weights of the IP address representation at each epoch, the neural network forms a type of inherent memory about the IP address feature in relation to the other features and the given label. We have obtained five categorical features from the NID KAGGLE dataset which can be expanded to a maximum of 17, 002 + 19, 112 + 64, 638 + 53, 791 + 3 = 154,546 features. Additionally, high performance can be achieved when using only the first three octets of the IP address are used, as opposed to the full IP address, allowing for a more robust feature that withstands changes that would typically be caused by updates from a DHCP server.

4. RECOGNITION RESULTS

In this section of thesis, evaluation and results of experiments are provided from using a fully connected deep feedforward neural network classifier on the NID Kaggle dataset.

4.1 EXPERIMENTAL SETUP

In this section, we have considered two parameters while analyzing the deep learning algorithm (DCNN) on the network devices. They are accuracy in terms of percentage and time in terms of seconds. The experiment is carried using MATLAB R2019b Windows environment on 4GB RAM and 2.4 GHz Intel Core i5. After carrying out several experiments using the deep convolutional neural network, we have come to present all our findings here. We carried out deep convolutional neural network algorithms on our dataset. We have done the test procedure with 20% data which is equal to 13583 data. The deep convolutional neural network was created and trained on all 115 features. Training was limited to 60 epochs with additional condition of early stopping, using functionality provided by deep learning toolbox. This ensures that the model is trained only until the score on validation set is not getting worse – this in turn helps to avoid overfitting the training set.

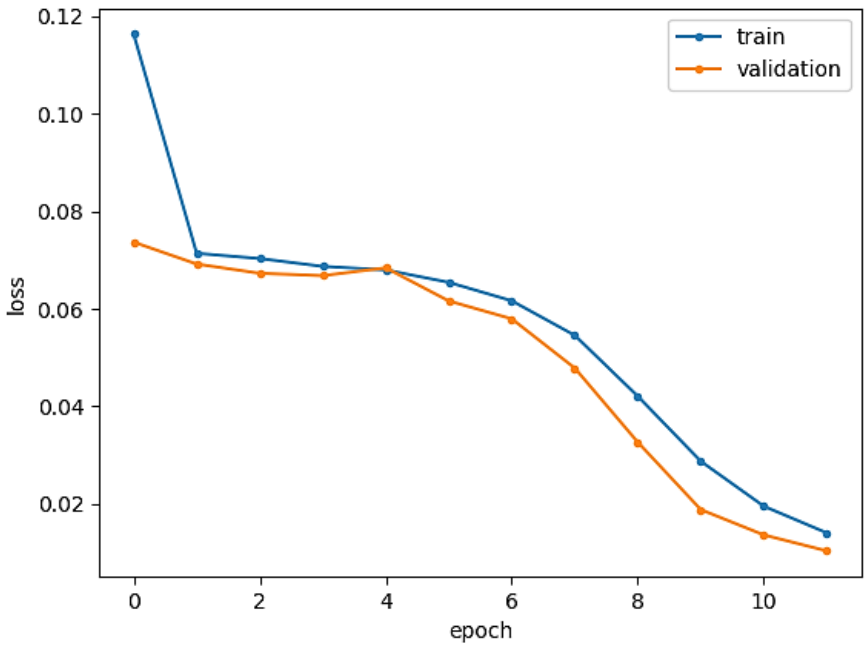

Figure 4.1: The validation of trained model stands at 10-epochs with respect to minimum loss.

4.2 OPTIMIZATION ASSUMPTIONS

When optimizing the categorical variables of Source/Destination IP Address and Ports, deep convolutional neural network performs exceptionally well at classifying malicious and benign flows. In addition, in experiments that omit the Source/Destination IP address features, but still utilizes detailed flow statistics, the neural network performs nearly as well, achieving an F1 Measure of 0.9730 in comparison to the experiment that included the IP address features, which achieved the 0.9980 F1 score. Therefore, training a neural network by omitting IP address features, given it contains detailed flow statistics, may be a viable way to use the trained model on a different computer network it has not seen before.

When using the IP address as a feature and representing it as a dense embedded vector at the first layer of the deep convolutional neural network, the best results are achieved. As the dense vector representation is among the first layer of the neural network, its representation gets updated at every epoch, in relation to the other features in the dataset as well as the supervised label [52, 53]. Therefore, it is theorized that the neural network forms a type of memory of the IP address in relation to the other features and the label, whereby it enables the highest performance for the classification task.

The deep convolutional neural network is trained to reconstruct inputs from a latent space representation based on only benign flows. Therefore, when it receives malicious flows as input, it is more difficult for the neural network to reconstruct the malicious flows, thus resulting in a higher reconstruction error. When there is sufficient amount of benign training examples, an deep convolutional neural network can be used to detect anomalous flows based on reconstruction error and setting a threshold.

4.3 INTRUSION DETECTION AND PREDICTION

A number of experiments and trials were run using the NID KAGGLE dataset, similar to the previous case study. Various neural network configurations and hyperparameters were used in order to achieve the best results, which are shown in Table 4.1. Using a neural network with three hidden layers of 64 units per layer, using the ReLU activation function for each hidden

layer, and with loss being calculated using the SoftMax cross entropy loss function. The optimizer used in obtaining these results is the DCNN optimizer.

As can been seen by the results, the deep convolutional neural network performs well in its ability to classify network flows as malicious and benign, achieving a low false positive rate of only 0.000127 and achieving an F1 score of 0.9980. This demonstrates that using the detailed flow statistics, in addition to using the embedding technique for the categorical variables of source/destination IP address and ports enables this classifier to achieve acceptable results.

Table 4.1: DCNN Evaluation Results – Metrics

Metrics	With IPs	Without IPs
True Negative	749, 299	745, 606
False Positive	237	3, 930
False Negative	137	5, 930
True Positive	183, 527	177, 734
Area Under the Curve	0.9995	0.9812
Accuracy	0.9996	0.9894
Error Rate	0.0008	0.0106
True Positive Rate (Sensitivity, Recall, Detection Rate)	0.9993	0.9677
True Negative Rate (Specificity)	0.9997	0.9948
False Positive Rate	0.0003	0.0052
False Negative Rate	0.0007	0.0323
Precision	0.9987	0.9784
F1 Measure	0.9990	0.9730

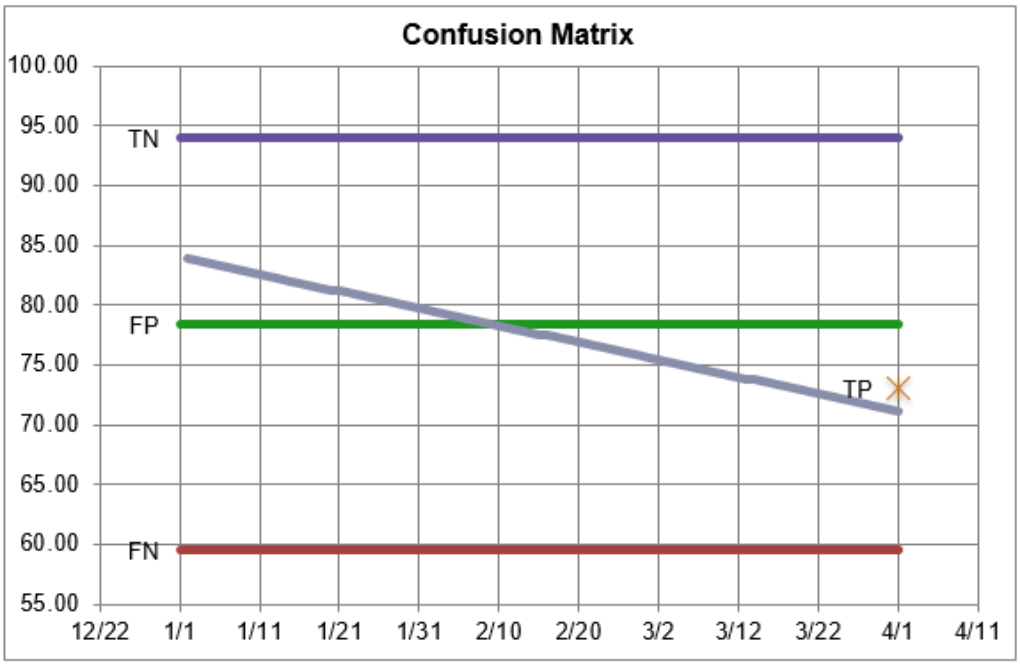

Figure 4.2: Network Intrusion Detection confusion matrix using embedding with all classes for evaluation.

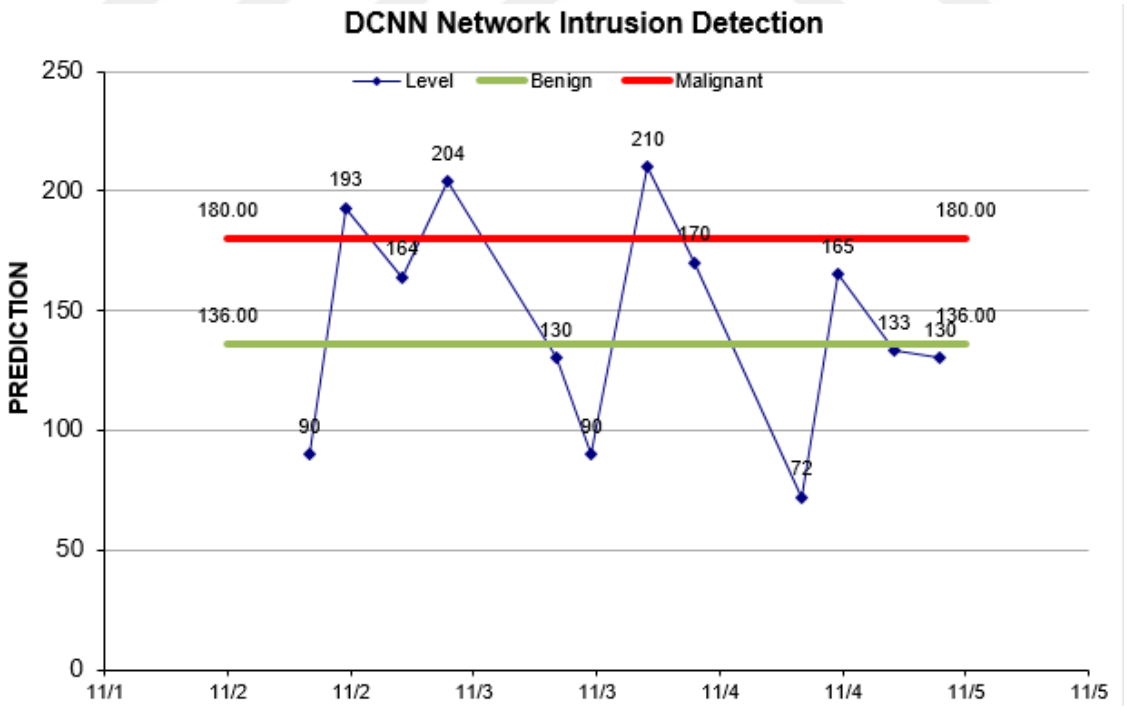

Figure 4.3: Prediction of attack on network with malignant percentile stands at 180 while the non-attacking benign percentile at 136 after 60% training of system on NID Kaggle Dataset using DCNN. Level represents the variation between the attacking and non-attacking parameters.

In reviewing these results, it shows that the using the IP address as a feature enables better performance of the classification task. However, in practice, the IP address can often change due to DHCP. Most often, the IP address will remain constant for the first three octets in a network environment, as this is determined by the configuration of routers and switches and is seldom changed. Instead, it is the last octet that can often change for a workstation on a network environment. Therefore, the key question is can IP address be used reliably as a feature in practice? How can the IP address be used while still accounting for its possibility to change due to DHCP? One approach is to use only the first three octets, instead of the entire IP address.

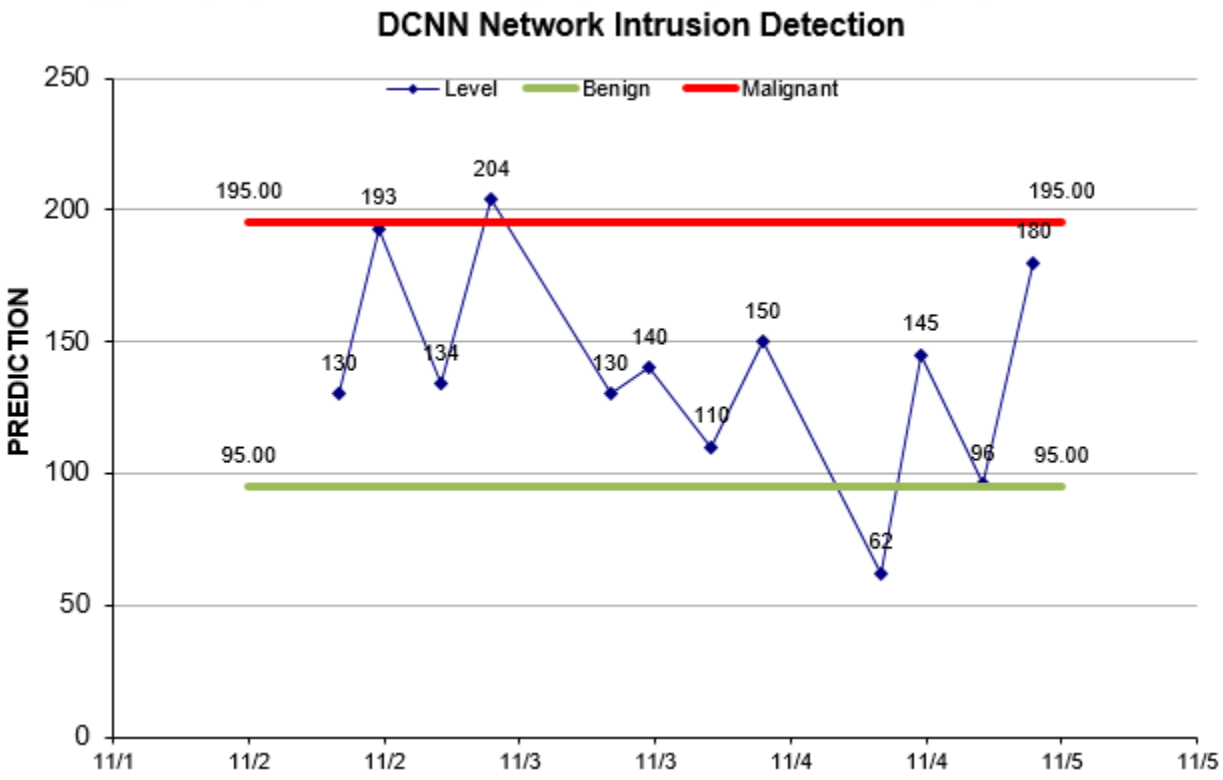

Figure 4.4: Prediction of attack on network with malignant percentile stands at 195 while the non-attacking benign percentile at 95 after 70% training of system on NID Kaggle Dataset using DCNN. Level represents the variation between the attacking and non-attacking parameters.

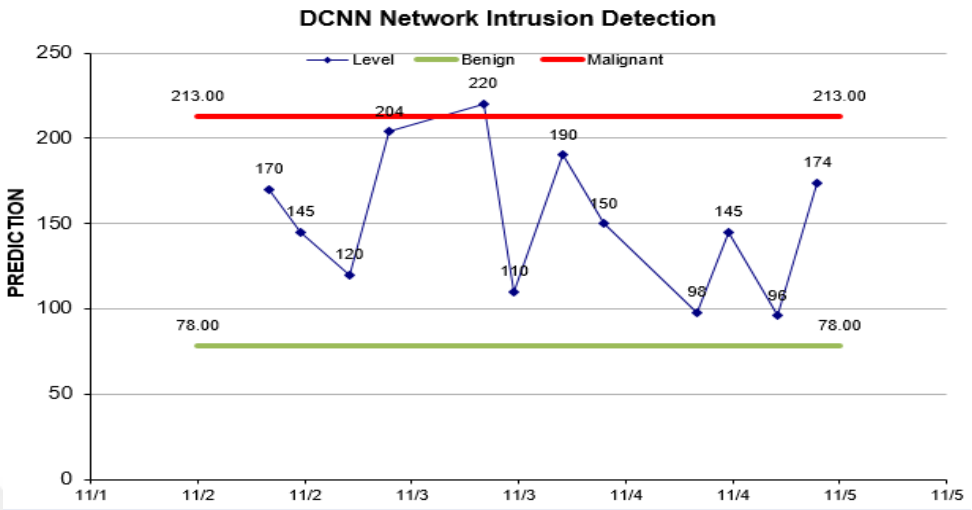

Figure 4.5: Prediction of attack on network with malignant percentile stands at 213 while the non-attacking benign percentile at 78 after 80% training of system on NID Kaggle Dataset using DCNN. Level represents the variation between the attacking and non-attacking parameters.

In addition to binary classification, the same neural network model was evaluated for multinomial classification. The multinomial classifier also performs well, for those categories for which it has a sufficient number of examples of each class.

Table 4.2: Multinomial Classification - Support Numbers

Class	Support
Bot	749, 536
PortScan	2, 619
Web Attack - Brute Force	1, 946
DoS Slowloris	1, 913
Web Attack - SQL Injection	1, 815
DoS Hulk	75, 941
Infiltration	3, 397
Heartbleed	4
SSH-Patator	497
Web Attack - XSS	215
DoS Slowhttptest	7
DoS Goldeneye	12
Benign	645
FTP-Patator	52, 405
DDoS	42, 248

In particular, our research used a dataset that had a higher amount of malicious training examples to learn from. The previous study with the ISCX 2012 dataset contained 2, 545, 935 total flows, of which 68, 910 (2.91%) were malicious. This work with the NID KAGGLE dataset contained 2, 830, 743 total flows, of which 557, 646 (19.70%) were malicious. In addition, the dataset for the previously used had less features (a total of 14, consisting of 7 continuous and 7 categorical), compared to the our research dataset (a total of 74, consisting of 69 continuous and 5 categorical), with the second dataset containing more continuous features made up of detailed flow statistics for each training example. Even when removing features from NID KAGGLE to make the two datasets have parity, the performance of the neural network was higher on NID KAGGLE with an F1 Score of 0.9990, compared to ISCX IDS 2012 with an F1 Score of 0.9491. The primary difference in this case was that NID KAGGLE contained a higher number of attacks to learn from. In addition, the NID KAGGLE dataset itself is more robust in how it was generated, as it adheres to the 11 criteria for building a robust benchmark dataset as we show in Figure 4.6.

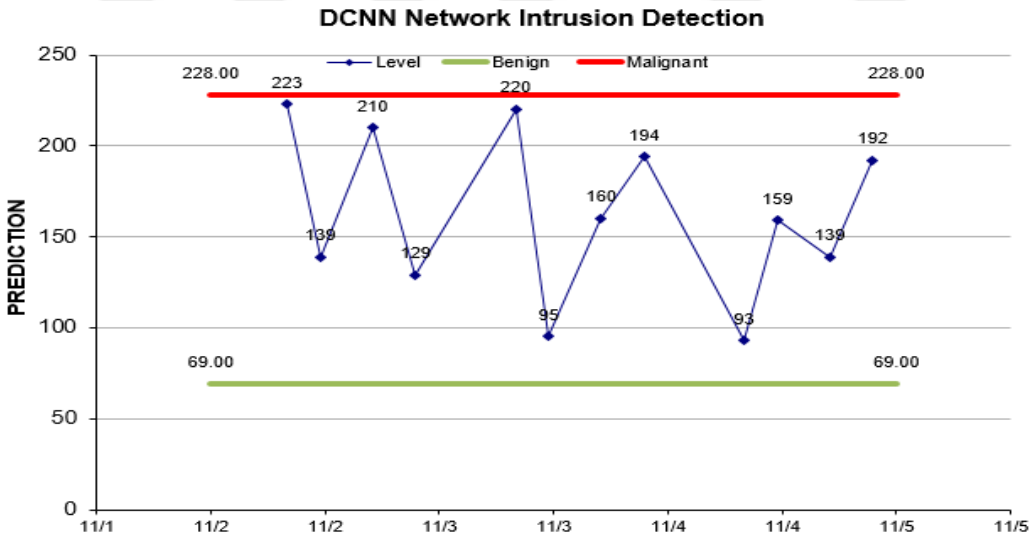

Figure 4.6: Prediction of attack on network with malignant percentile stands at 228 while the non-attacking benign percentile at 69 after 90% training of system on NID Kaggle Dataset using DCNN. Level represents the variation between the attacking and non-attacking parameters.

However, for those categories, there is only 4, 497, 215, 7, 12, and 645 amount of support (examples) in the dataset for each respective category. Therefore, this shows that the deep

convolutional neural network does not needs a sufficient number of examples to learn from in order to perform classification accurately.

4.4 LIMITATION OF STUDY

These experiments show promising results, especially when including the source and destination IP address as features. In some environments, it may be viable to include the IP address as a feature; however, in other environments it is often the case that IP addresses change due to DHCP. The experiments without IP addresses still performed well, but could not reach the level of results as when IP addresses were included as embedded categorical features. One way to overcome the limitation of IP addresses changing due to DHCP is to use only the first 3 octets of the IP address as a feature. Experiments show that this research performs just as well as using the complete 4 octets of the IP address.

4.5 SUMMERY

Deep convolutional neural networks are effective when working with tabular data that consists of lots of examples, and with categorical variables of high cardinality, which are present in the domain of Cybersecurity and Network Intrusion Detection. The technique for embedding high cardinality categorical variables, which are common in cybersecurity data, leverages the power of deep convolutional neural networks to achieve better results than other leading techniques without needing to perform much in the way of hand-engineered features, especially when the amount of training data is larger. Furthermore, it was found that using the IP Address as an embedded categorical feature enabled the best performance for the deep convolutional neural network. It is theorized that by using the embedding technique and updating the weights of the IP address representation at each epoch, the neural network forms a type of inherent memory about the IP address feature in relation to the other features and the given label. Additionally, high performance was still achieved when using only the first three octets of the IP address are used, as opposed to the full IP address, allowing for a more robust feature that withstands changes that would typically be caused by updates from a DHCP server.

In this chapter, results were presented that evaluated the use of deep fully connected neural networks for classifying network flows as benign or malicious. To perform the case studies,

recent benchmark dataset was used. The experiments showed results that outperformed other leading techniques when using deep convolutional neural networks for classification of network flows as benign or malicious. Further, it was shown that the use of flow statistics generated by the DCNN, in addition to a technique of dimensionality reduction of the categorical variables using the embedding technique, enabled the deep convolutional neural network to achieve an F1 measure of 0.9990, a detection rate of 0.9993, and a false positive rate of only 0.0003.

5. DISCUSSION

5.1 DISCUSSION

Deep convolutional neural networks are effective when working with tabular data that consists of lots of examples, and with categorical variables of high cardinality, which are present in the domain of Cybersecurity and Network Intrusion Detection. The technique for embedding high cardinality categorical variables, which are common in cybersecurity data, leverages the power of deep convolutional neural networks to achieve better results than other leading techniques without needing to perform much in the way of hand-engineered features, especially when the amount of training data is larger. Furthermore, it was found that using the IP Address as an embedded categorical feature enabled the best performance for the deep convolutional neural network. It is theorized that by using the embedding technique and updating the weights of the IP address representation at each epoch, the neural network forms a type of inherent memory about the IP address feature in relation to the other features and the given label. Additionally, high performance was still achieved when using only the first three octets of the IP address are used, as opposed to the full IP address, allowing for a more robust feature that withstands changes that would typically be caused by updates from a DHCP server.

In this work, network intrusion of a network was presented that evaluated the use of deep fully connected neural networks for classifying network flows as benign or malicious. To perform the case studies, recent benchmark dataset was used. The experiments showed results that outperformed other leading techniques when using deep convolutional neural networks for classification of network flows as benign or malicious. Further, it was shown that the use of flow statistics generated by the DCNN, in addition to a technique of dimensionality reduction of the categorical variables using the embedding technique, enabled the deep convolutional neural network to achieve an F1 measure of 0.9990, a detection rate of 0.9993, and a false positive rate of only 0.0003.

5.2 INVESTIGATION OF STUDY

Since we are working with intrusion detection and classification techniques, this requires that the system to identify threats that are known as well as unknown. Intrusion detection and

classification in the Internet of things based system is a supervised learning technique so it needs data set to train its system. We are using choosing to construct a defensive system for both Application Layer, which is host based and Network Later which is network based for a set of chosen machine learning algorithms. Since acquiring data set of labeled data in IoT for intrusion detection is tough so for the application layer data set. Each sensor board contains temperature sensor, humidity sensor, light sensor and voltage between the batteries. Due to condition of the labs being similar most of the data that we collected are similar. As for the network layer data set, we are using NID KAGGLE dataset that contains categorical features. These features are identifying attributes between hosts, attributes representing protocol connections, attributes of IP addressing, attributes of packet's start/end and round-trip time, general features like protection of the service of protocols and connection features and lastly labeled feature. On these data sets the deep learning algorithms are then implemented.

5.3 ANALYSIS OF STUDY

Deep convolutional neural network algorithm is a classification algorithm that classifies data into groups that augments the utility for researchers. Using DCNN we can split data according to its classification. DCNN cannot work well on data that is continuous too, because if the regularization parameters are set high, it shall still form a very thin hyper plane to set the data even when it is continuous. Since we have lots of data and it is arranged in a continuous manner after implementing DCNN on the data set, it shows a run-time accuracy of 80 percent in 2.76s. Similarly, DCNN is tested on the network layer data where it classifies by creating a hyper plane to separate attacking and non-attacking classes. Since the computational power required on the actual data set was high, we ran it on a pre-processed data set and where it concluded with an accuracy of 99.93% but took 4.88s to finish.

Table 5.1: Result comparison of DCNN with previously performed techniques

Technique	Avg. DR	Avg. FPR
Hybrid IDS - Decision Tree + Rule-based [27]	0.94475	.01145
WISARD [29]	0.48175	0.02865
Forest PA [29]	.92920	0.03550
J48 Consolidated [29]	0.92020	0.06645
LIBSVM [32]	0.54595	0.05130
FURIA [31]	0.90500	0.03165
Random Forest [31]	0.93050	0.01880
REP Tree [31]	0.91640	0.04835
MLP [31]	0.77830	0.07350
Naive Bayes [32]	0.82510	0.33455
Jrip [31]	0.93400	0.04470
J48 [31]	0.91990	0.05040
DCNN with IPs	**0.9993**	**0.0003**
DCNN without IPs	**0.9677**	**0.0052**

Deep convolutional neural network was also used and it works by using multi-variable analysis as it predicts a single outcome from multiple variable features. Our application layer data set has 14 features and has two classes. The model is trained by using the neural function which sets a threshold value in probability that finally determines the two classes. This algorithm shows an accuracy of 96.77% in 1.68s. Deep convolutional neural network is implemented on the network layer and input features that were trained was IP, protocols, port numbers etc. After the algorithm was tested the result column predicts whether the packet is an attacking packet or not and shows 96.77% accuracy in just 0.57s.

5.4 DIVERSIFICATION OF RESULTS

For the network intrusion detection and running environment for intrusion detection and classification, it can be concluded that deep convolutional neural network is the best algorithm in terms of detection and classification of attacks as it is the fastest algorithm. Moreover, DCNN's run time and accuracy is very high in terms of intrusion detection in network. Hence, it can be included while detecting intrusions. Furthermore, for the network layer, deep convolutional neural network has a very decent accuracy with least execution time.

Table 5.2: Result comparison of DCNN with other in terms of confusion matrix

Technique	Precision	Recall	F1 Score
KNN [33]	0.96	.96	.96
Random Forest (RF) [33]	0.98	0.97	.97
ID3 [34, 35]	.98	0.98	.98
Adaboost [34, 35]	0.77	0.84	.77
MLP [35]	0.77	0.83	.76
Naive-Bayes [36]	0.88	0.04	.04
Quadratic Discriminant Analysis (QDA) [36]	0.97	0.88	.92
DCNN with IPs	**0.9987**	**0.9993**	**0.9990**
DCNN without IPs	**0.9784**	**0.9677**	**0.9730**

Although we have achieved very good accuracy with the deep convolutional neural network algorithms, there are other data sets with more features and different complexities where these algorithms might not detect intrusions with great precision. On those data sets, other deep learning algorithms or intrusion detection techniques might not work better. Lastly, the intrusion detection and classification is done on the end devices. We can carry out this detection and classification on the cloud or server in the near future with the help of deep learning as recommended.

6. CONCLUSION

6.1 CONCLUSION

In the field of network intrusion detection, there is an abundance of data available from packet captures of network flows. These packet captures can be converted into network flows that contain rich metadata about the statistics of each flow, which are composed of the captured packet data. This work focuses on using newer benchmark datasets that have recently become available to the research community, the NID KAGGLE datasets. This dataset is splitted in 80% for training and 20% for testing with 10-cross fold for validation. The deep convolutional neural network performs well in its ability to classify network flows as malicious and benign, achieving a low false positive rate of only 0.000127 and achieving an F1-score of 0.9980. After the algorithm was tested the result column predicts whether the packet is an attacking packet or not and shows 96.77% accuracy in just 0.57s.

Deep convolutional neural network can be a powerful tool when used with tabular data, especially when it contains categorical features of high dimensionality. Using an embedding technique, these high cardinality categorical variables can be converted to dense floating-point vectors that are orders of magnitude smaller. Furthermore, on the NID KAGGLE dataset, which contained more examples for each class, and many more features composed of flow statistics, the performance of the neural network increased. Also, a technique for embedding IP address features using only the first three octets was discovered that overcomes the limitations imposed by DHCP – using only the first three octets performed just as well as using the full IP address.

6.2 FUTURE RECOMMANDATION

There are a number of future directions that can be pursued, including the following:

- Use time domain information as an embedded categorical variable. For example, use different resolutions such as day of week, hour of day, or minute of hour as embedded categorical variables. This is highly dependent on the signature of an attack, but if an adversary chooses to attack at a certain time, this may help increase precision and recall. In addition, evaluate these features on their ability to improve the performance of the DCNN.

- Perform additional strategies for embedding the IP address as a categorical variable. For environments where the IP address frequently changes, other techniques for including the IP address as a categorical embedded feature may be useful.

- Explore Recurrent Neural Network structures and how they can be used to leverage the time domain for classification of benign and malicious flows.

- Leverage categorical variables with the autoencoder approach to improve its performance in detecting anomalous flows. The current case study in this Thesis which evaluated an DCNN utilized only the continuous flow statistics as features.

- Perform a systematic evaluation of varying features and hyperparameter optimizations to determine the deep convolutional neural network configuration that produces the best results (both supervised and unsupervised) on the benchmark datasets used in this Thesis.

- CIC recently released a new dataset in 2020 that was created using the same framework for building the CIC IDS2017 dataset [11]. Therefore, training a deep convolutional neural network on one computer network traffic environment, and determining how well it generalizes to classifying flows on an entirely different environment will be a good next step. It would be of interest to determine algorithms and techniques in which a neural network trained on one network intrusion detection dataset can be generalized to perform effectively on a new dataset coming from a different computer network environment.

REFERENCES

[1] Xin, Y.; Kong, L.S.; Liu, Z.; Chen, Y.L. Machine learning and deep learning methods for cybersecurity. IEEE Access 2018, 6, 35365–35381.

[2] Ambusaidi, M.A.; He, X.J.; Nanda, P.; Tan, Z.Y. Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Trans. Comput. 2016, 65, 2986–2998.

[3] Ghazy, R.A.; El-Rabaie, E.M.; Dessouky, M.I.; El-Fishawy, N.A.; Abd El-Samie, F.E. Efficient techniques for attack detection using different features selection algorithms and classifiers. Wirel. Pers. Commun. 2018, 100, 1689–1706.

[4] Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J. Comput. Sci. 2018, 25, 152–160.

[5] Kang, S.H.; Kim, K.J. A feature selection approach to find optimal feature subsets for the network intrusion detection system. Cluster Comput. 2016, 19, 325–333.

[6] Salo, F.; Nassif, A.B.; Essex, A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. Comput. Netw. 2019, 148, 164–175.

[7] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2019 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2019; pp. 1–6.

[8] Beulah, J.R.; Punithavathani, D.S. A hybrid feature selection method for improved detection of wired/wireless network intrusions. Wirel. Pers. Commun. 2018, 98, 1853–1869.

[9] Bostani, H.; Sheikhan, M. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. Soft Comput. 2017, 21, 2307–2324.

[10] Acharya, N.; Singh, S. An IWD-based feature selection method for intrusion detection system. Soft Comput. 2017, 22, 4407–4416.

[11] NID KAGGLE. Available online: https://www.kaggle.com/sampadab17/network-intrusion-detection.

[12] Akashdeep, S.; Manzoor, I.; Kumar, N. A feature reduced intrusion detection system using ANN classifier. Expert Syst. Appl. 2017, 88, 249–257.

[13] Akyol, A.; Hacibeyoglu, M.; Karlik, B. Design of multilevel hybrid classifier with variant feature sets for intrusion detection system. IEICE Trans. Inf. Syst. 2016, ED99, 1810–1821.

[14] Bhattacharya, S.; Selvakumar, S. LAWRA: A layered wrapper feature selection approach for network attack detection. Secur. Commun. Netw. 2015, 8, 3459–3468.

[15] Panda, M.; Abraham, A.; Patra, M.R. Hybrid intelligent systems for detecting network intrusions. Secur. Commun. Netw. 2015, 8, 2741–2749.

[16] Ahmad, I.; Basheri, M.; Iqbal, M.J.; Rahim, A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access 2018, 6, 33789–33795.

[17] Aburomman, A.A.; Reaz, M.B. A survey of intrusion detection systems based on ensemble and hybrid classifiers. Comput. Secur. 2017, 65, 135–152.

[18] Lilakiatsakun, W.; Somwang, P. Anomaly traffic detection based on PCA and SFAM. Int. Arab J. Inf. Technol. 2017, 12, 253–260.

[19] Alabdallah, A.; Awad, M. Using weighted support vector machine to address the imbalanced classes problem of intrusion detection system. KSII Trans. Internet Inf. Syst. 2018, 12, 5143–5158.

[20] Li, L.J.; Yu, Y.; Bai, S.S.; Hou, Y.; Chen, X.Y. An effective two-step intrusion detection approach based on binary classification and kNN. IEEE Access 2018, 6, 12060–12073.

[21] Demir, N.; Dalkilic, G. Modified stacking ensemble approach to detect network intrusion. Turk. J. Electr. Eng. Comput. Sci. 2018, 26, 418–433.

[22] Kamarudin, M.H.; Maple, C.; Watson, T.; Safa, N.S. A LogitBoost-based algorithm for detecting known and unknown web attacks. IEEE Access 2017, 5, 26190–26200.

[23] Tian, Y.J.; Mirzabagheri, M.; Bamakan, S.M.H.; Wang, H.D.; Qu, Q. Ramp loss one-class support vector machine: A robust and effective approach to anomaly detection problems. Neurocomputing 2018, 310, 223–235.

[24] Kabir, E.; Hu, J.K.; Wang, H.; Zhuo, G.P. A novel statistical technique for intrusion detection systems. Future Gener. Comput. Syst. 2018, 79, 303–318.

[25] Ahmim, A.; Derdour, M.; Ferrag, M.A. An intrusion detection system based on combining probability predictions of a tree of classifiers. Int. J. Commun. Syst. 2018, 31, 1–14.

[26] Aburomman, A.A.; Reaz, M.B. A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. Inf. Sci. 2017, 414, 225–246.

[27] Yan, B.H.; Han, G.D. LA-GRU: Building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural network. Secur. Commun. Netw. 2018, 1, 1–13.

[28] Idhammad, M.; Afdel, K.; Belouch, M. Semi-supervised machine learning approach for DDoS detection. Appl. Intell. 2018, 48, 3193–3208.

[29] Mohammadi, S.; Namadchian, A. A new deep learning approach for anomaly base IDS using memetic classifier. Int. J. Comput. Commun. 2017, 12, 677–688.

[30] Imamverdiyev, Y.; Abdullayeva, F. Deep learning method for denial of service attack detection based on restricted boltzmann machine. Big Data-US 2018, 6, 159–169.

[31] Ma, T.; Wang, F.; Cheng, J.; Yu, Y.; Chen, X. A hybrid spectral clustering and deep convolutional neural network ensemble algorithm for intrusion detection in sensor networks. Sensors (Basel) 2016, 16, 1701.

[32] Shamshirband, S.; Daghighi, B.; Anuar, N.B.; Kiah, M.L.M.; Patel, A.; Abraham, A. Co-FQL: Anomaly detection using cooperative fuzzy Q-learning in network. J. Intell. Fuzzy Syst. 2015, 28, 1345–1357

[33] Al-Qatf, M.; Yu, L.S.; Al-Habib, M.; Al-Sabahi, K. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 2018, 6, 52843–52856.

[34] Hussain, J.; Lalmuanawma, S.; Chhakchhuak, L. A two-stage hybrid classification technique for network intrusion detection system. Int. J. Comput. Int. Syst. 2016, 9, 863–875.

[35] Li, L.J.; Yu, Y.; Bai, S.S.; Cheng, J.J.; Chen, X.Y. Towards effective network intrusion detection: A hybrid model integrating Gini index and GBDT with PSO. J. Sens. 2018, 6, 1–9.

[36] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), PIEAS, Islamabad, Pakistan, 26–27 August 2016; pp. 770–778.

[37] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

[38] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein,M.; et al. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. 2016, 115, 1–37.

[39] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. Comput. Sci. 2016, 9, 1–14.

[40] He, K.; Zhang, X.; Ren, S.; Jian, S. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1–11.

[41] He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the 2017 European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2015; pp. 630–645.

[42] Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. 2017, 16, 321–357.

[43] Wu, K.H.; Chen, Z.G.; Li, W. A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access 2018, 6, 50850–50859.

[44] Le, T.T.H.; Kim, Y.; Kim, H. Network intrusion detection based on novel feature selection model and various recurrent neural networks. Appl. Sci.-Basel 2019, 9, 1392.

[45] Panda, M.; Abraham, A.; Patra, M.R. Discriminative multinomial naive Bayes for network intrusion detection. In Proceedings of the 2017 Sixth International Conference on Information Assurance and Security, Atlanta, GA, USA, 23–25 August 2017; pp. 5–10.

[46] Salama, M.A.; Eid, H.F.; Ramadan, R.A.; Darwish, A.; Hassanien, A.E. Hybrid intelligent intrusion detection scheme. Soft Comput. Ind. Appl. 2016, 96, 293–303.

[47] Gogoi, P.; Bhuyan, M.H.; Bhattacharyya, D.; Kalita, J.K. Packet and flow based network intrusion dataset. Contemp. Comput. 2017, 306, 322–334.

[48] Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 2017, 5, 21954–21961.

[49] Singh, R.; Kumar, H.; Singla, R.K. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. Expert Syst. Appl. 2015, 42, 8609–8624.

[50] Yang, Y.Q.; Zheng, K.F.; Wu, C.H.; Niu, X.X.; Yang, Y.X. Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. Appl. Sci.-Basel 2019, 9, 238.

[51] Kayacik, H.G.; Zincir-Heywood, A.N.; Heywood, M.I. Ahierarchical SOM-based intrusion detection system. Eng. Appl. Artif. Intell. 2017, 20, 439–451.

[52] Tsang, C.H.; Kwong, S.; Wang, H. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognit. 2017, 40, 2373–2391.

[53] Bamakan, S.M.H.; Wang, H.; Yingjie, T.; Shi, Y. An effective intrusion detection framework based on MCLP/SVM optimized by timevarying chaos particle swarm optimization. Neurocomputing 2016, 199, 90–102.