



REPUBLIC OF TURKEY

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**A QOS-AWARE SELF-UPDATING INTRUSION
DETECTION SYSTEM USING
REINFORCEMENT LEARNING**

Atheel Sabih Shaker ALAWSI

Doctor of Philosophy Thesis

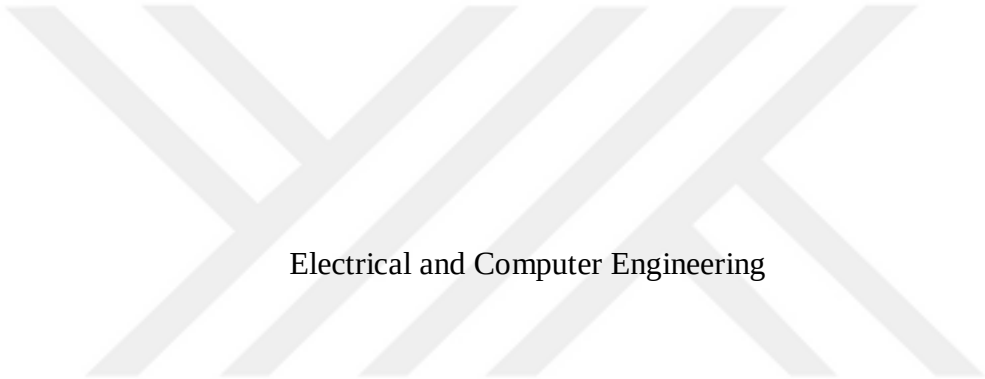
Supervisor

Asst. Prof. Dr. Sefer KURNAZ

Istanbul, 2022

A QOS-AWARE SELF-UPDATING INTRUSION DETECTION SYSTEM USING REINFORCEMENT LEARNING

Atheel Sabih Shaker ALAWSI



Electrical and Computer Engineering

Doctor of Philosophy Thesis

ALTINBAŞ UNIVERSITY

2022

The dissertation titled A QOS-AWARE SELF-UPDATING INTRUSION DETECTION SYSTEM USING REINFORCEMENT LEARNING prepared by ATHEEL SABIH SHAKER ALAWSI and submitted on 12/12/2022 has been **accepted unanimously** for the degree of PhD in Electrical and Computer Engineering.

Asst. Prof. Dr. Sefer KURNAZ

the Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Sefer KURNAZ	Department of Software Engineering, Altınbaş University	_____
Prof. Dr. Osman NURİ UÇAN	Department Of Electrical and Electronics Engineering, Altınbaş University	_____
Assoc. Prof. Dr. Adil Deniz DURU	Department of Coaching Training, Marmara University	_____
Asst. Prof. Dr. Oğuz KARAN	Department Of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Zeynep ALTAN	Department of Software Engineering, Beykent University	_____

I hereby declare that this dissertation meets all format and submission requirements of a PhD dissertation.

Submission date of the thesis to Institute of Graduate Studies: ____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Atheel Sabih Shaker ALAWSI

Signature

DEDICATION

I dedicate and promise this study effort to my supervisor, who has been instrumental in helping me through the entire research process, as well as my family, who has always been supportive during my difficult times.



PREFACE

First and foremost, I want to express my gratitude to my supervisor, Asst. Prof. Dr. Sefer KURNAZ, for guiding and assisting me during the composition of this dissertation. I met with my supervisor, Asst. Prof. Dr. Sefer KURNAZ weekly discusses my progress, challenges, and suggestions. This helped me enormously in grasping the reasoning behind the research. It helped me understand the technological requirements for this study project.



ABSTRACT

A QOS-AWARE SELF-UPDATING INTRUSION DETECTION SYSTEM USING REINFORCEMENT LEARNING

ALAWSI, Atheel Sabih Shaker

Ph.D., Electrical and Computer Engineering, Altınbaş University

Supervisor: Asst. Prof. Dr. Sefer KURNAZ

Date: December / 2022

Pages: 75

Intrusion Detection Systems are increasingly employing Machine Learning (ML) technology to keep up with hackers' ever-evolving methods for breaking into computer networks (IDS). Compared to unsupervised approaches, supervised algorithms have demonstrated capable of rapidly assessing and predicting the kind of an incoming packet. In contrast to supervised machine learning systems, which require additional training every time a new form of network attack is introduced, unsupervised methods have proven to be more easily updated. In response to this issue, this research presents a novel IDS based on RL. It can automatically identify patterns in new types of attacks and block packets that fit those patterns. This recognition is contingent upon the RL's ability to adapt to changes in the environment by evaluating the rewards of the selected activities, which in the proposed method are determined based on the Quality of Service (QoS) being provided on the network. To avoid blocking all incoming traffic and enhance QoS and, by extension, incentive value, the suggested technique incorporates the ratio of network-accessible traffic into the award computations. This is due to the fact that the most benefit is obtained when all normal traffic is permitted access. Several tests are undertaken to evaluate the hyperparameters of the proposed technique and its capacity to adapt to new attack traffic while maintaining network security. Results indicate that the suggested strategy is effective, with an accuracy of 96.72 percent on traffic comprising assaults that were not used during neural network training for the RL agent. Since a neural network based on categorization failed to identify any of the packets comprising these attacks, the proposed method is preferable.

Keywords: Cyber Security; Intrusion Detection System; Reinforcement Learning; Artificial Neural Networks; Machine Learning.

TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
ABBREVIATIONS.....	xiii
1.INTRODUCTION	1
1.1. PROBLEM STATEMENT.....	4
1.2. AIM OF THE THESIS	5
1.3. THESIS ORGANIZATION	6
2.LITERATURE REVIEW	7
2.1. BACKGROUND.....	7
2.2. NETWORK DEVICES	12
2.3. NETWORK SECURITY.....	13
2.3.1. Intruder.....	13
2.3.2. Intrusion	14
2.4. HISTORY OF INTRUSION DETECTION SYSTEMS.....	14
2.5. TYPES OF NETWORK ATTACKS	15
2.5.1. Passive Attacks.....	15
2.5.2. Active Attacks	15
2.6. INTRUSION DETECTION USING MACHINE LEARNING AND DATA MINING	16
2.6.1. Unsupervised Machine Learning.....	17
2.6.2. Supervised Machine Learning.....	17
2.6.3. Reinforcement Learning	21
2.7. ARTIFICIAL NEURAL NETWORKS	24
2.8. DEEP REINFORCEMENT LEARNING	30

2.9. RELATED WORKS	31
3.METHODOLOGY.....	34
3.1. OVERVIEW	34
3.2. THE AGENT NEURAL NETWORK	34
3.3. REWARD VALUE COMPUTATION AND NEURAL NETWORK TRAINING.....	36
3.4. ACTION SELECTION	38
4.EXPERIMENTAL RESULTS.....	39
5.DISCUSSION	52
6.CONCLUSION.....	54
REFERENCES	55

LIST OF TABLES

	<u>Pages</u>
Table 3.1: Structure of the neural network for the proposed method.	36
Table 4.1: A detailed analysis of this dataset can be found in [138].	39
Table 4.2: Performance summary of the proposed method when $\alpha = 0.3$	41
Table 4.3: Performance summary of the proposed method when $\alpha = 0.5$	43
Table 4.4: Performance summary of the proposed method when $\alpha = 0.7$	45
Table 4.5: Performance summary of the proposed method when $m=10$	47
Table 4.6: Performance measures of the classification-based neural network.....	50
Table 5.1: Comparison between. RL-based and classification-based neural networks in IDS.	53

LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: Overview of machine learning categories [13].....	2
Figure 1.2: Overview of the agent interaction with environment in RL [26].	4
Figure 2.1: Examples of SVM classification borders.	9
Figure 2.2: Fuzzy membership function illustration.....	11
Figure 2.3: Potential boundaries in a two-dimensional space that separate records.....	19
Figure 2.4: Margins surrounding the lines dividing the domain space.	19
Figure 2.5: A reinforcement learning example of the interaction between the agent and the environment.	22
Figure 2.6: The Basic Elements of Neurons [95].	25
Figure 2.7: The Computations of the Neuron are Visualized [98].	26
Figure 2.8: Neurons' Activation Function [101, 103].	27
Figure 2.9: Examples of Artificial Neural Networks that Feed Forward.....	28
Figure 2.10: Function that is convex versus non-convex [107].	29
Figure 2.11: both DQN and DDQN DQN, from top to bottom; the upgraded DDQN [119].	31
Figure 3.1: The proposed RL-based IDS's proposed hierarchy.....	34
Figure 3.2: The design of the neural network used to power the RL agent.	35
Figure 4.1: Confusion matrix of the predictions of the proposed method when $\alpha = 0.3$	41
Figure 4.2: Response of the proposed IDS to the network attacks when $\alpha = 0.3$	42
Figure 4.3: Confusion matrix of the predictions of the proposed method when $\alpha = 0.5$	43
Figure 4.4: Response of the proposed IDS to the network attacks when $\alpha = 0.5$	44
Figure 4.5: Confusion matrix of the predictions of the proposed method when $\alpha = 0.7$	45
Figure 4.6: Response of the proposed IDS to the network attacks when $\alpha = 0.5$	46
Figure 4.7: Performance of the suggested method with a parameter m value of 10.	47
Figure 4.8: Confusion matrix of the predictions of the proposed method when $m = 10$	48
Figure 4.9: performance of the suggested technique with $m = 5$	49
Figure 4.10: Confusion matrix of the predictions collected from the classification-based neural network.	50
Figure 4.11: The effectiveness of the classification-based deep neural network method [123].	51

Figure 5.1: Performance summary of the proposed method at different values of the parameter α with $m=5$	52
---	----



ABBREVIATIONS

DoS	: Denial Of Service
DDoS	: Distributed Denial Of Service
ML	: Machine Learning
RL	: Reinforcement Learning
QoS	:Quality Of Service
IDS	: Intrusion Detection System
U2R	: User To Root
R2L	: Remote To Local
K-NN	: K Nearest Neighbor
SVM	: Support Vector Machine
RF	: Random Forest
FAR	: False Alarm Rate
RIP	: Routing Information Protocol
HIDS	: Host-Based Intrusion Detection System
NMS	: Network Monitoring System
ASIM	: Automated Security Incident Measurement
CIDF	: Common Intrusion Detection Framework
NADIR	: Network Anomaly Detection And Intrusion Reporter
DIDS	: Distributed Intrusion Detection System
ARP	: Address Resolution Protocol
TCP	: Transmission Control Protocol
DL	: Deep Learning
TanH	: Hyperbolic Tangent

ReLU : Rectified Linear Unit
SSE : Sum Of Squared Errors
MSE : Mean Squared Error
DQN : Deep Q-Network
DDQN : Dueling Deep Q-Network
RNN : Recurrent Neural Network
CNN : Convolutional Neural Network



1. INTRODUCTION

Recent emphasis has been paid to the security of the computer networks used to deliver digital services online due to the rapid increase in the number of digital services available online. Because they are widely accessible online [1, 2], attackers have access to the networks and servers that provide these services. This makes it more difficult than ever to differentiate between malicious and benign data flow, and as a result, attackers' tactics for launching attacks are constantly evolving [3, 4]. In order to evade discovery, bad actors can make their communications appear to originate from reputable sources.

Before choosing whether to provide network access, traditional network security solutions analyse each packet based on its type using static rules. For example, a rule that expressly permits users to utilize the network's web service, which is meant to provide such a service, would permit all incoming traffic destined for the network. If the database management server is not available to the general public, no packets submitted to the service will be allowed to enter the network. Therefore, membership is restricted to those who want to utilize the network's services for legal purposes. Denial of service (DoS) attacks, in which an excessive number of requests are sent to a service in an effort to block or slow down access by legitimate users [5-7], are one example of the numerous contemporary attack techniques that exploit legitimate services to carry out the intended abusive network usage.

Due to the availability of such attacking tactics, the usage of static rules for each packet has become obsolete, and increasingly complex procedures are being developed to protect networks against such attacks. By specifying the average data transfer rate of each authorized user, for instance, networks can be protected from Denial of Service attacks. If a client makes an excessive number of requests, they are labeled as an intrusion and all interactions from their site are either temporarily or permanently prohibited. Distributed Denial-of-Service (DDoS) attacks are one way to circumvent these safeguards [5, 8, 9].

Multiple computers unleash a coordinated traffic onslaught against a single target, typically a server or network, during a Distributed Denial of Service (DDoS) attack. Consequently, the network gets flooded with data, but no hosts exceed the limit. Additionally, more machines may be infected by the attacker and used as "zombies" to

carry out the assault, resulting in fewer requests being sent to individual hosts but a continuous high volume of requests being sent to the network or server [9, 11].

As a result of the rapid evolution and rising sophistication of attackers' attack techniques, the packet characteristics of both criminal and normal network traffic have remarkable converged. ML approaches are used to comprehend the intricate relationship between the features that each packet uses to uniquely identify itself and the type of traffic it belongs to, such as normal or attack traffic. The three primary disciplines of machine learning (ML) are reinforcement learning, unsupervised techniques, and supervised procedures [12, 13]. (RL). Figure 1.1 provides an overview of the many types and uses of machine learning.

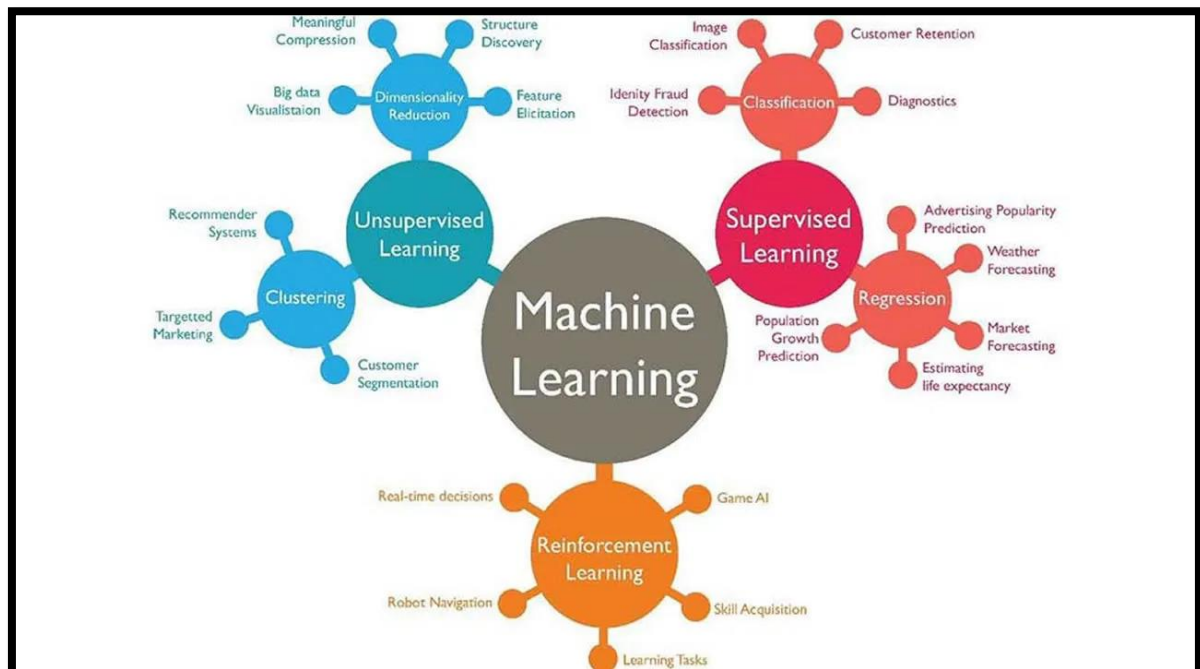


Figure 1.1: Overview of machine learning categories [13].

In order to identify the relationship between the instance properties and the labels assigned, supervised machine learning approaches require a large number of samples from the environment together with their labels, which provide extra information. Because the labels given to the dataset represent additional information from an expert in the environment from which the data are being collected, the process of determining the associations between the input features and the labels is known as supervised learning. In classification, a supervised ML field frequently used in intrusion detection for network security, these labels define the categorization of each occurrence, such as attack or normal, as well as the type of attack, when applicable. After collecting the essential

knowledge, the classifier may predict the nature of future inputs (packets) via pattern matching [14-16]. Therefore, if the network suspects a packet is part of attack traffic, it will not permit it to transit. Otherwise, the packet is presumed to originate from a trusted source and is permitted to enter the network. Due to the rapid evolution of attack packet characteristics, the classifier must be retrained whenever a new attack is presented [17].

Unsupervised machine learning (ML) algorithms search for and employ patterns already existing in the cases to attain the goal. Therefore, the environment samples obtained by the unsupervised ML algorithm do not require any further information. Clustering, a prominent unsupervised ML technique, splits data into groups whose members are more similar than those of other clusters [18]. This method is used for intrusion detection by presenting examples of both malicious and normal network traffic. The cluster to which an incoming packet belongs can therefore be used to deduce the nature of the packet, i.e., whether it is an attack or benign datagram. This strategy expedites the process of updating the intrusion detection mechanism by providing examples of the new sorts of attacks. As a result of the huge number of instances that must be clustered with the incoming packet, the processing of these techniques is more complicated compared to the use of classification, where the packet is treated solely [19, 20].

Unlike other machine learning (ML) techniques, Reinforcement Learning (RL) does not require samples from the real-world environment in which it functions. This sort of machine learning depicted in Figure 1.2 relies on receiving feedback from the environment it is dealing with, i.e. executing an action depending on the state of the agent selecting the action, which is represented by the observation that the agent collects from the environment. To enhance its forecast of the reward that will be taken from the environment when the action chosen at that observation or state is completed, the agent uses the reward that is taken from the environment. This modification enables the agent to reconsider its decision in light of fresh information and adapt to environmental changes that may affect the effectiveness of its operations. Outside of network intrusion detection, there are other applications for this type of ML, including robot navigation and autonomous vehicles [21-23]. However, this area of ML techniques has not garnered enough attention for the same reason both unsupervised and supervised methods for intrusion detection in computer networks have [24, 25].

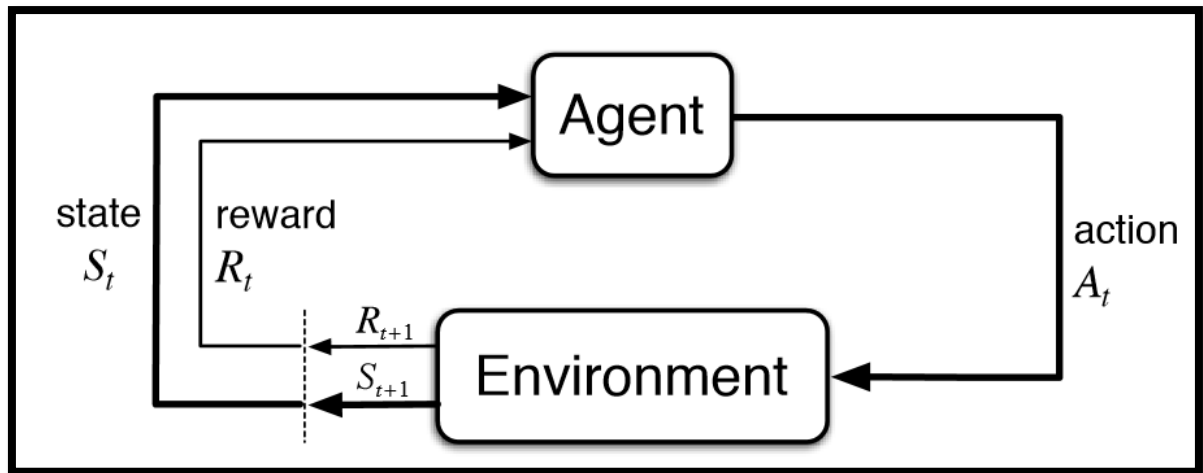


Figure 1.2: Overview of the agent interaction with environment in RL [26].

The major difficulty of RL is estimating a function that can anticipate the amount of reward for each action based on data collected from the environment. One of the most effective strategies for approximating a function is the use of artificial neural networks, which can accomplish this goal by modifying a set of parameters used in a mathematical representation that enables it to accomplish this goal by simply receiving the appropriate inputs and outputs. Consequently, neural networks are used to manage the huge amount of possible observations and to forecast the reward values for actions in scenarios the agent has never faced before, based on prior knowledge acquired from comparable observations. Utilizing deep neural networks has resulted in the creation of deep reinforcement learning, which has proven to be highly effective in a variety of scenarios.

1.1. PROBLEM STATEMENT

The expansion of online services over the past few years has led to an increase in the importance of protecting the servers and databases that underpin these services. This defense relies on detecting and removing malicious network traffic initiated by an attacker. Maintaining the quality of service (QoS) for authorized users and protecting sensitive data stored on the servers [27, 28].

In recent years, the sophistication of the methods used to break such networks has increased. Due to the obvious similarities between malicious and lawful traffic, typical methods of traffic analysis, such as comparing packet data to a predefined rule set, are insufficient. As a result, it is becoming increasingly difficult to discern between various forms of traffic [29].

The capacity to discover detailed relationships between input feature values and the intended output is what makes Machine Learning (ML) techniques valuable for differentiating between legitimate and attack traffic. Most intrusion detection systems rely on supervised approaches, namely various classifiers, to identify attack packet properties and filter malicious data (IDS). To train classifiers, we require labeled data in which the properties of each packet indicate not just its attack or normal type, but also its status. These classifiers can predict the state of incoming network packets by comparing their features to those of the training examples [30].

Many of these approaches are effective [15, 31, 32], however the emergence of new attack techniques poses difficulties for the functioning of IDSs based on these classifiers. Unfortunately, the classifier has sufficient knowledge on the properties of the packets utilized in the new assaults to properly identify them. It is necessary to incorporate a collection of packet samples from these novel attacks to the existing training samples in order to train the classifiers to recognize them. [33].

1.2. AIM OF THE THESIS

A new method is designed in this thesis, to address the limitations in the use of ML in IDS, by using RL instead of the supervised or unsupervised approached. The proposed method aims to achieve the following requirements:

- a. Protect the computer network by predicting the type of each packeting attempting to access the network, i.e., is it a normal packet or from attack traffic.
- b. Automatically learn the new patterns in attacks that are being executed against the network, without the need for any samples from these attacks or additional training to the proposed method.
- c. Maintain high Quality of Service (QoS) of the network by blocking any suspicious traffic.
- d. Maintain the ability to reconsider its decision to avoid blocking legitimate users, when false positives occur in the system.

A RL method meets these requirements; it is automatically updated by rewarding the agent with higher network QoS. As a result, the agent is able to reevaluate its choice to provide packet access to the network anytime a new type of assault is conducted and the QoS drops. If quality of service (QoS) is the only factor utilized to determine the agent's

reward, the agent may be biased toward restricting all packets from entering the network to maximize incentives. Consequently, the reward value of the proposed solution is maximized only when the agent permits all genuine traffic to flow through, i.e., when the agent permits most of the traffic to access the network while preserving good QoS.

1.3. THESIS ORGANIZATION

Following this introduction, the subsequent chapters of this thesis are as follows:

- i. In Chapter 2, the approaches implemented in the recommended methodology for intrusion detection with machine learning are reviewed along with a literature survey.
- ii. In Chapter Three, the recommended method is described in detail, including the neural network structure used by the agent of the reinforcement learning (RL) approach and the formula used to determine the reward value, all with the objective of allowing maximum traffic to utilize the network while preserving excellent QoS.
- iii. Chapter 4 presents experimental findings evaluating the suggested strategy and highlighting its benefits for intrusion detection.
- iv. Chapter Five examines, evaluates, and compares the findings of the performance evaluation studies to current state-of-the-art methodology to demonstrate that the network functions more effectively when the suggested method is used to detect intrusion attempts.
- v. This dissertation's chapter 6 discusses the research's findings and recommendations.

2. LITERATURE REVIEW

2.1. BACKGROUND

Intrusion detection research makes considerable use of the KDD Cup [34] dataset. Each tuple in this dataset is classified as either an attack packet or a regular packet depending on the presence or absence of 41 distinct criteria. Data on four different attack types—probing, remote-to-local, denial of service, and user-to-root—is gathered (U2R). The method used to gather this information has two main problems. In the training dataset, 78% of the records are duplicates, compared to 75% in the testing dataset. This might lead to a bias in training that favors creating duplicate records. Our analysis of the dataset's complexity, which shows that even basic classifiers are capable of producing accurate predictions for records in the training and testing sets, strengthens our second problem.

Wei-Chao Lin and others[35].

A k-NN classifier used by IDS was trained using KDD CUP'99. The proposed method achieves 99.89% precision with a k-NN classifier. Being lazy, the K-Nearest Neighbors (k-NN) classifier waits to train or extract data until a new prediction is needed. Based on a new record's feature values, the k records with the highest similarity are used to forecast the class of the new record. Because it must determine how similar each prediction is to each record in the dataset, the k-nearest neighbor classifier takes longer to run than other classifiers.

The method put out by Sakchi Jaiswal et al. [36] has sped up prediction by reducing the amount of attributes the classifier uses to find records that are similar to the new record. The computing time needed is cut down with this technique. This method chooses attributes based on the information at hand. Equation 2.1 demonstrates that the information gain is equal to the product of an attribute's probability (p_i) and its entropy ($H(X)$). Equation 2.2 can be used to calculate the obtained information if the entropy value is known. The suggested technique's overall accuracy drops from 95% to 94.77% when fewer features are used than before.

$$H(X) = - \sum_{i=1}^m p_i \log_2 p_i \quad \text{Equation 2.1}$$

$$H(X) = H(X) - H(X|Y) \quad \text{Equation 2.2}$$

According to [37], an intrusion detection system employs the alternativeness decision tree classifier. On both training and testing iterations of the KDDCUP'99 dataset, the proposed technique achieved a maximum accuracy of 95.09 percent. By applying attribute values to each record, the decision tree classifier gives sets of IF/THEN rules that can be used to predict the class of each record. The sets are grouped into tiers, with the condition of the subsequent tier selected by the result of the current tier's condition test. These categories are produced by combining the attribute values and record classes from the training dataset.

The method created by Vrushali D. Mane and SN. Pawar [38] categorizes each entry in the KDDCUP'99 dataset as either benign or hazardous neural network activity. 17 of the original 41 qualities are used as inputs to the dataset to reduce the size of the neural network required to do this assignment to a single hidden layer with 10 neurons. Due to the fact that only 10% of the data is utilized, training time is cut by 90%. Nonetheless, this network's forecasts are 98.0% correct on average.

The more recent NSL-KDD CUP'99 dataset contains a number of the problems noted by John McHugh [39]. Even though duplicate records have been removed, the attacks that plagued the original KDD CUP'99 dataset still persist. However, the utility of artificially generated information is questioned, and the architecture of the ecosystem is cause for concern. Using this information, numerous intrusion detection system tests are developed.

The approach presented in [40] detects intrusions using a Support Vector Machine (SVM) classifier trained on the NSL-KDD CUP'99 dataset. When the proposed method was used to allow the SVM classifier access to all of the dataset's attributes, the classification accuracy was 82.37 percent. Figure 2.1 demonstrates that the SVM predicts the class to which new, unlabeled data belong based on their position within the domain and the set constraints. Support vector machine-based classifiers generate a dimensional space with the same number of dimensions as the dataset's characteristics. The differences between document types are then quantified.

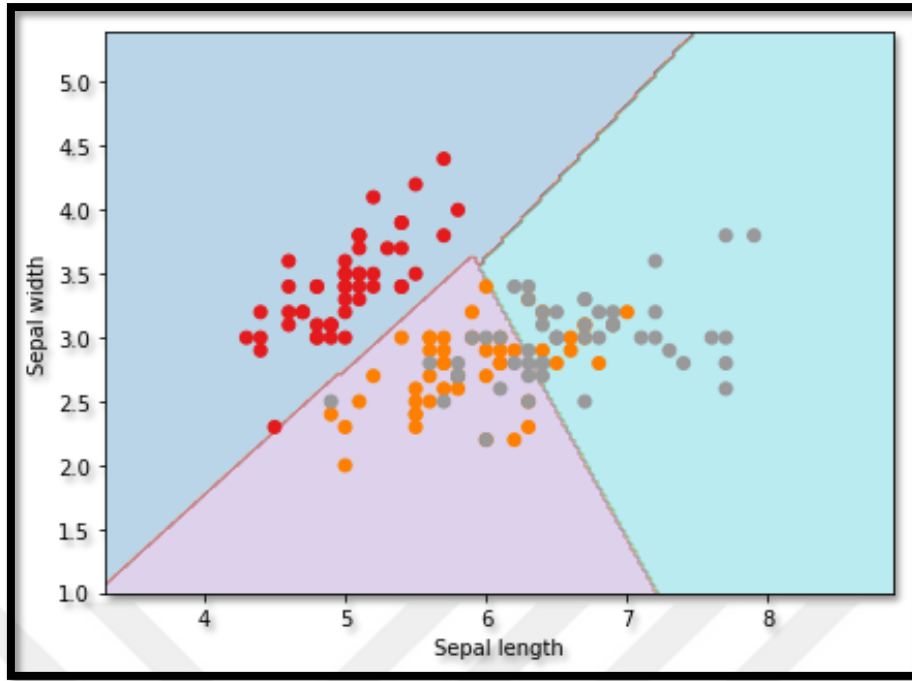


Figure 2.1: Examples of SVM classification borders.

In their experiment, L Dhanabal and SP Shantharajah [41] compare three distinct intrusion detection systems. Each system was trained using the NSL-KDD dataset. This classification group includes the decision tree classifier, the support vector machine classifier, and the Nave Bayes classifier. The accuracy scores of the aforementioned classifiers are 98.18%, 95.200%, and 73.320%. The Nave Bayes classifier allocates a record to a category by estimating probabilities for each attribute value based on the existing classifications in the dataset. We will select the most probable classification for the record. Using the modified values of the record's attributes, probabilities for each class within the dataset are then determined.

In [42], Bhupendra Ingre and Anamika Yadav describe two neural network-based categorization algorithms. The first model uses multiclass classification to place each dataset record into one of five categories, whereas the second model uses binary classification to determine whether a record represents an attack or a normal packet. Based on their relative relevance to the classification process, only 29 of the dataset's 41 attributes are used as inputs to the neural network. With an accuracy rate of 81.2%, records may be classified into five categories, but with an accuracy rate of 81.2%, they can be classified into two categories. The following section of the study includes extensive information regarding the mechanism of neural network classifiers.

Nour Moustafa and Jill Slay [43] advocate using the UNSW-NB15 dataset as it is more recent. This dataset is representative of a real-world communications network because it contains both usual and abnormal packets. There are ten separate packet types in this data set, one for normal traffic and nine for attacks. This type of attack typically manifests as worms, shellcode, reconnaissance, generic malware, and fuzzers.

Mustapha Belouch et al. [44] developed an intrusion detection system with a multi-stage decision tree classifier. During system training and evaluation, the NSL-KDD and UNSW-NB15 datasets are utilized. There are two steps to the procedure. First, we analyze if the packet represents safe network activity or an attempted intrusion. If an assault is suspected during the third step, its nature is identified. Using the proposed method, it is possible to classify the NSL-KDD and UNSW-NB15 datasets with an accuracy of 89.85%.

Using genetic algorithms to minimize the number of important classification criteria, a support vector machine (SVM) classifier learns from the dataset and predicts the classification of packets. [45] Hossein Gharaee and Hamid Hosseinvand are the developers of this IDS. Along with the normal class, the KDDCUP'99 and UNSW-NB15 datasets are utilized to test the implemented system. The UNSW-NB15 data set contains nine assaults, however only six are examined. The offered method improves precision from 99.26% on KDDCUP'99 to 93.25% on UNSW-NB15.

Ashfaq et al. [31] offer a fuzziness-based semi-supervised learning method that combines labeled and unlabeled data to enhance the quality of training data utilized by the classifier. The semi-supervised method uses fuzziness vectors to classify labeled and unlabeled data into groups from which predictions on the labels of the unlabeled data can be formed. After the unlabeled data have been labeled, additional data can be utilized for training since domain experts are no longer required to identify each item in the dataset. A single hidden layer is present in a feed-forward neural network that receives input data. The classification accuracy of semi-supervised learning on the NSL-KDD dataset was 84.12%, whereas Nave Bayes tree, J48, and SVM achieved 81.59%, 81.05%, and 69.50%, respectively. The precision of its classification is greater than that of the J48, the Nave Bayes tree, and the support vector machine.

Partha Sarathi Bhattacharjee et al. [46] designed a genetic algorithm-based intrusion detection system by combining weighted vectorized fitness functions with fuzzy

membership functions. Login predictions organize data into categories, as depicted in Figure 2.2, and the fuzzy membership function determines the probability that a given value belongs to a specific category. With the assistance of the Fuzzy Vectorized Genetic Algorithm, it appears that the NSL-KDD dataset's classification accuracy can be increased to a remarkable 99.18%.

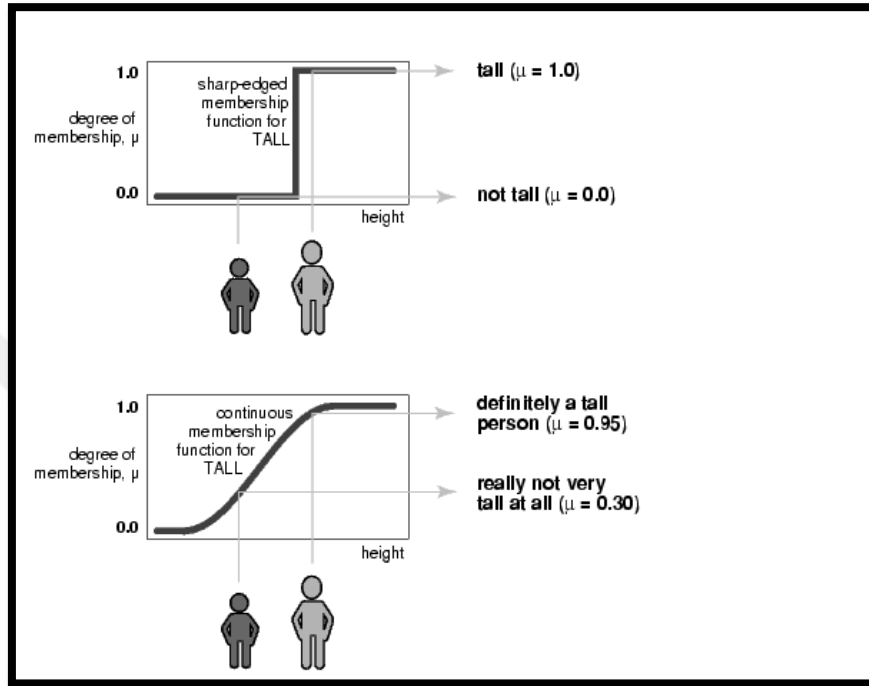


Figure 2.2: Fuzzy membership function illustration.

Moustafa and Slay [47] present an intrusion detection system that identifies incoming packets as either benign traffic or malicious attempts using linear regression. Based on the corresponding values of the data in the training dataset, linear regression produces a distribution of probabilities for tuples to belong to a class. To estimate the likelihood that newly collected data belongs to one of the preexisting classes, the attribute values are projected onto the distribution. Using the UNSW-NB15 and NSL-KDD datasets, it is examined. The result revealed that the implementation was more precise than both Expectation-Maximization and Naive Bayes. On both the UNSW-NB15 (83%) and the NSL-KDD (82.1%), the proposed strategy outperforms Expectation-Maximization and Naive Bayes.

According to Rifkie Primartha and Bayu Adhi Tama [48], the Random Forest (RF) classifier excels above decision trees, random trees, and multi-layer perceptrons. On the

KDD'99, NSL-KDD, and UNSW-NB15 network traffic data sets, random forest classifiers with 800 trees perform best on average. With a false alarm rate (FAR) of 7.22 percent and a success rate (CR) of 95.5%, the UNSW-NB15 random forest classifier performs admirably. False alarm rate is the fraction of innocuous packets that are mistakenly identified as incursion packets.

Malek Al-Zewairi et al. When the neural network has more layers and more neurons in each layer, more complex attributes can be extracted from the input data. The experimental results demonstrate that the deep learning model achieves a 98.99% accuracy, which is significantly greater than the other models studied in the study, including the decision tree, logistic regression, Nave Bayes, and neural network classifiers.

2.2. NETWORK DEVICES

Regardless of the medium used to connect network nodes, three distinct sets of devices are often involved in the construction process. "network hardware" typically refers to devices such as routers, firewalls, and switches in a computer network. To enable devices to connect without a central hub, we established a local area network with switches. Prior to the invention of switches, LAN nodes were connected via hubs. The hub will forward any packets delivered to it to all of its related ports, ensuring that all local area network devices have access to the data. Due to the fact that hubs share bandwidth, it is more likely that data traveling between two devices will be intercepted by other devices on the same subnetwork. However, switches remember which devices are connected to which ports, so data delivered to a known host is mirrored only on that port, and data sent to an unknown host is reflected on all ports. As the switch is aware of the source address and port of each packet it receives, it can keep track of the hosts that are transmitting data. This method improves the security of the switch by allocating each port its own bandwidth [50].

Routers are used to allow communication between endpoints on distinct networks and to connect disparate networks. Each port on a router can connect to a separate network with its own set of rules. Since the router transports data between subnetworks, every port on the network is directly accessible to every device within a subnetwork. Each router has its own table detailing the networks to which it may connect and how. This data may be entered manually by the administrator or generated dynamically by Routing Information Protocol messages exchanged between each pair of routes (RIP). Routers use only the

destination device's address from the routing table to determine where to deliver the following packet [51, 52].

Firewalls are hardware components that actively monitor and analyze incoming and outgoing data packets to prevent illegal communication. In contrast to routers, firewalls evaluate each packet's source address, source port, destination address, and destination port before deciding whether to allow it through. Firewalls are used to secure networks by prohibiting unauthorized users and limiting access to particular network resources. Therefore, firewalls should always incorporate network security [53, 54].

2.3. NETWORK SECURITY

A network's measures to prevent other networks from interfering with its services or abusing its resources. Modern networks are constantly under attack [55]; these attacks can originate from a multitude of channels and cause harm to a large number of devices. The vulnerabilities of business, government, and other private networks are identical to those of public networks [56, 57].

Due to the huge increase in demand for online services, the vast majority of businesses and service providers today feel forced to give clients with as many online options as possible. Due to this transition, even government services are now accessible online; therefore, it is essential for any business or institution that values its reputation to maintain a strong online presence. The provision of these services is contingent upon the storing of consumer information on internet-accessible servers. The companies in this industry owe it to their clients to respect their privacy and provide superior service. [58].

2.3.1. Intruder

A network intruder is a person who attempts unauthorized access to restricted resources via one or more networked devices [59]. Criminals, impersonators, and covert users are the three most common types of unauthorized entrants. A misfeisor is a network user who has been authorized access to some resources but attempts to get access to additional resources. This is in stark contrast to a masquerader, who attacks the network from the outside but has no actual access to its resources. In addition, hackers use the network devices to which they have access to steal user data. This type of attack may be launched locally, over the same network, or from outside the network, and would require the same degree of system access permissions.

2.3.2. Intrusion

An intrusion is performed when an unauthorized person accesses a service or piece of data on this network with the aim to impair the quality of that service or data [60]. These attacks may come from the victim's network or from another network with which it communicates. Additionally, some forms of infiltration rely on a single infected device, whilst others rely on a network of infiltrated devices to achieve their aims [61].

2.4. HISTORY OF INTRUSION DETECTION SYSTEMS

This study [62] is the first of its type to attempt to evaluate the data packets being transmitted over a specific network in order to learn about typical user activity and discover any unusual behaviors that could constitute a threat to the network. This study is widely regarded as the inspiration for all subsequent Host-Based Infiltration Detection Systems (HIDS), which seek to detect and block network intrusion attempts. On the basis of the study's findings, [63] provides an intrusion detection system that profiles authorized users of a government mainframe based on their online behavior and denies access to anyone who does not match the profile.

Morris was the first known malicious program to spread independently across the Internet. It was made accessible in November 1988. During the height of the worm's Internet destruction, two of the world's largest companies' DEC VAX and Sun-3 servers were rendered inoperable. Multiple infections will cause a machine to run extremely slowly and eventually crash. Before launching an attack, the code was incapable of determining whether or not the worm was already present on the machine. Thus, whenever the system is accessible, it will start attacks on other computers [64].

Since [65] introduced an intrusion detection system also known as Network Security Monitor, significant time and resources have been spent to the creation of several network security models (NSM). This solution employs hybrid techniques to identify risky individuals and subsequently deny them access to the network. Around the same time, the US Air Force's Cryptologic Support Center developed the Automated Security Incident Measurement (ASIM) system. This is the first system of its kind, and it uses specialized hardware and original software to establish an intrusion detection network. Maintainability, scalability, and effectiveness are among the most common metrics used in IDS evaluations; effort has begun to document these metrics. Distributed intrusion detection systems (DIDS) and network anomaly detection and intrusion reporting systems (NADIR) are two more classifications based on the design of these systems (DIDS).

These programs are designed to monitor and respond to network activities in order to defend several websites against intrusion [65].

Numerous commercial intrusion detection systems have been developed to safeguard networks from intruders since then. RealSecure by ISS, OmniGuard Intruder Alert by Axent, and NetRanger by Cisco are examples of commercial IDSs. As these systems rely on attack signatures, they must be updated anytime a new type of intrusion is identified. Numerous attempts have been made to create a Uniform Intrusion Detection Framework to standardize the terminology used to report intrusions (CIDF). However, there is currently no Intrusion Detection System (IDS) that can reliably identify all sorts of intrusions [66], making the development of such a framework challenging.

2.5. TYPES OF NETWORK ATTACKS

There are numerous existing and emerging types of network attacks, but they can be broadly categorized as passive or aggressive. Even if two attacks appear identical at first glance, they may contain subtle distinctions that enable them to circumvent common defenses [67–70]. Here is a list of all the major classifications, as well as a sample of the most common attacks within each category.

2.5.1. Passive Attacks

During passive assaults, the hacker monitors and analyzes network packet transfers without notifying the parties involved in the transaction. Unauthorized access is gained by an intruder to the information transferred between two confirmed persons. This form of intrusion is difficult to detect since the intruder may not leave traces indicating that they are monitoring data in transit [71–73]. Among the most common forms of passive assault include eavesdropping, traffic monitoring, and the distribution of private messages.

2.5.2. Active Attacks

During an active attack, intruders can send, receive, alter, and respond to network messages in order to obtain unauthorized access to data or services. It is possible that certain active assaults do not steal data, but rather delay or cease network service operation. Even while active attacks are easier to spot since the attackers engage with the servers, they can have a substantial impact on network performance [74-76]. Recent network intrusions include the following:

- i. Denial of Service (DoS): In these attacks, the target is the network itself, and more especially the servers and other network devices that grant authorized users access to their data. The objective of these assaults is to exhaust the network's resources (such as processing power or bandwidth) by transmitting a deluge of data that looks to originate from genuine users. The objective of such an attack is to damage the quality of these networks' services without gaining actual data access.
- ii. Spoofing Attack: In these attacks, the bad guys on the internet pose as a trusted user or service provider in an attempt to fool you into giving them access to your data. For instance, a hacker may transmit network packets to a server using a spoofed email address, pretending to be an authorized user attempting to access restricted data. The intruder is at risk of discovering sensitive information or making illegal use of limited resources in both cases.
- iii. The Man in the Middle Attack: This type of attack aims to interrupt a conversation between two trusted parties by inserting an unauthorized third party into the conversation. In this instance, the intruder not only acquires information but also has the ability to modify, add to, or remove portions of the parties' communications.
- iv. ARP Poisoning Attack: At the lowest layers of the Transmission Control Protocol (TCP), the Address Resolution Protocol (ARP) is used to determine the actual IP addresses of network interfaces that data is destined for. In these types of attacks, the attacker pretends to be the legitimate owner of the target address by responding to all or a subset of resolve requests as if they were for that address. This could result in the intruder gaining access to sensitive data, or it could prevent users from accessing essential services due to intercepted communications.
- v. Buffer Overflow Attack: Network interface buffers temporarily hold data before sending it to the next process. These buffers have a fixed amount of storage space, so if more data is transmitted to the network interface than the device can handle, the buffer will become full and begin deleting old data to make room for new data. The result is that the malicious user's data overwrites the data of the legitimate users.

2.6. INTRUSION DETECTION USING MACHINE LEARNING AND DATA MINING

"Machine learning" refers to giving computers the ability to learn from or make choices about the outside environment without human intervention. Even when the algorithms have never seen the inputs before, the same machine learning techniques can provide

distinct results. Data mining is an area of study within machine learning. Supervised and unsupervised machine learning methods are the two primary groups. [77].

2.6.1. Unsupervised Machine Learning

Unsupervised machine learning enables the extraction of data from unprocessed datasets, including attribute value associations. This approach to machine learning did not require labeling the dataset's records because the algorithms tended to detect correlations between records based on the characteristics of each record. Clustering is the most prevalent unsupervised data mining technique, and it classifies dataset elements according to their values. There are more links between clustered records than between clusters. Therefore, clustering produces redundant data sets [78].

The number of clusters has the greatest influence on the effectiveness of the clustering procedure. When there are too few groups, meaningless clusters form, and when there are too many groups, the processing time of the dataset's records increases without providing further benefits. It is necessary to evaluate the distribution of attribute values within the dataset in order to cluster the records into an appropriate number of clusters. This optimal cluster size can be chosen manually for the clustering method, or it can be generated using any of a variety of approaches that test several cluster sizes prior to determining the optimal size for the current dataset[79].

2.6.2. Supervised Machine Learning

Prior to analysis in this field of machine learning, inputs must be tagged. Certain types of machine learning, known as supervised methods, investigate the relationships between data inputs and their associated labels. Classification is a common supervised data mining technique that uses a record's label to determine its category. The classifiers then determine the links between the values of the attributes that constitute the definition of a record and its class. It is possible to make educated assumptions regarding unlabeled new data using these statistics. This projection may help predict how this new record [80] will behave in the future if it is comparable to other records in its category.

A labeled dataset is necessary to train an information-extracting classifier. This data set is referred to as a "training dataset." Because classifiers are used to make predictions, their efficacy cannot be evaluated on unlabeled data. In addition, it is not recommended to use the same training dataset because the classifier was trained with knowledge of the classes of these records; hence, the evaluation does not accurately reflect the accuracy of

the predictions. For more precise results, the labeled dataset is split in half. The classifier is tested with the second component after being trained with the first. To achieve accurate evaluation metrics, the testing dataset is fed into the classifier and its predictions for the records of the testing dataset are compared to their actual classes. This method leverages previously-established record classes as opposed to assessment data for training. Using a variety of classifiers, researchers can mine datasets for new information. In numerous ways, these classifiers extract the knowledge. However, the classifier's performance may vary based on the input data. For instance, the global effectiveness of a certain classifier may drop when applied to a specific dataset but increase when applied to a different dataset. Before selecting the most efficient classifier, it is required to assess the performance of various classifiers on a dataset. Some classifiers are more effective than others, including Deep Learning (DL), Random Forests, and Support Vector Machine (SVM) (SVM).

Each record in the dataset is represented by a point-like representation in a domain space with as many dimensions as there are attributes in the dataset. Using this domain space, we may interpret the dataset's information. The SVM classifier divides a given domain into subdomains based on the number of classes inside it. The SVM creates equations for the slipping borders at these spots to apply this approach. When a new record needs a forecast, the prediction's location is compared to these restrictions to determine an acceptable range. The class size projection is then recorded [82]

A simple representation of a two-dimensional region containing recordings, each with a unique label, is shown in Figure 2.3. The graphic displays multiple possible separations between the two zones, with each separation containing record label-specific information. Therefore, selecting the exact border for splitting the regions is essential for more precise future estimates.

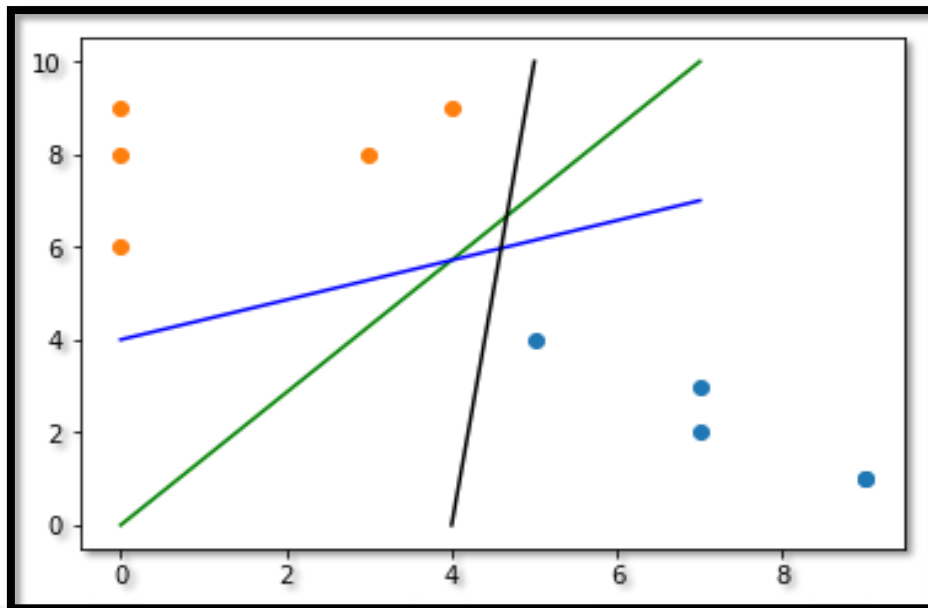


Figure 2.3: Potential boundaries in a two-dimensional space that separate records.

SVM classifiers gauge the accuracy of a variable's prediction based on the distance between the record and the boundaries. To increase prediction confidence and lower error rates, margins are formed around the space's boundaries to represent the shortest distances between the boundary and the nearest point of each class. The classifier seeks optimal margins given the stated restrictions. When evaluating new data, the green line is chosen based on the boundary margins depicted in Figure 2.4.

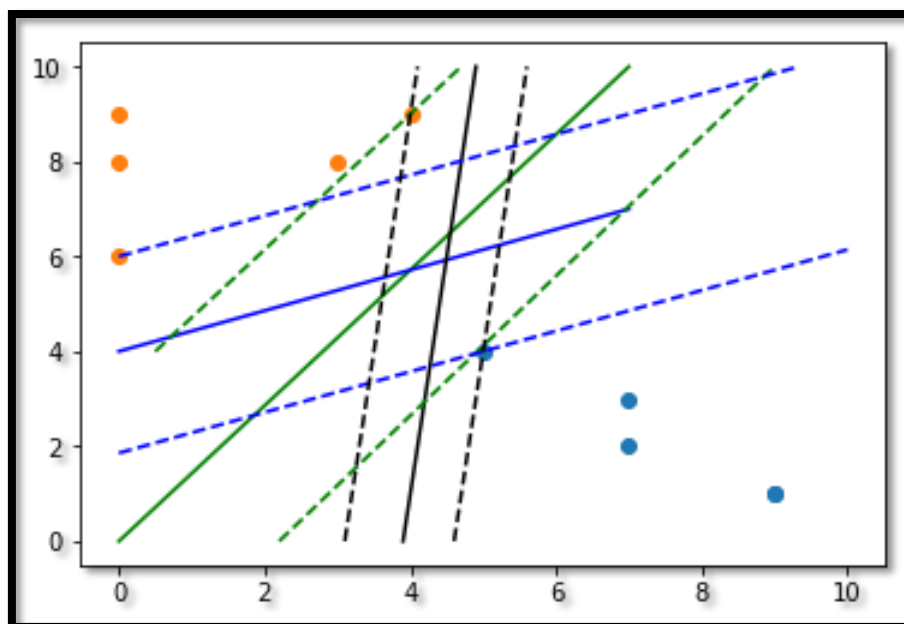


Figure 2.4: Margins surrounding the lines dividing the domain space.

The information for the Random Forests classifier is produced using IF/THEN-structured decision trees. The structure of the statements resembles a tree, with a conditional statement at the base and all other statements extending from it. As you make choices at each level, the comparisons at succeeding levels are modified to reflect your preferences. Each "leaf" [83] represents a comparison of two levels, whereas each "leaf" [84] represents either a prediction judgment or a comparison of two levels above.

The training dataset for a Random Forests classifier comprises as many partitions as there are trees in a forest. In the succeeding stage, a decision tree is independently generated for each dataset. It is possible that no two trees in the same forest would appear identical, but this is not a certainty. This method reduces reliance on a single measure value, resulting in more robust and accurate projections. Focusing on a single value rather than a set of values enhances precision when that value dominates the class; nevertheless, it fails to correctly identify the class when that value is unusual. The Random Forest employs a number of techniques to assign a category to incoming data. Each tree is assigned a single prediction, and the most popular of these forecasts is implemented. This technique generates multiple alternative forecasts for each category before selecting the best one [84], which increases processing time.

Due to their superior performance, particularly in data classification, neural network-based applications have garnered increased interest recently. Neural networks quantitatively represent the structure and connection of an individual's neural network, thereby demonstrating their signal-processing ability. Due to the fact that particular neurons may have weak or slack connections to other neurons, diverse inputs can activate different combinations of neurons [85].

A typical neural network often comprises numerous layers, with each layer containing a unique type of neuron. Each layer may exist in one of three states: input, output, or hidden. As its name suggests, the input layer is responsible for collecting data from the external environment, whereas the output layer is responsible for producing the desired outcome. Additional layers are routinely added between the hidden and output levels to improve the neural network's performance. We call these strata "hidden" because they are inaccessible from the exterior. As opposed to the output layer, the number of neurons in the input layer is governed by the number of inputs or the quality of each input. Even if additional neurons cannot be added to the input or output layers, increasing the total number of neurons can improve performance, as there are now more permutations of

neurons that can create the desired outcomes. These neurons are introduced into the subsurface layers as a result of [86].

In addition, a problem is more difficult when there are more subtleties that can be analyzed to get a more confident conclusion. However, as the number of nodes in a network or the number of its hidden layers increases, so do the required periods for analysis and processing. Before adding neurons and layers, it is essential to start with a simpler network to avoid increasing network complexity without increasing output. Because the activation properties of neurons in the second and subsequent hidden layers are more complex than those computed by the first hidden layer [87], the term "deep learning network" refers to a neural network with several hidden layers.

Weighted arrows connect neurons in one layer to those in the next, with a greater weight indicating a stronger connection. Each neuron in a neural network with complete connectivity adds the weighted outputs of the neurons in the layer below it. The neuron will enter this sum into an activation function. The output of a neuron can be transformed into a nonlinear function by employing a variety of application functions. Each neuron amplifies its inputs according to the weights it gets from neurons in the layer above. This increases the network's adaptability. During training, the weights of the network are modified using backpropagation based on the size of the error between the actual and desired values [88]. Numerous techniques exist for randomly determining the initial weight values of a neural network.

2.6.3. Reinforcement Learning

These machine learning algorithms employ an architecture consisting of an agent, environment, state, action, and reward [89, 91]. Figure 2.5 depicts how the agent's environment responds to his or her actions by outputting the agent's new condition and reward. Agents use the reward and the new state to choose their next action, which is frequently to produce the reward and the new state again from the environment. However, the agent is uninformed of the environmental consequences of its actions. An agent utilizing reinforcement learning duplicates the environment's behavior and its responses to the agent's actions [92, 93] in order to predict the action that would maximize rewards received from the environment.

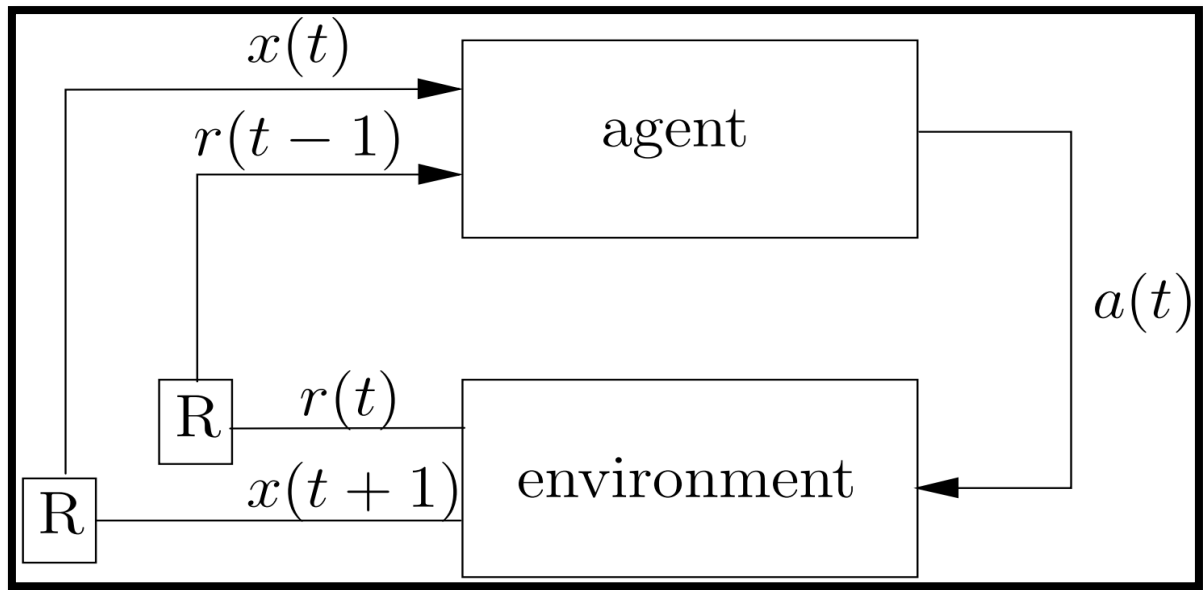


Figure 2.5: A reinforcement learning example of the interaction between the agent and the environment.

The main components in RL applications are defined as follows:

- a. **Agent:** The component of a system that analyzes gathered environmental data to decide the ideal course of action for completing a task, such as driving to a particular place or delivering goods using a drone..
- b. **Action (A):** identifies the range of options available to an agent and the projected rewards for each option under specified conditions. A self-driving vehicle can move forward, reverse, turn left or right, and stop without human intervention. RL agents are limited to only these behaviors; adding more would boost performance but complicate decisions.
- c. **Discount Factor:** To encourage the agent to prioritize global reward optimization over immediate reward, the highest reward from the new state the agent enters after performing an action is factored into the present reward calculation. The award value of the subsequent step is balanced between the instant reward and the overall reward by multiplying the award value by the discount factor. For example, if an autonomous vehicle's compensation is only dependent on instantaneous values, the agent may not slow down in potentially dangerous conditions if doing so would not result in the highest possible immediate reward. When the end benefits are incorporated in the calculations, the agent's capacity to evaluate the value of accident prevention improves, enabling them to make the best decisions. In addition, if the agent is exclusively motivated by the

outcome, they can be persuaded to engage in undesirable conduct, such as departing from the best course of action. Therefore, it is vital to identify the discount factor in order for the agent to consistently produce optimal results.

d. Environment: The domain in which the agent executes operations and is compensated. In the context of autonomous vehicles, "the environment" refers to the road and other vehicles on it.

e. State (S): The agent's current environment description, which could be provided to it in a variety of ways. For instance, a self-driving automobile must be aware of its surroundings, including the other vehicles on the road and any obstructions in its path.

f. Reward (R): Depicts how the environment responds to the agent's action. Activities with greater reward values are more pertinent to the situation, whilst those with lower reward values are less so. When the road is clear and the speed is low, the benefits of decelerating the vehicle may be little; but, when an oncoming vehicle that could cause an accident is present, the benefits may be substantially greater.

g. Policy (π): Under the given conditions, does the agent's technique for picking the action maximize reward.

h. Value (V): According to the policy, the agent's discount factor is applied to the current condition's long-term reward expectation $V(s)$. This value permits the agent to avoid making decisions that maximize short-term gain at the expense of long-term return. Driving over the speed limit, for example, can enhance the initial reward because more distance can be covered in less time, but the agent can make more responsible decisions if they also consider the chance of a fine or an accident.

i. Q-Value (Q): Q represents the entire reward for a particular activity at a given time (s, a). This parameter has a significant impact on the agents' decisions, resulting to the option that maximizes predicted utility.

Bellman's equation, which was developed by American mathematician Richard Bellman, serves as the foundation for this type of fortification. This equation can be used to determine each action's reward for a particular situation, taking into account both the immediate reward and all prizes won up until the end of the episode [93, 94]. This conclusion can occur when the agent achieves its objective or when a predetermined number of actions have been carried out. Equation 2.3 can be utilized to calculate this award.

$$Q^\pi(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma R_{t+3} + \dots | s_t, a_t] \quad 2.3$$

Equation 2.4 illustrates how the largest Q value can be multiplied by γ to obtain the Q value for each action that places the agent in a particular state.

$$new\ Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)] \quad 2.4$$

Q value for the selected current state action influences the trade-off between present and future profits, and there are instances where a lower discount rate is suitable due to greater levels of education. The new value of Q is then utilized to update the environment-representation function, which generates the exact reward for performing the action as opposed to a close approximation. This number, which represents the actual reward received from the environment, is applied to determine the reward values anticipated for preceding states.

2.7. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks rely on the constant back-and-forth of a vast number of nerve cells exchanging electrical signals in response to external data, similar to how the human brain performs arithmetic. The average human brain consists of 1011 linked biological neurons. As indicated in Figure 2.6, dendrites play a crucial role in the passage of modified electrical impulses from the cell body to the soma of neurons. A single dendrite can receive millions of inputs from several neurons or body sites, including the optic nerves in the eyes and the olfactory receptors in the nose. Changing the conductivity of the electrochemical solution at the proper synapses is sufficient to alter the effect of an input on the output of a cell. A cell determines whether an input is excitatory or inhibitory, with excitatory inputs increasing output and inhibitory inputs decreasing it. In conclusion, the output of one neuron can function as the input for another neuron or be converted into a decision or physical action in the human body [95, 97]. Consequently, the cell body regulates the effect of the electrical signal, also known as membranes, on output, whereas the synapse regulates the signal's magnitude.

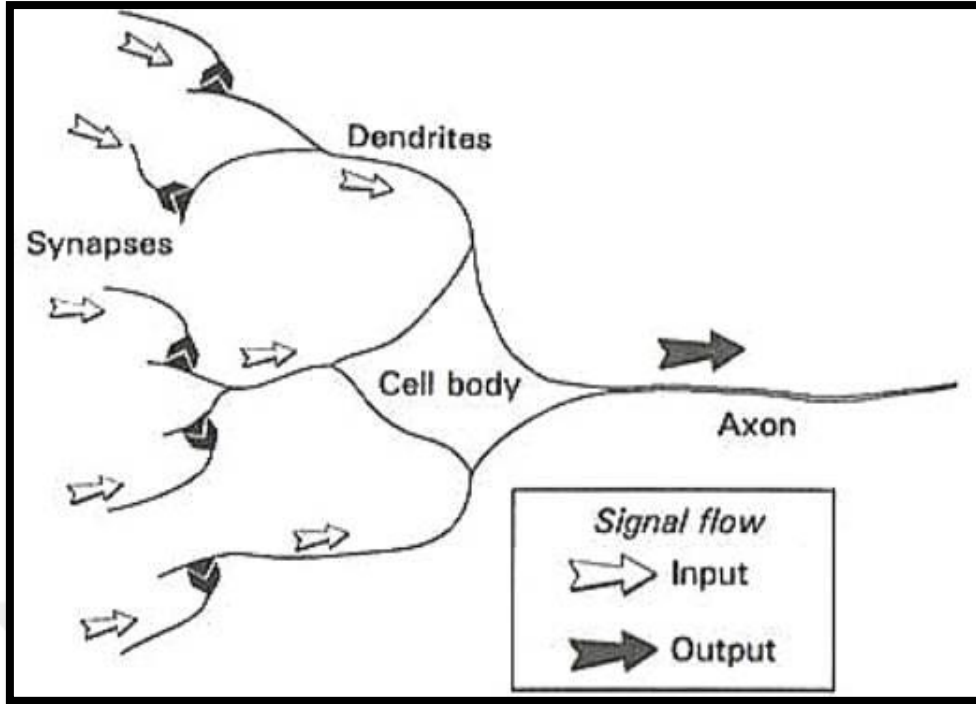


Figure 2.6: The Basic Elements of Neurons [95].

In order to statistically evaluate individual artificial neurons and the networks they form, the weight controls the proportional relevance of each input to the final output. The weight's magnitude effects the magnitude of the input, but the weight's sign defines the nature of the influence (i.e., whether it is excitatory or inhibitory) (i.e., whether it is excitatory or inhibitory). According to Equation 2.5, the output of a neuron is derived by adding the weighted inputs and their associated weight values, w . During adaptation, the neuron employs a bias value b [98] to boost its computational flexibility. The activation function of neurons, which is visually illustrated in Figure 2.7, is an additional essential feature.

$$S_j = \sum_i x_i w_{ij} + b_j \quad 2.5$$

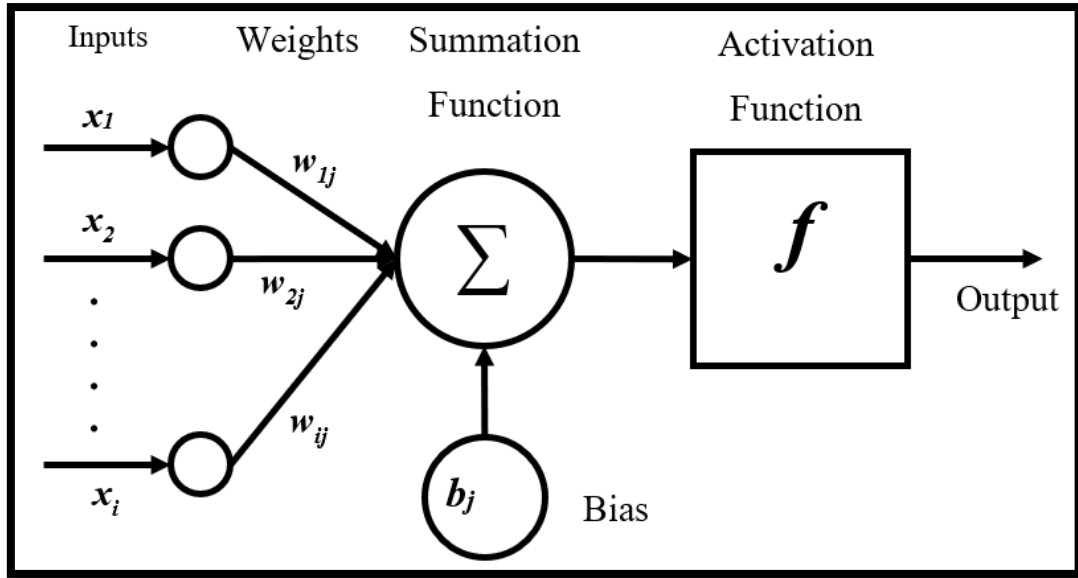


Figure 2.7: The Computations of the Neuron are Visualized [98].

As stated in Equation 2.5, the output of a neuron is linear with respect to its inputs when the weights and biases are held constant. The use of activation functions to the neuron output, following the application of a bias value and weighted inputs, enables neural networks to handle and overcome nonlinear difficulties. Nonlinearity or a different pattern in the incoming input enables the neuron to develop characteristics that are more readily identifiable. As depicted in Figure 2.8, several activation functions improve the performance of an artificial neural network. In comparison, the Hyperbolic Tangent (TanH) function output range is $[-1, 1]$. Activation functions are significant because they can restrict the range of input and output values that a neural network accepts and outputs. Rectified Linear Units (ReLUs) in hidden layers have significantly improved the performance of artificial neural networks across a broad range of applications [101, 102], and depending on the range of values required from the neural network, these functions are typically implemented in the output layer.

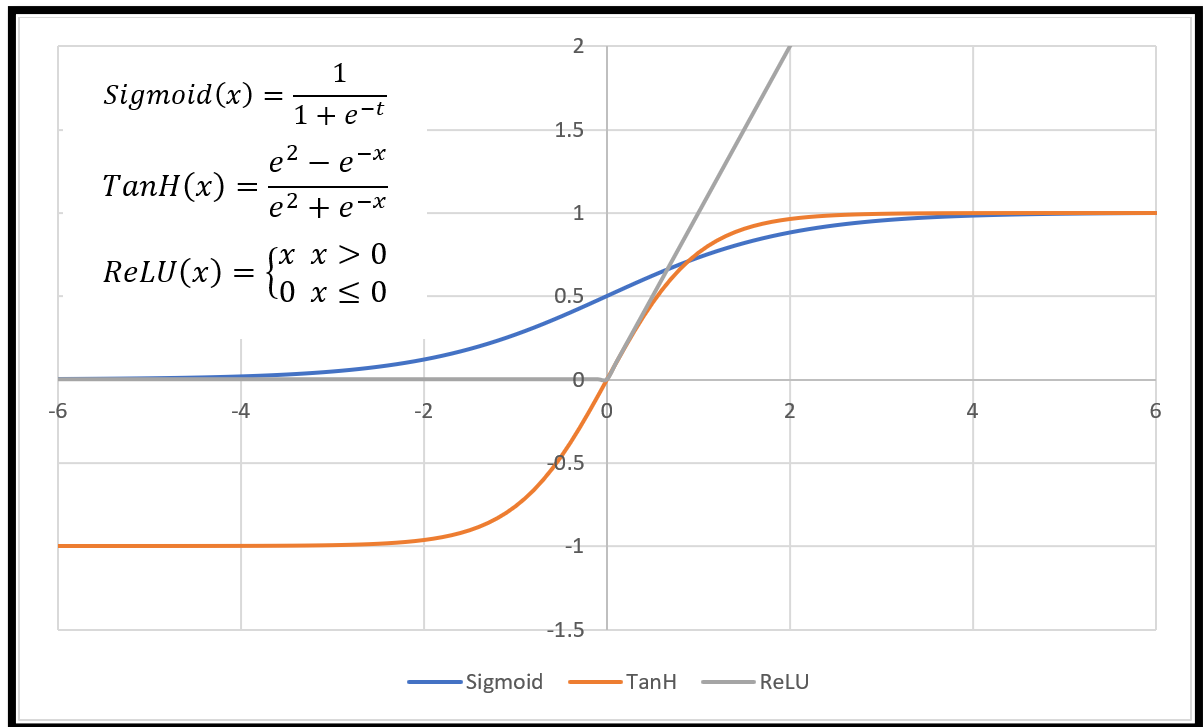


Figure 2.8: Neurons' Activation Function [101, 103].

Each layer of neurons in a typical neural network processes the data from the layer behind it. Due to this distribution, neurons are able to comprehend complex aspects by combining the qualities indicated in the layer below, which may be a mix of the qualities indicated in the layer below or the input itself. In addition to input layers that accept data from the environment and output layers that present computed values to the environment, every artificial neural network contains hidden layers that actively seek these features [104]. Figure 2.9 depicts a straightforward feedforward neural network with four inputs, two outputs (indicating the number of neurons in the input and output layers), and two hidden layers. Increasing the number of hidden layers in a neural network will improve the network's ability to recognize subtle patterns and characteristics. However, as the number of weights, biases, and neurons increases, the required computations get more complex. By increasing the number of neurons in the relevant layer, a neural network is able to identify a greater range of patterns while preserving its computational complexity. A balanced topology is necessary for identifying the characteristics that enable the neural network to do the needed task with the least amount of complexity.

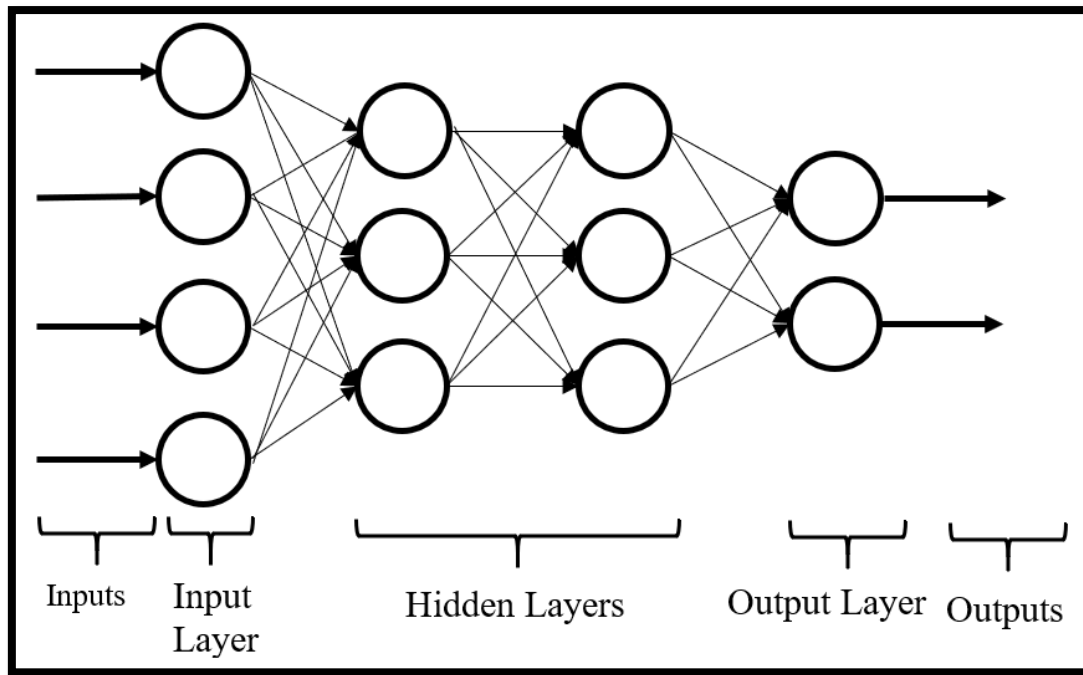


Figure 2.9: Examples of Artificial Neural Networks that Feed Forward.

In an artificial neural network, the weights and bias values of each neuron are utilized to compute the right network output based on the calculations performed by each neuron and the distribution of neurons. By adjusting these parameters, we can reduce the loss, or the difference between the neural network's outputs and the outputs required for each input. The purpose of optimizing neural networks is not limited to a particular optimization technique; however, rapid optimization processes are required due to the necessity to rapidly enhance a large number of parameters. As seen in Figure 2.10, gradient descent optimization requires a convex loss function. As part of this optimization approach, the partial derivative of each parameter with regard to the loss is computed to illustrate the loss's growth trend in relation to the parameter's value. A paradoxical optimization technique can reduce loss, defined as the gap between ideal and actual values. [105, 106] When used with non-convex loss functions, these optimization strategies have a tendency to stall at local minima.

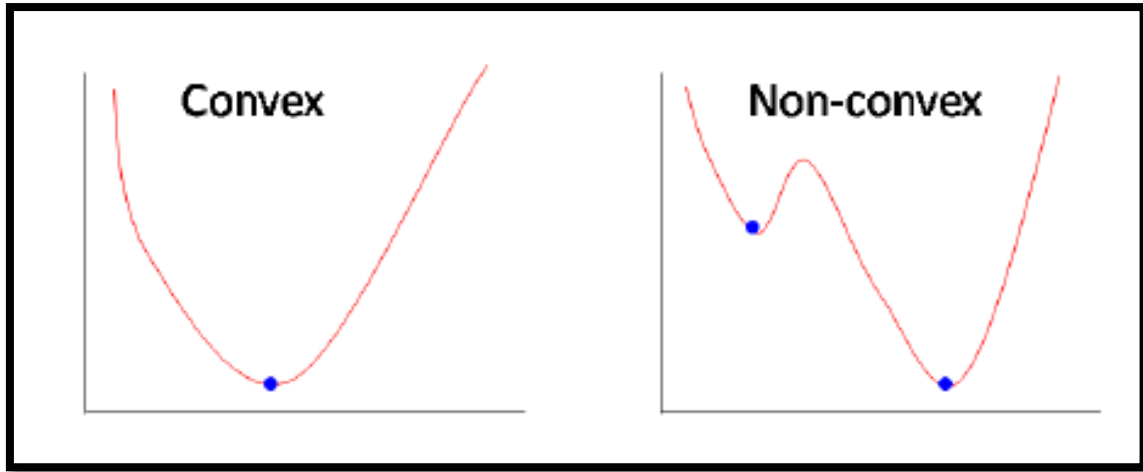


Figure 2.10: Function that is convex versus non-convex [107].

Usually, the Sum of Squared Errors (SSE) loss function is employed while training neural networks for regression problems. The network's neuronal layers process the input values and generate the output value, y . The neural network receives the expected and actual values for y as inputs, which are used to generate the loss function. Therefore, the neural network can only adjust its weights and biases to minimize loss and achieve the intended outcomes. [108].

$$L = \sum_i (y_i - \hat{y}_i)^2 \quad 2.6$$

During the training cycle, the forward pass computes the output of the network given the input, and the backpropagation phase adjusts the network's weights and biases. By updating the neural network's weights and biases in reverse, we can reduce the amount of needless calculations. Because the partial derivatives are derived using the neural network's output, the loss, this is the case. By recycling the computations of the weights and biases of the neurons in the output layer, it is possible to compute the partial derivatives of the weights of the final hidden layer with respect to the loss. When the bias of a neuron is not considered a fixed input, but rather a weight in addition to the other weights, its value can be modified to get the same outcomes [109].

Calculate w using Equation 2.7, where L is the loss and E is the learning rate. By removing w/L from the initial weight, where w/L indicates the gradient increase, loss is prevented and output homogeneity is ensured. Since the performance of the neural network worsens as the weight values decrease, the optimizer may miss the global minimum of the loss if the weight values are altered significantly. Consequently, this procedure is repeated until

a specified condition is met, such as when a predetermined loss amount is reached or a predetermined number of times has passed [110, 112].

$$\hat{w} = w - \frac{\partial w}{\partial L} \times E \times L \quad 2.7$$

2.8. DEEP REINFORCEMENT LEARNING

As demonstrated, the primary goal of reinforcement learning is for the agent to comprehend the reward value of each action at a particular state based on its observations of the environment. To calculate the reward, each action must be performed in every possible condition the agent could be in. Consequently, subsequent state action is determined by the course of action with the highest anticipated benefit. This method is unsuccessful, however, because the agent may exist in numerous states simultaneously. Calculating the predicted value of each action's reward employs an observation-accepting function as a stand-in. The agent continuously updates its estimation of the reward function, and then performs the action that it believes will yield the greatest reward in the environment. It is possible to improve the approximate function and future projections by integrating the actual compensation the agent receives for completing the task [113, 114].

Using artificial neural networks to approximate complex functions is becoming increasingly popular. The agent's actions and the resulting rewards are used to teach a neural network for use in future circumstances. This indicates that the neural network of the RL agent has as many potential actions as output neurons. On the basis of the neural network's suggestions, decisions are taken. Rather than the value predicted by a neural network, the value of the reward acquired from the environment is utilized. Then, these inputs are used to train the neural network to identify and correct faults [93, 115, 116]. Q values, which signify the quality of each action the agent can take in the environment, are used to represent the values produced by the neural network, giving rise to the moniker "Q-Learning" [117, 118] for this particular version of RL.

However, Dueling DQN (DDQN) has exhibited faster convergence of the neural network towards forecasting precise Q values. Deep Q-Network (DQN) can achieve satisfactory performance when used by the RL agent to predict Q values for each state. Figure 2.11 demonstrates that the DDQN divides the neural network in half and then unites the two parts. The first fork defines the normal salary for workers, whereas the second fork

assesses the advantages and disadvantages of alternative activities. This strategy beats DQN in terms of learning speed and convergence to the right values [119], as the amount of neurons in the output layer is inversely related to the expected actions made by the agent.

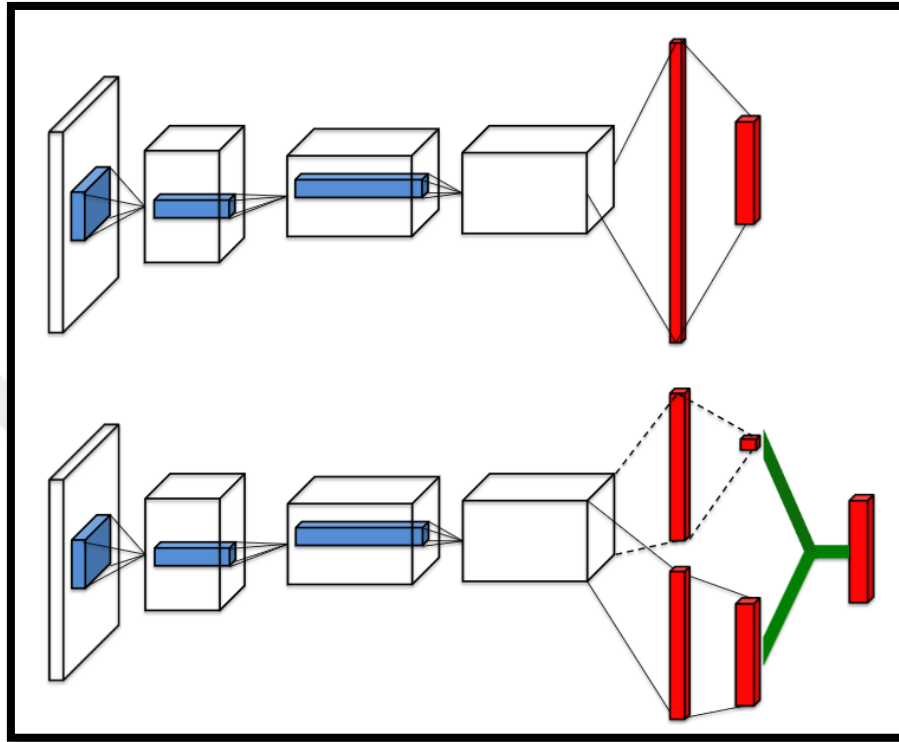


Figure 2.11: both DQN and DDQN DQN, from top to bottom; the upgraded DDQN [119].

2.9. RELATED WORKS

A number of implementation strategies for ML-based intrusion detection systems have been provided by the research. In classification techniques, also known as supervised learning, it is common practice to classify the packets in a training dataset as either attack or benign in order to find trends within each class. Based on the detected patterns, these classifiers subsequently make predictions regarding incoming packets. When these classifiers were applied to various intrusion datasets, the linkages between the feature values that define the packets and each category were obtained using distinct techniques, resulting in varying performance ratings.

Classification-based approaches utilize either patterns discovered during training [120-123] or, in the case of lazy classifiers [124, 125], patterns already existing in the training dataset. Using a range of artificial neural network types, this approach classifies the

incoming packet based on the values of its attributes. RNNs and CNNs are convolutional and recurrent neural networks, respectively [127, 128]. (CNNs). This sort of neural network is used to represent dynamic data because it can accommodate a variety of inputs. Since arriving packets are time-sensitive, this representation is required for intrusion detection. Numerous comparison studies [129, 130] have demonstrated that these networks perform similarly under a variety of situations. CNNs and RNNs, which are more sophisticated than Feed-Forward Neural Networks, require extensive training [131] in order to extract the key relations and attain the required performance. (FFNNs). Due to the time and effort required to train these networks, they cannot be utilized in scenarios requiring prompt responses to environmental changes [132].

The primary disadvantage of classification-based IDSs is that the classifier must be retrained whenever a new category of threats is added in order to keep the extracted knowledge current. If the attack generates packets with identical feature values to the benign packets required to train the classifier, the attack will be effective. In addition, until the classifier is trained and updated in real time, all attack packets can pass through the network, which can drastically reduce the network's quality of service (QoS).

Numerous IDSs circumvent this limitation by employing unsupervised ML techniques such as clustering [20]. Clustering splits data instances into comparable groups based on their feature values and does not require specialized knowledge; hence, anyone may perform it. ML techniques are utilized to design IDSs that can be updated rapidly during operation. By combining real-world instances of the recently reported assault into the clustering approach [133-135], it is able to categorize packets belonging to the same cluster as these samples as attacks without retraining. These algorithms can update the IDS more rapidly than classification-based ones, but they still require human intervention to acquire clustering data. As with classification-based approaches, attack packets can still access the network and impair its service quality until these samples are supplied.

In their article on page 24, Lopez-Martin et al. provide a comprehensive analysis and evaluation of the usage of RL in intrusion detection. Dueling Deep Q-Learning (DDQN) and reinforcement learning were used to get the optimum performance, according to the findings (RL). In both Deep Q-Learning and DDQN, the neural network is trained to predict the consequence of an action by analyzing the environment's response. Nonetheless, it has been demonstrated that DDQN arrives at the solution more quickly and exactly. Although this and other other IDSs [23, 136, 137] utilize reinforcement

learning, the reward values used to train the agent in classification-based techniques are indicative of the prediction's correctness. Consequently, there are no advantages to employing the RL approach.



3. METHODOLOGY

3.1. OVERVIEW

This paper explains a Quality of Service (QoS)-based method for protecting a network. The bulk of attacks on computers within a network try to degrade the performance of the network's services, making this tactic highly effective. The training incentive is determined based on the evaluation of the network's QoS. The service quality is determined by the performance and dependability of the connected devices and connections. One statistic used to measure the quality of a service is the amount of time it takes to react. The QoS value is used to reevaluate the decision-making process for each packet since reinforcement learning attempts to maximize reward from the environment. If the neural network senses a decrease in QoS, it will reevaluate all previous decisions and use the results to reinforce the less certain ones for the next time interval. Figure 3.1 depicts the organizational structure of the proposed IDS.

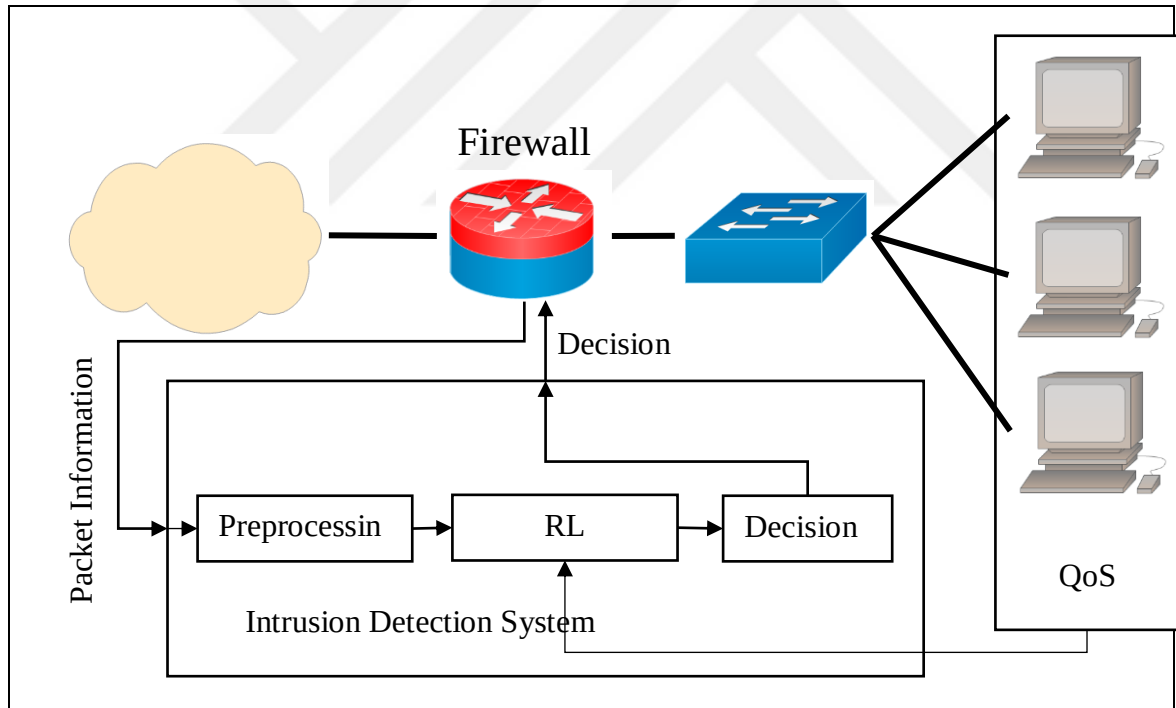


Figure 3.1: The proposed RL-based IDS's proposed hierarchy.

3.2. THE AGENT NEURAL NETWORK

Figure 3.2 depicts how a DDQN topology is generated for the agent in the proposed technique, leading to improved performance. Input layer packet descriptors include the same number of characteristics as the layer's neurons. The processing of data occurs in

two hidden layers: one to determine the value of the current state (V), and another to determine the consequences of each action on V. The V-Single layer concatenated the output of the hidden layer into a single number, whilst the A-Advantage layer computed the benefits of each action available in that state. The benefit values range from negative to zero to positive based on the extent to which doing the action can enhance the agent's state. The agent has the choice to permit access to the packet network. The result of the output layer is two numbers: the sum of the benefits and the value of state V minus the average of the benefits. Table 3.1 provides a full description of the proposed neural network's topology.

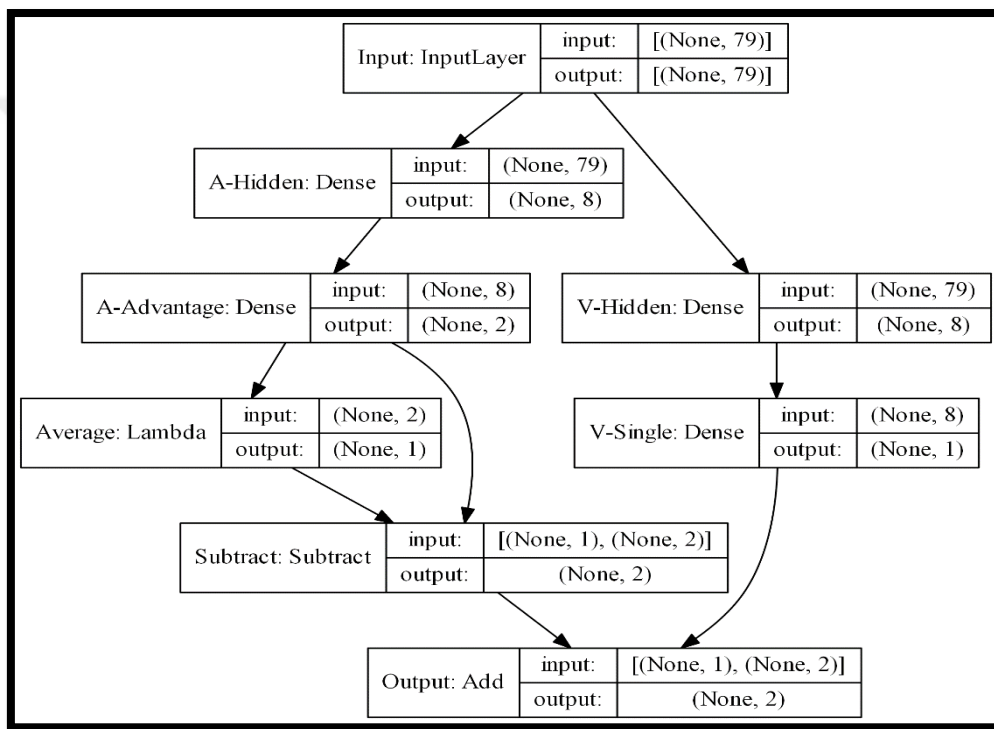


Figure 3.2: The design of the neural network used to power the RL agent.

Table 3.1: Structure of the neural network for the proposed method.

Layer (type)	Output Shape	Param #	Connected to
Input (InputLayer)	[(None, 79)]	0	
A-Hidden (Dense)	(None, 8)	640	Input[0][0]
A-Advantage (Dense)	(None, 2)	18	A-Hidden[0][0]
V-Hidden (Dense)	(None, 8)	640	Input[0][0]
Average (Lambda)	(None, 1)	0	A-Advantage[0][0]
V-Single (Dense)	(None, 1)	9	V-Hidden[0][0]
Subtract (Subtract)	(None, 2)	0	Average[0][0], A-Advantage[0][0]
Output (Add)	(None, 2)	0	V-Single[0][0], Subtract[0][0]
Total params: 1,307			
Trainable params: 1,307			
Non-trainable params: 0			

3.3. REWARD VALUE COMPUTATION AND NEURAL NETWORK TRAINING

In the proposed technique, the quality of service (QoS) metric is employed to determine the incentive for training the neural network that the agent would use to choose the optimal action. The quality of a service can be tested on the actual server providing the service or on a collection of servers across the network. In all instances, the suggested method requires Quality of Service (QoS) values between 0 and 1, with 1 indicating the highest attainable QoS. However, when measuring QoS from within the network, the suggested method raises substantial obstacles, as the agent might quickly learn to block all incoming traffic in an effort to enhance the perceived QoS. According to Equation 3.1, reward levels are proportional to the proportion of packets granted network access.

$$R = \alpha \cdot QoS - (1 - \alpha) \cdot \frac{|D|}{|T|} \quad (3.1)$$

where,

D is the percentage of total packets dropped by the network, T, whereas R is the incentive. adjusts the proportion of packets that are denied network access based on their expected R value and the overall impact of QoS.

The foregoing calculation illustrates that the maximum reward value of one is only achievable when QoS is enforced flawlessly and no packets are denied network access. Additionally, the algorithm ensures that the incentive amount is smaller than one if network connectivity is disallowed. By blocking only attack packets that have a major impact on quality of service, the RL agent is able to maintain a high reward value and stable operations.

Periodically, the RL agent, which maintains information on packets that have requested network access since the last QoS measurement, is updated on the network's current quality of service. R is the difference between these two numbers, which is calculated by subtracting the second number from the result. As demonstrated in Equation 3.2, this change is reflected in the neural network's prediction of the packet's subsequent stages.

$$\dot{Q}(s, a) = Q(s, a) + \Delta R \quad (3.2)$$

where,

The feature vector of the packet, represented by the letter s, indicates the current state of the agent, whereas the letter A denotes the action that was done.

The mismatch demonstrates that the neural network's forecast was incorrect. If a packet is assumed to be a normal packet and given network access, its quality of service (QoS) and R value will both decline. A second look at the option is now possible due to training the neural network with the value Q in light of the lower R. (s,a). To prevent the accumulation of small values in Equation 2 from becoming extreme values, the values of Q are clipped to the interval [-m, m] to keep the possibility to reconsider. This clipping was produced using the formula in Equation 3.

$$\hat{Q}(s, a) = \begin{cases} \dot{Q}(s, a), & -m < \dot{Q}(s, a) < m \\ m, & \dot{Q}(s, a) \geq m \\ -m, & \dot{Q}(s, a) \leq -m \end{cases} \quad (3.3)$$

3.4. ACTION SELECTION

The R value of the QoS and the percentage of missed packets are utilized to reevaluate the RL agent's decisions in the proposed method. As seen by Equation 2, other packets are taken into account when calculating the value of R for a single packet. The resulting shifts in predicted values are rather minor. If you apply Eq. 3 or avoid extremely big neural network predictions, you can maintain a pretty steady relationship between forecasts and updated values. Due to the fact that the R value is determined for so many packets, these updates may differ significantly from what the neural network predicted. Since the expected consequences of both acts are comparable, the agent cannot "attempt" a new strategy if the projected outcomes fall within a given range. By evaluating the alternative, the agent can determine if it enhances performance in the given scenario.

The proposed method chooses a solution strategy according to the probabilistic distribution of outcomes. Given that Equation 4 demonstrates that the total of any two soft-max outputs cannot exceed one, it makes logical to use this function to determine the likelihood of each potential action. In light of these odds, the next step is to determine what to do. As a result, the agent can make course modifications more rapidly in order to evaluate their effectiveness. If this operation enhances the network's performance, R will become positive and the succeeding predictions will be more accurate than those produced prior to the activity. Therefore, it is likely that you will travel in that direction.

$$P_a = \frac{e^{Q(s,a)}}{e^{Q(s,a)} + e^{Q(s,b)}} \quad (4)$$

where $Q(s, b)$ amount that the neural network anticipated the other action to be at that state.

4. EXPERIMENTAL RESULTS

Python implements and tests the proposed method. Tensorflow trains the neural network. Development and evaluation use the Canadian Institute for Cyber Security's CICIDS2017 dataset. This archive's files were collected on many Monday–Friday weekdays. Each file lists packet types.

Table 4.1: A detailed analysis of this dataset can be found in [138].

Day	Packet Type	Count	Percentage (%)
Monday	Benign	529918	100
Tuesday	Benign	432074	96.90
	FTP-Patator	7938	1.78
	SSH-Patator	5897	1.32
Wednesday	Benign	440031	63.52
	DoS GoldenEye	10293	1.49
	DoS Hulk	231073	33.36
	DoS Slowhttptest	5499	0.79
	DoS slowloris	5796	0.84
	Heartbleed	11	0.01
Thursday	Benign	168186	98.72
	Web Attack / Brute Force	1507	0.88
	Web Attack / Sql Injection	21	0.01
	Web Attack /XSS	652	0.38
Friday	Benign	702888	70.87
	DDoS	128027	12.91
	PortScan	158930	16.02
	Bot	1966	0.20
	Infiltration	36	0.01

On Mondays and Tuesdays, the sample is dominated by packets from genuine traffic, while on Wednesdays and Fridays, the majority of attacks take place. Because the

suggested approach for recognizing valid traffic and permitting network access may be trained on data from the first two days of traffic, the distribution of attacks had a crucial influence in choosing this dataset for analysis. The success of the proposed technique in preventing new attacks while maintaining network quality of service can then be tested over the next few days. This study assumes that quality of service is affected by all network risks.

The same approaches are used to evaluate and train staff personnel. Using the approach given in Section 3.4, the next steps to be taken with each packet are determined. The QoS value, feature vector, and action done are all kept in a database once 100 packets have been sampled up until the packet is received. Even in the portion where the simulated runtime of the operation is utilized, this figure is used to train the neural network. This training allows the proposed technique to adapt to altering packet patterns induced by novel forms of attack.

False Positive (FP) and False Negative (FN) rates during the lifespan of a network are utilized to demonstrate the efficacy of the proposed technique. A false positive (FP) indicator is displayed when an attack packet is allowed into the network, while a false negative (FN) shows that a benign packet was denied access. As these figures decrease, the IDS performs better. To assist examination, the moving average of the thousand most recent data points for each performance parameter is computed. Since there are so many packets in the dataset, smoothing the graphic with moving averages can help the reader comprehend the performance.

First, the proposed procedure's effectiveness is evaluated in light of the parameter. Figure 4.1 illustrates the effectiveness of the proposed technique by demonstrating that the confusion matrix is minimized when this parameter is set to 0.3. These results, which are displayed in Table 4.2, suggest a 64.83 percent success rate for the proposed strategy. In addition, they reveal that the great majority of packets are accepted because they look to be valid. Due to its low value, the proposed technique must prioritize packet access above guaranteeing excellent QoS, resulting in the observed behavior. Therefore, the RL agent strongly benefits from the fact that the vast majority of attack packets are permitted to enter the network. As shown in Figure 4.2, the proposed method allows nearly all packets to reach the network even when an assault is in progress.

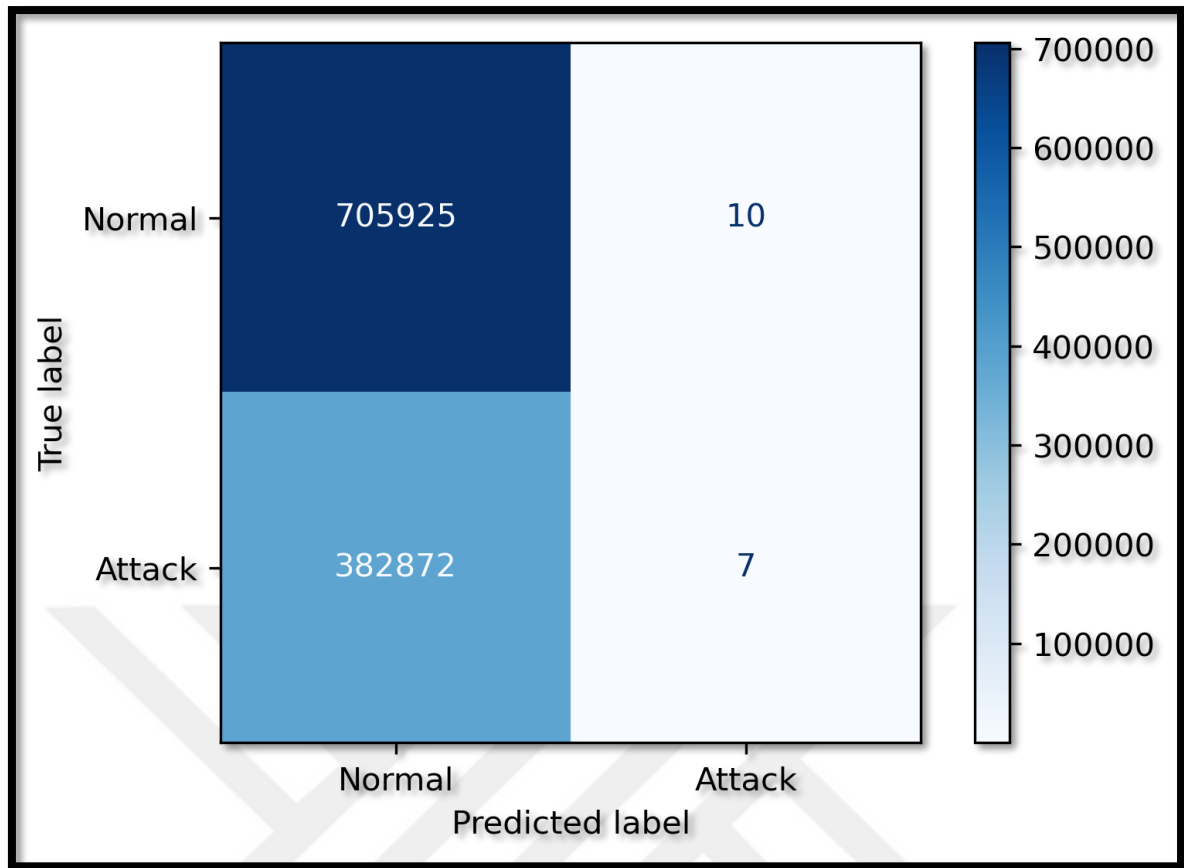


Figure 4.1: Confusion matrix of the predictions of the proposed method when $\alpha = 0.3$.

Table 4.2: Performance summary of the proposed method when $\alpha = 0.3$.

	Precision	Recall	F1-Score	Support
Normal	0.6484	1	0.7867	705935
Attack	0.4118	0	0	382879
Accuracy			0.6483	1088814
Macro avg	0.5301	0.5	0.3933	1088814
Weighted avg	0.5652	0.6483	0.51	1088814

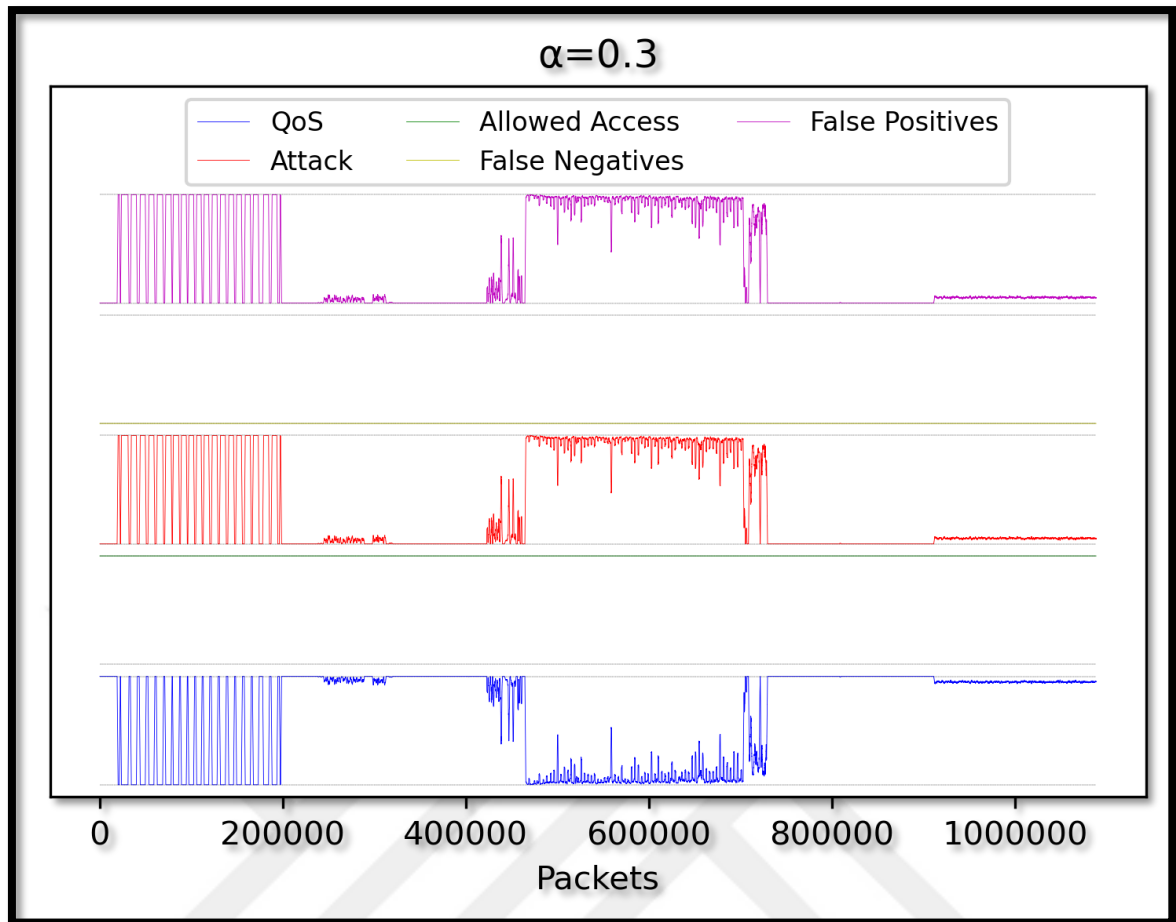


Figure 4.2: Response of the proposed IDS to the network attacks when $\alpha=0.3$.

In order to observe the confusion matrix depicted in Figure 4.3, we will assume a value of 0.5 for the next experiment. These findings indicate that the proposed approach for this value of produced an accuracy of 96.72%, which is much higher than the accuracy at $\alpha=0.3$. Figure 4.4 depicts how the network responds when the proposed method is applied with a value of, demonstrating how it responds more quickly to attacks while they are being carried out against the network by blocking their access to the network and more quickly to resuming normal operations after the attack has ceased. Not only has the precision improved, but this is also evident.

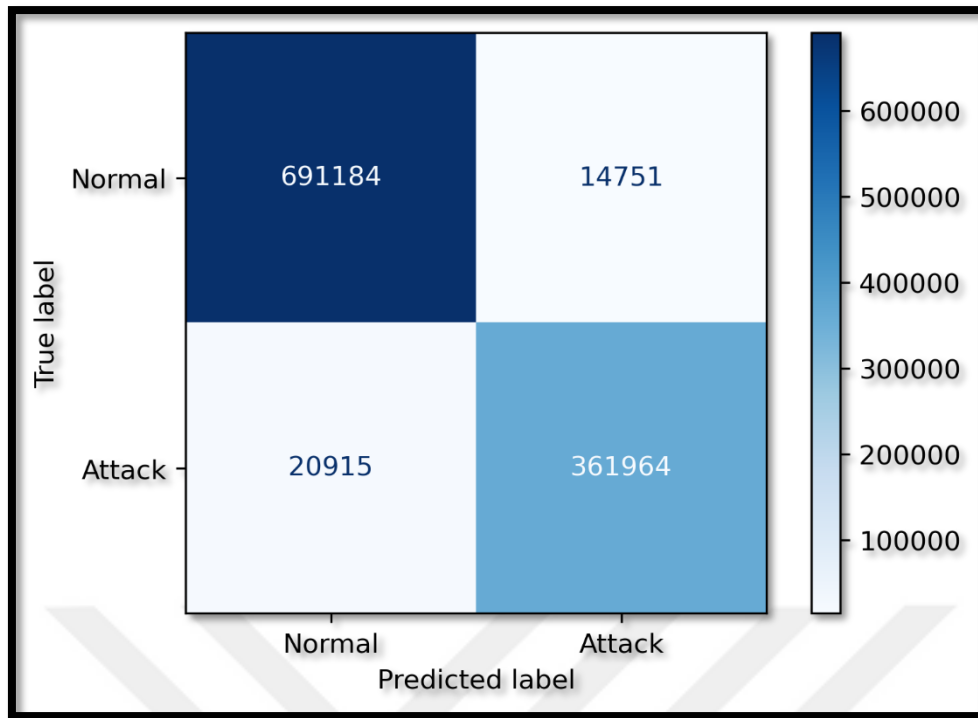


Figure 4.3: Confusion matrix of the predictions of the proposed method when $\alpha = 0.5$.

Table 4.3: Performance summary of the proposed method when $\alpha = 0.5$.

	Precision	Recall	F1-Score	Support
Normal	0.9706	0.9791	0.9748	705935
Attack	0.9608	0.9454	0.953	382879
Accuracy			0.9672	1088814
Macro avg	0.9657	0.9622	0.9639	1088814
Weighted avg	0.9672	0.9672	0.9672	1088814

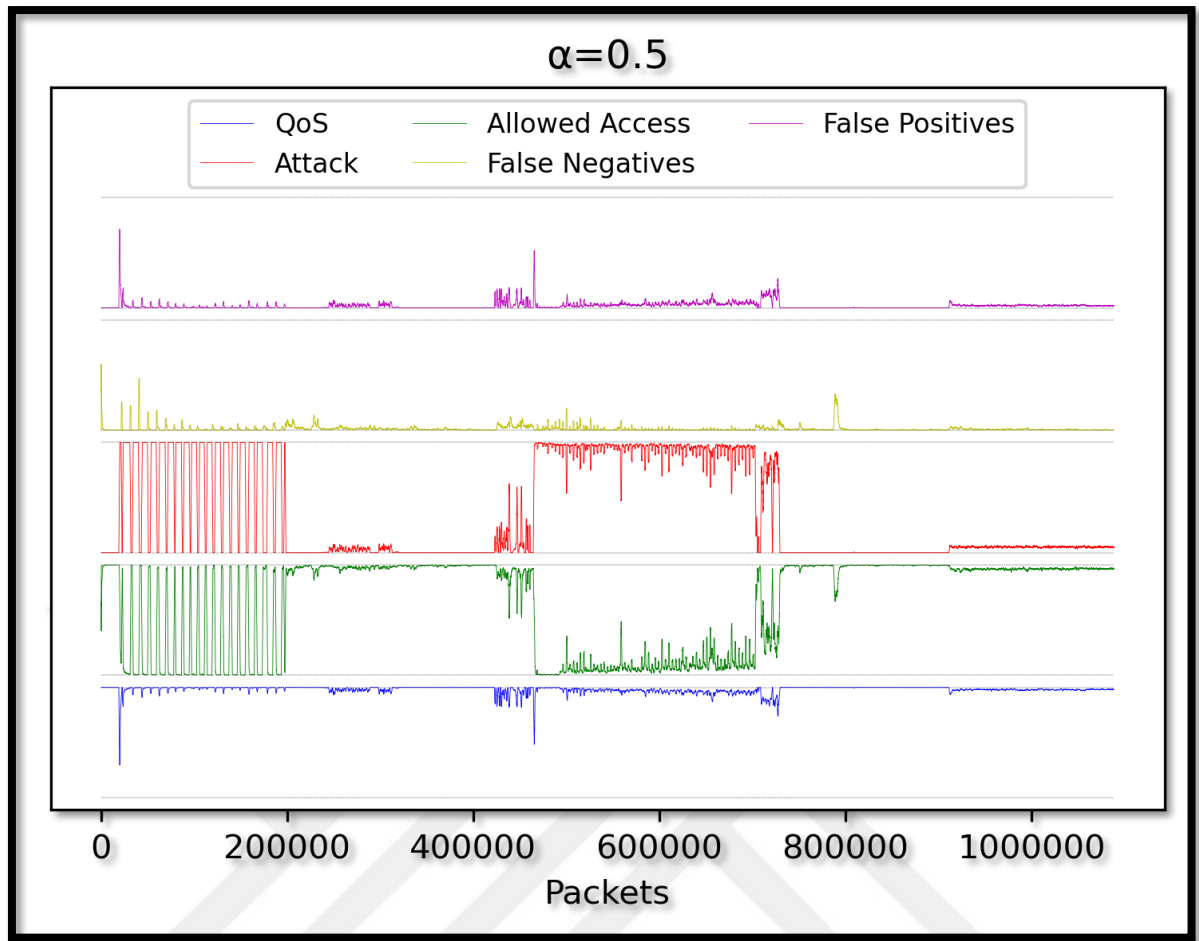


Figure 4.4: Response of the proposed IDS to the network attacks when $\alpha=0.5$.

The proposed approach works well with a value of 0.5; however, it is vital to study the possibility of further improvements if it is increased. Setting this parameter to 0.7 and watching the outcomes in Figure 4.5 permits us to assess the proposed method. As evidenced by these outcomes, the proposed approach with this value of achieved an overall accuracy of 35.17 percent. Table 4.2 also includes, for your convenience, measurements of the method's efficacy at this value. These results demonstrate that the proposed approach has often allowed attack packets to pass through, despite the fact that the RL agent incurs greater penalties when network access is denied (as indicated by the lower value of). Figure 4.6 demonstrates the suggested method's response to QoS changes caused by the introduction of new network attacks. Since the suggested strategy at this value of prioritizes QoS by preventing any packet from accessing the network (as depicted in Figure 4.6), the vast majority of traffic has been prevented.

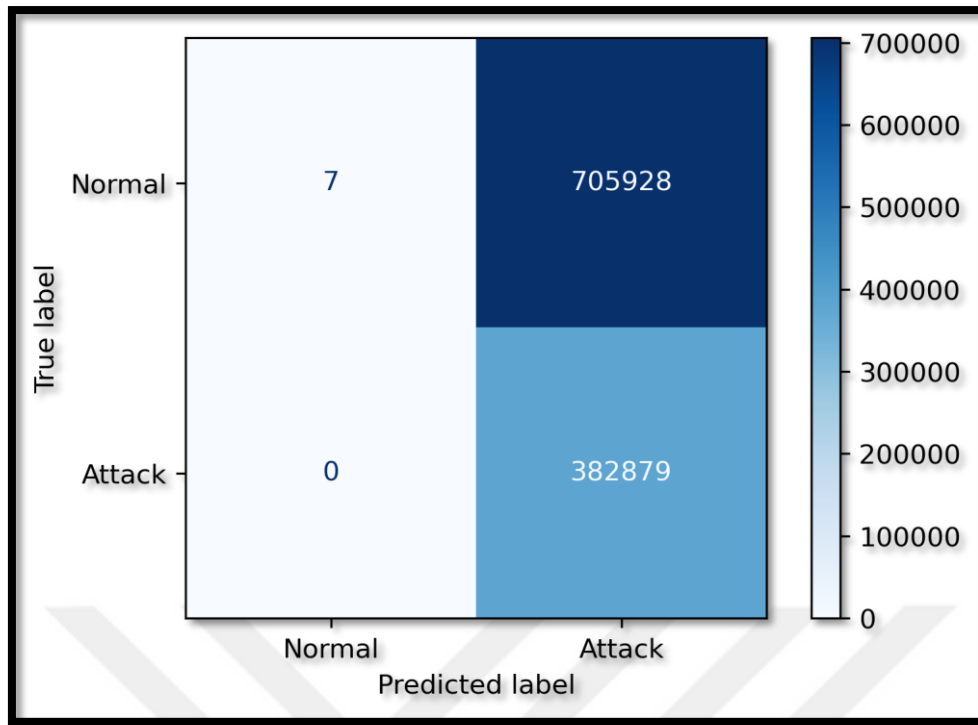


Figure 4.5: Confusion matrix of the predictions of the proposed method when $\alpha = 0.7$.

Table 4.4: Performance summary of the proposed method when $\alpha = 0.7$.

	Precision	Recall	F1-Score	Support
Normal	1	0	0	705935
Attack	0.3517	1	0.5203	382879
Accuracy			0.3517	1088814
Macro avg	0.6758	0.5	0.2602	1088814
Weighted avg	0.772	0.3517	0.183	1088814

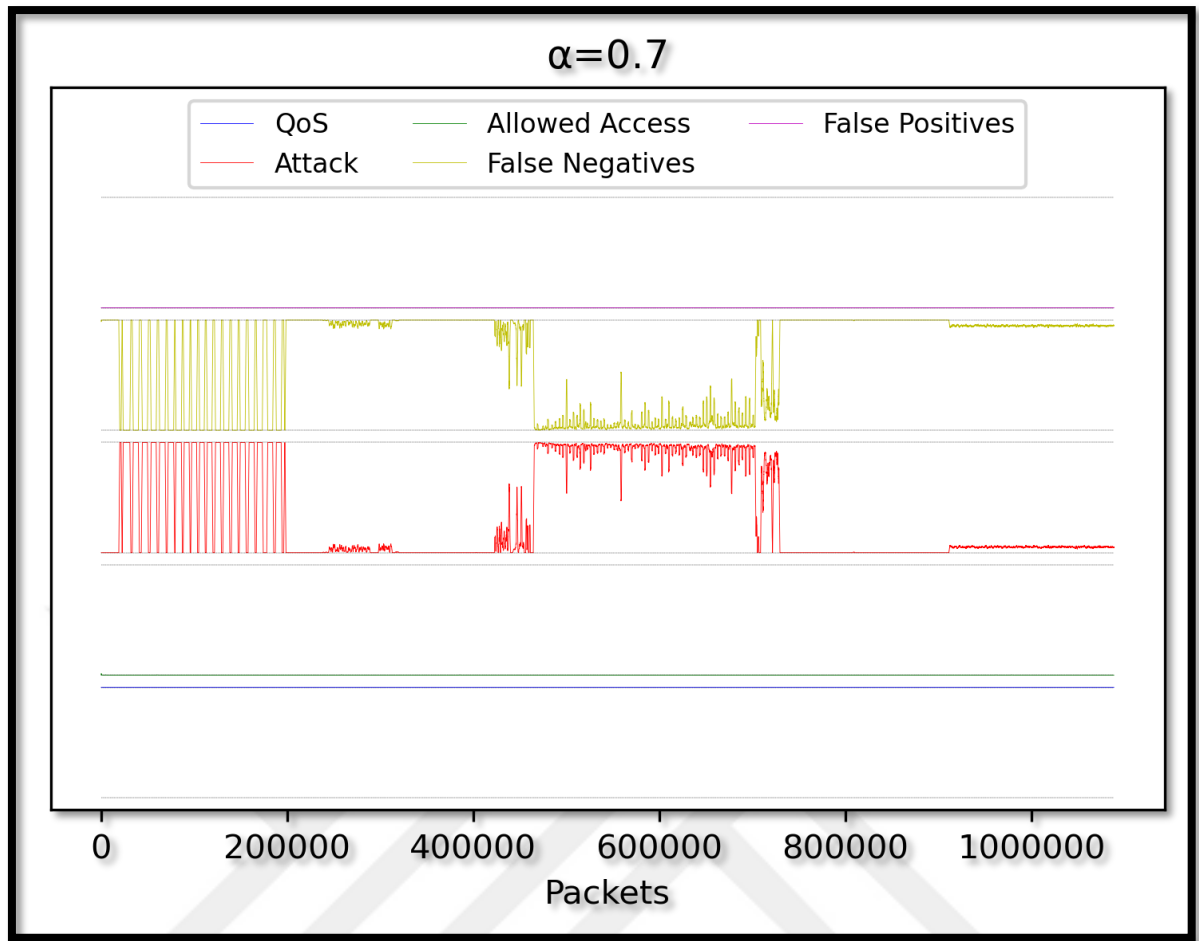


Figure 4.6: Response of the proposed IDS to the network attacks when $\alpha=0.5$.

The usefulness of the suggested method is also impacted by the parameter m , which represents the upper training range limit for the neural network. Due to the sensitivity of the methodology to fluctuations in action output values, increasing this parameter gives the neural network more latitude to select the optimal action given the circumstances. In previous studies, m was always assigned the value 5. As seen in Figure 4.7, the response time of the neural network rose dramatically when this number was increased to 10. Even after the network stopped receiving assault packets after the 500,000th packet, the neural network continued to predict assault packets. Even with a good Quality of Service, FN rates are high until just before the millionth packet, after which the results begin to deviate from expectations. Numerous authorized users have their network access revoked as a result of such activity. Based on the results of this experiment and the confusion matrix provided in Figure 4.8, Table 4.4 demonstrates that increasing the value of m decreases the performance of the suggested technique, which lowers to 74.28% accuracy.

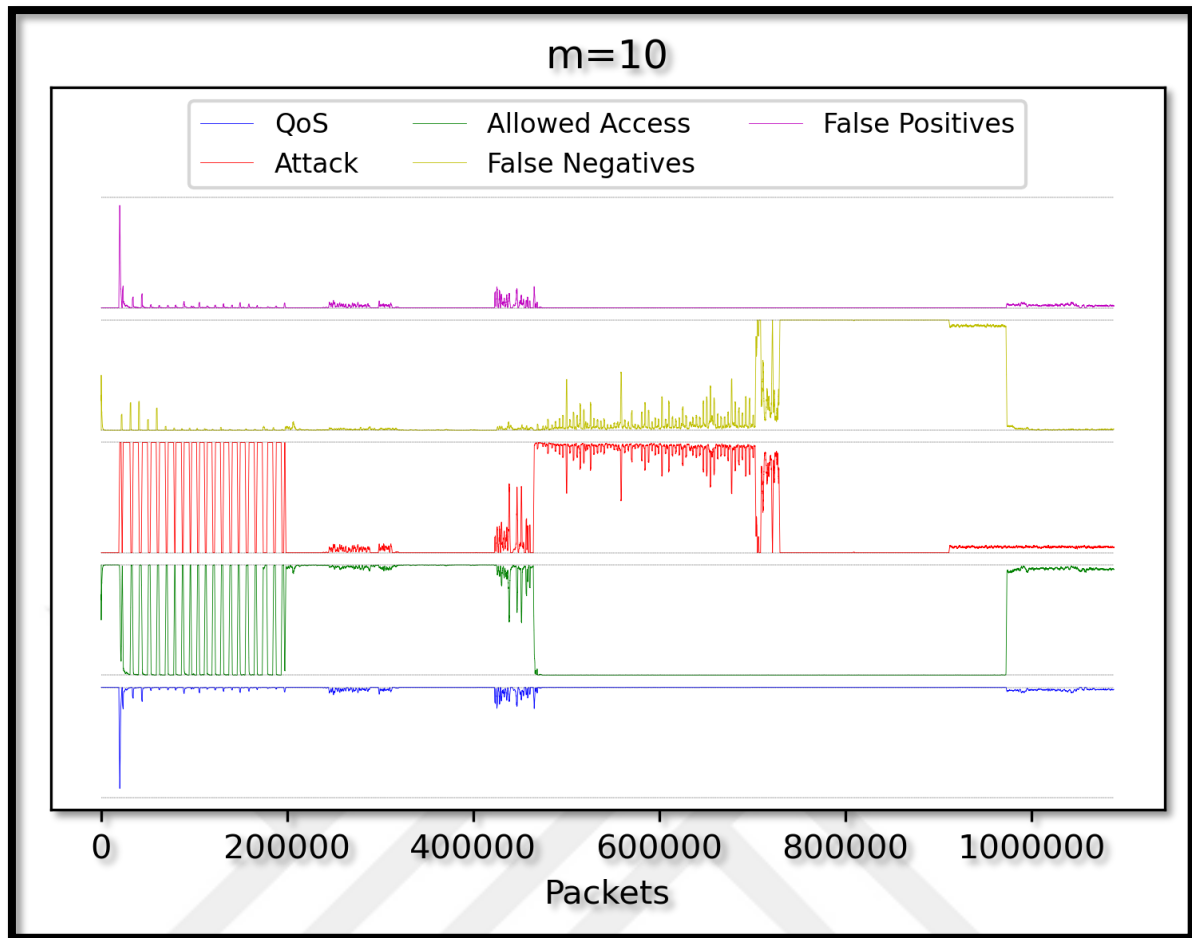


Figure 4.7: Performance of the suggested method with a parameter m value of 10.

Table 4.5: Performance summary of the proposed method when $m=10$.

	Precision	Recall	F1-Score	Support
Normal	0.9798	0.616	0.7564	705935
Attack	0.5797	0.9766	0.7276	382879
Accuracy			0.7428	1088814
Macro avg	0.7798	0.7963	0.742	1088814
Weighted avg	0.8391	0.7428	0.7463	1088814

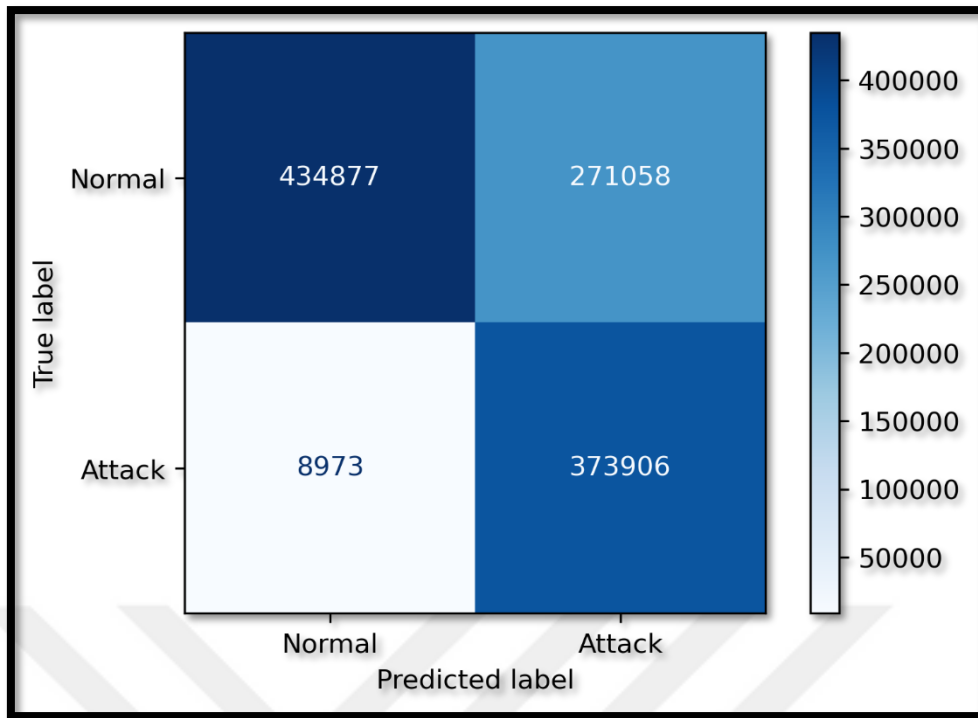


Figure 4.8: Confusion matrix of the predictions of the proposed method when $m=10$.

This value is decreased until the proposed approach achieves the desired outcomes, demonstrating behavior that strikes a balance between making the appropriate selections and fast adapting to new conditions. The system's behavior at $m = 5$ is represented in Figure 4.9. Even if both FN and FP are present throughout the operation of the suggested method for $m=5$, the balance of these values results in a more acceptable level of performance. Contrary to what is portrayed in Figure 4.8, the QoS has been maintained at a high level despite fluctuations in its value, especially when attack packets are received by the network. When m is set to 5, the trade-off between an unavailable flawless service (perfect QoS) and a somewhat compromised service that is nevertheless accessible to the majority of legitimate consumers is significantly improved.

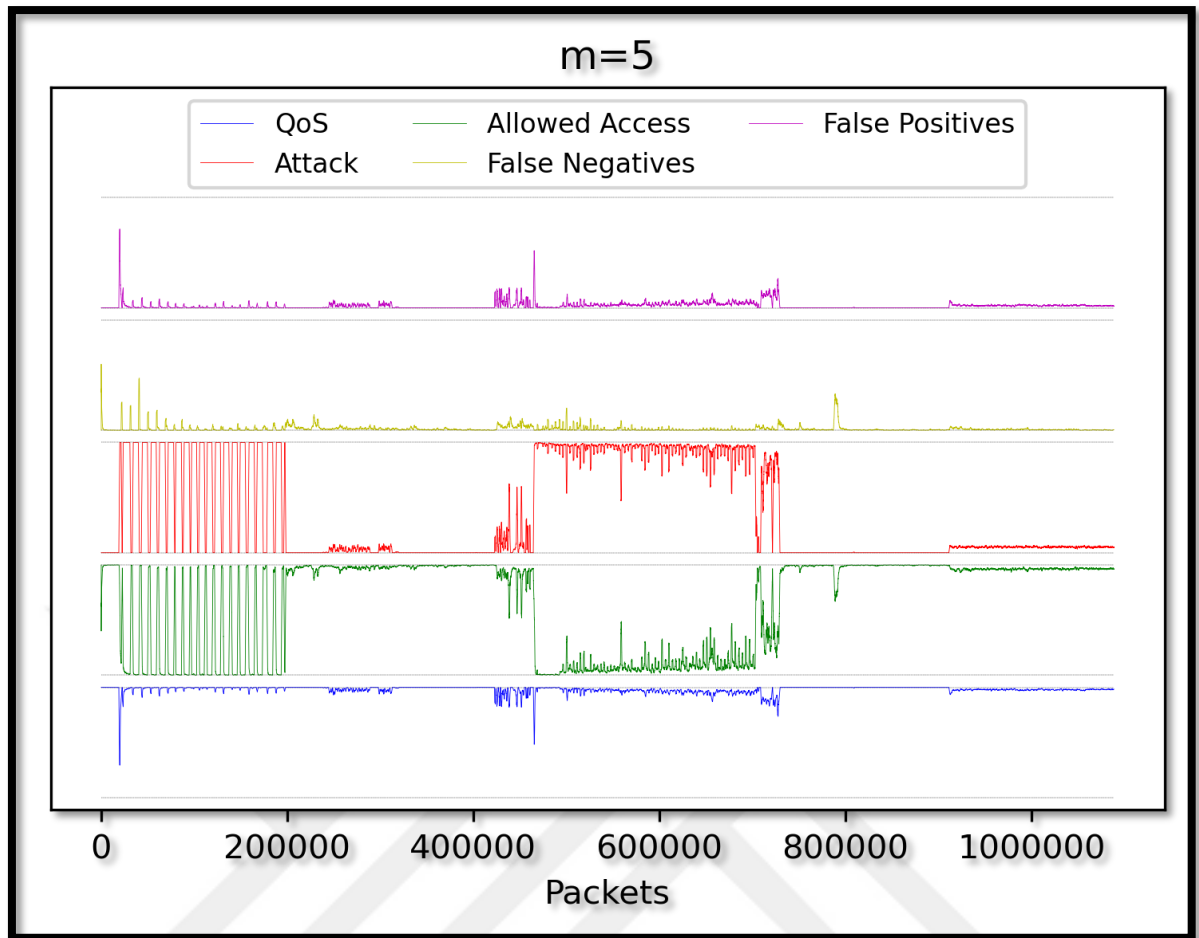


Figure 4.9: performance of the suggested technique with $m=5$.

Utilizing [123]'s proposed structure for a deep neural network, the suggested method is implemented successfully. This method was chosen to represent the cutting edge of classification-based strategies. Figure 4.10 illustrates the approach's inability to recognize any of the newly launched network attacks, which corresponds to Table 4.5's poor accuracy. Figure 4.11 demonstrates that, as a result, this strategy is unable to differentiate between training-coverage-only and other forms of attacks. On Wednesday and Friday, attacks on FTP-Patator and SSH-Patator are more comparable to benign traffic than they are on Monday and Tuesday, as seen in Table 4.1.

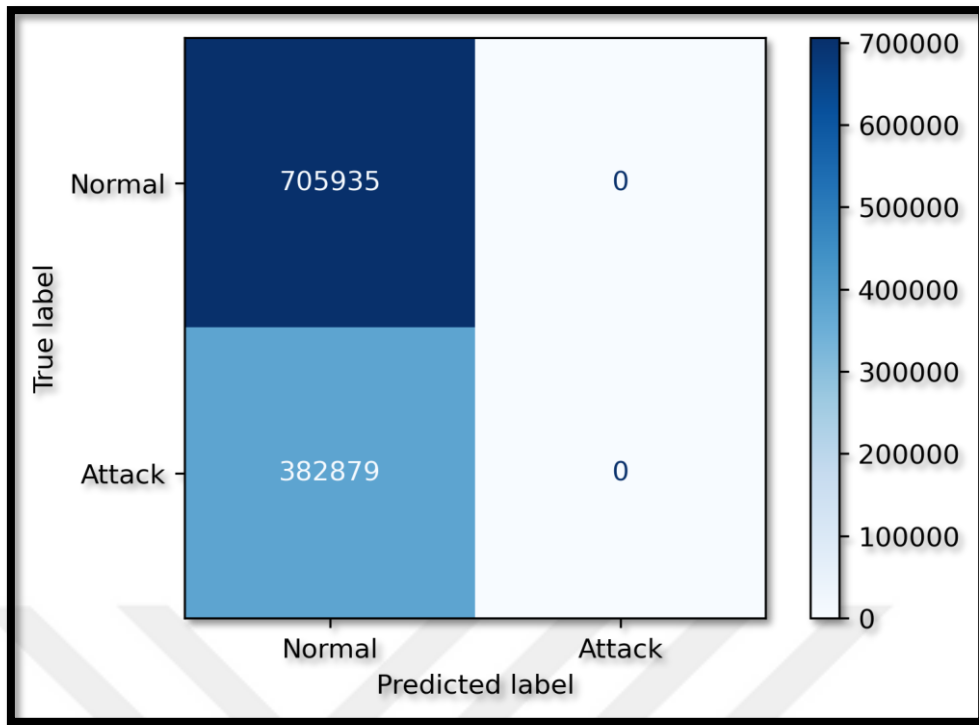


Figure 4.10: Confusion matrix of the predictions collected from the classification-based neural network.

Table 4.6: Performance measures of the classification-based neural network.

	Precision	Recall	F1-Score	Support
Normal	0.6484	1	0.7867	705935
Attack	0	0	0	382879
Accuracy			0.6484	1088814
Macro avg	0.3242	0.5	0.3933	1088814
Weighted avg	0.4204	0.6484	0.51	1088814

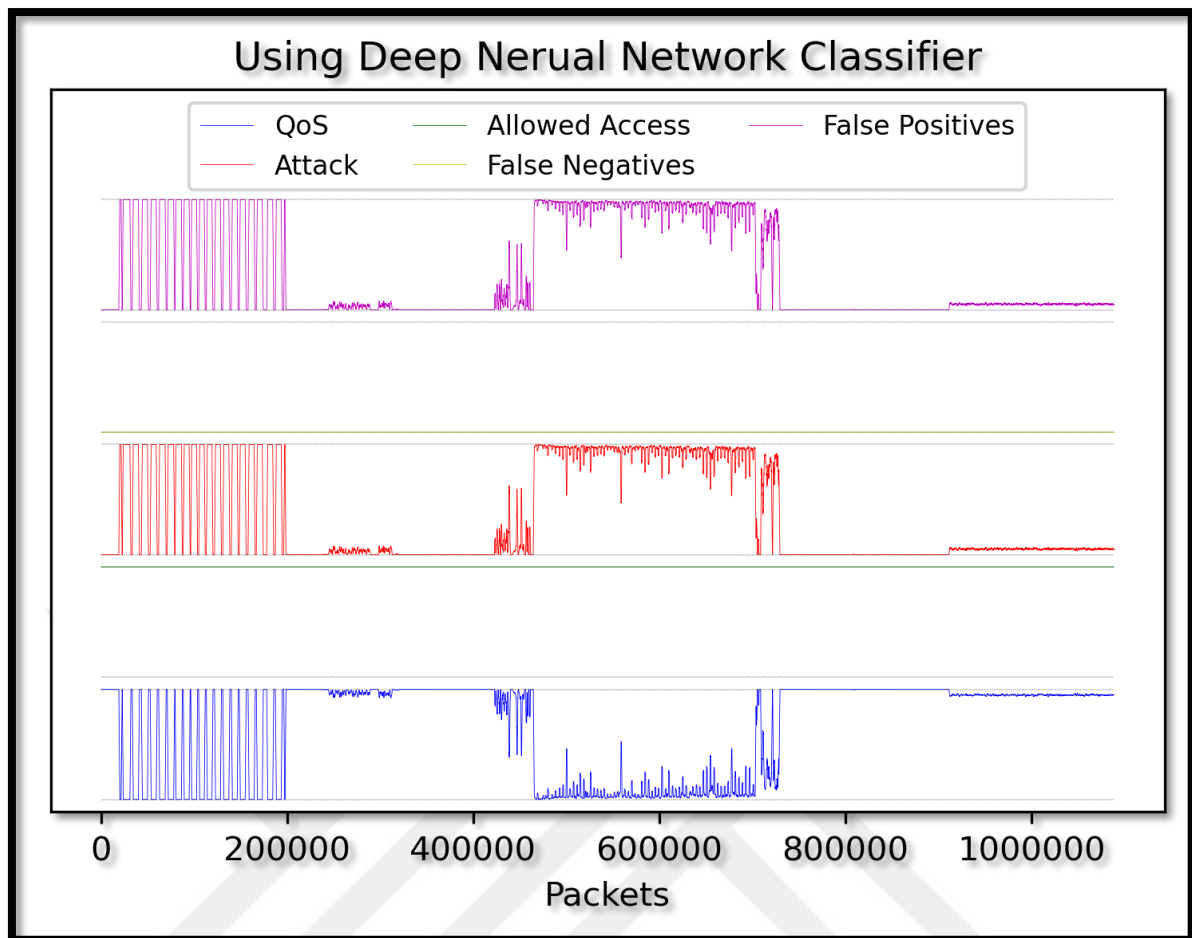


Figure 4.11: The effectiveness of the classification-based deep neural network method [123].

5. DISCUSSION

As shown in Figure 5.1, the value has been set to 0.5, giving the suggested technique a moderate amount of input from both the packet access ratio and the quality of service. Given these characteristics, the proposed method has demonstrated the highest degree of precision. These results demonstrate not only the great accuracy attained by the proposed method, but also its capacity to automatically recognize and restrict access to any form of assault, regardless of whether these attacks were included in the neural network's training or not. In contrast, the system's quality of service (QoS) is worsened as a result of the suggested method's successful analysis of incoming data and identification of patterns in the packets' characteristics.

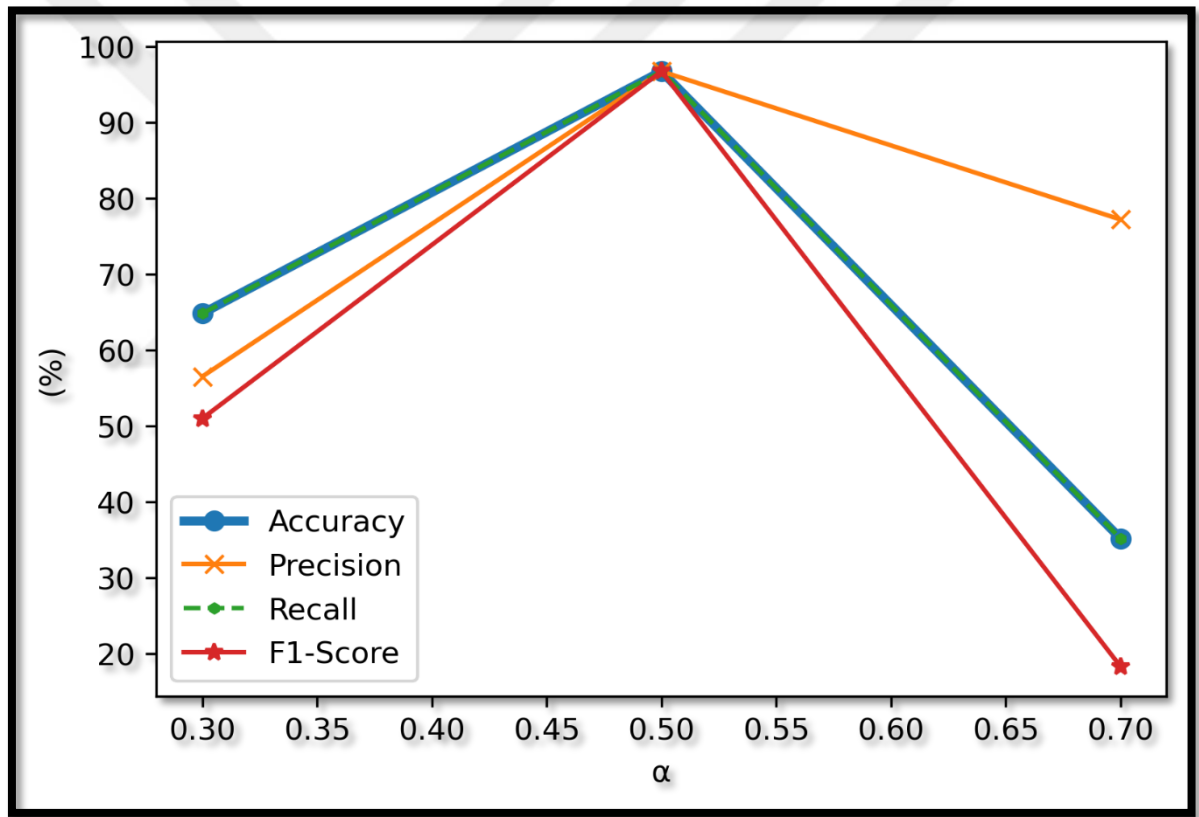


Figure 5.1: Performance summary of the proposed method at different values of the parameter α with $m=5$.

These results indicate not only RL's adaptation to new situations, but also the adaptability of the suggested approach to deal with ever-changing intruder techniques for degrading network quality of service (QoS). In contrast to previous forms of ML, RL may

continuously evaluate its conclusions in light of fresh information and alter its method accordingly. These findings are consistent with those of prior investigations [139-142] into the deployment of RL in various contexts. Therefore, the suggested method analyses patterns that, when eliminated, increase the network's QoS whenever its QoS declines. Additionally, the ratio of blocked traffic displays the capacity to counteract such behavior, ensuring that all users continue to have access.

To further demonstrate the capacity of the suggested method to adapt to the changing characteristics of packets received from assaults, a neural network similar to the one proposed in [123] is also developed and trained using the same training set as the proposed method. As illustrated in Table 5.1, this method has not been able to identify any of the new attacks that will be added to the network in the following days, i.e. attacks that are not included in the training. This behavior is shown by the fact that the classification-based neural network proposed in [123] achieved a recall and accuracy of 0 for the attack class, suggesting that none of the cases it classified were properly predicted. Figure 4.10 depicts a confusion matrix which, despite the neural network's outstanding precision during training, demonstrates that none of these occurrences constitute assaults. This also illustrates how innovative attacks masquerade their traffic as normal traffic by employing identical attribute values; however, the classifier has never experienced these attacks before, thus it has categorized the packets as normal..

Table 5.1: Comparison between. RL-based and classification-based neural networks in IDS.

	Proposed Method			Classification-Based [123]		
	Benign	Attack	Overall	Benign	Attack	Overall
Precision	0.97	0.96	0.97	0.65	0.00	0.42
Recall	0.98	0.95	0.97	1.00	0.00	0.65
F1-Score	0.97	0.97	0.97	0.79	0.00	0.51

6. CONCLUSION

The growing popularity of internet services has rekindled the public's concern regarding the security of the online infrastructure. As the sophistication of hackers' attempts to breach computer networks and impair the quality of services delivered over them increases, it becomes harder to discern malicious packets from genuine network data. Multiple machine learning (ML) attempts to identify such traffic have been impeded by the necessity for human training of the ML approach whenever a new type of assault is launched. In more recent experiments, unsupervised learning has been utilized to rapidly update IDS by providing examples of new types of assaults. Due to the necessity of clustering several data instances prior to making the appropriate selection, the execution time of such algorithms is increased.

In this study, we offer a unique IDS that automatically adapts its agent to new threats that may disrupt a network's service via reinforcement learning (s). The suggested method examines the agent's efficacy by examining how the packet flow through the network impacts service quality. If the solution determines that a user's actions are degrading service quality, network access for that user will be blocked. The suggested method evaluates the quality of service by considering the ratio of packets permitted access to the network in addition to performance, preventing the agent from blocking all network traffic. As previously observed, the most effective incentive plan was one that included both quality of service and packet-to-rate ratio. In contrast to the 0.51 total F1-Score attained by a neural network based on classification, the proposed method maintained a score of 0.97 despite the inclusion of new attack types throughout the test. This technology can dramatically improve the performance of computer networks by automatically identifying and restricting access to packets that degrade network performance.

Eventually, research will be done to determine the efficacy of various feature-selection strategies. In the past, feature selection was frequently employed in ML-based IDSs. As the suggested method is capable of self-updating, the input vector's properties may become more or less important based on the patterns observed in attack packets. If the significance of features vary from attack to attack, the proposed approach is ineffective, even if reducing the needed model will improve the IDS's performance.

REFERENCES

- [1] G. Kutsuri, S. Kamberdieva, V. K. Dedegkaev, I. Sopoeva, and T. Shelkunova, "Impact of digitalization on improvement of economy, IT and Internet of business," in *Journal of Physics: Conference Series*, 2019, p. 033008.
- [2] M. Reinsdorf and P. Schreyer, "Measuring consumer inflation in a digital economy," in *Measuring Economic Growth and Productivity*, ed: Elsevier, 2020, pp. 339-362.
- [3] A. D. Goolsbee and P. J. Klenow, "Internet rising, prices falling: Measuring inflation in a world of e-commerce," in *Aea papers and proceedings*, 2018, pp. 488-92.
- [4] S. Greenstein, "Digital infrastructure," in *Economics of Infrastructure Investment*, ed: University of Chicago Press, 2019.
- [5] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147-167, 2019.
- [6] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1222-1228.
- [7] S. Otoum, B. Kantarci, and H. T. Mouftah, "On the feasibility of deep learning in sensor network intrusion detection," *IEEE Networking Letters*, vol. 1, pp. 68-71, 2019.
- [8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, pp. 1-22, 2019.
- [9] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, et al., "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," 2018.
- [10] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 2671-2701, 2019.

- [11] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys (CSUR)*, vol. 51, pp. 1-36, 2018.
- [12] A. Dey, "Machine learning algorithms: a review," *International Journal of Computer Science and Information Technologies*, vol. 7, pp. 1174-1179, 2016.
- [13] P. Lison, "An introduction to machine learning," *Language Technology Group (LTG)*, vol. 1, 2015.
- [14] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3-24, 2007.
- [15] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21954-21961, 2017.
- [16] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. M. Hossain, S. Ikhlaiq, and S. Hossain, "Cyber Intrusion Detection Using Machine Learning Classification Techniques," in *International Conference on Computing Science, Communication and Security*, 2020, pp. 121-131.
- [17] M. Belouch, S. El Hadaaj, and M. Idhammad, "Performance evaluation of intrusion detection based on machine learning using Apache Spark," *Procedia Computer Science*, vol. 127, pp. 1-6, 2018.
- [18] A. Ahmad and S. S. Khan, "Survey of state-of-the-art mixed data clustering algorithms," *Ieee Access*, vol. 7, pp. 31883-31902, 2019.
- [19] W. Liang, K.-C. Li, J. Long, X. Kui, and A. Y. Zomaya, "An industrial network intrusion detection algorithm based on multifeature data clustering optimization model," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 2063-2071, 2019.
- [20] K. Peng, V. C. Leung, and Q. Huang, "Clustering approach based on mini batch kmeans for intrusion detection system over big data," *IEEE Access*, vol. 6, pp. 11897-11906, 2018.
- [21] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera, and Y. V. Madhav, "A context-aware robust intrusion detection system: a reinforcement learning-based approach," *International Journal of Information Security*, vol. 19, pp. 657-678, 2020.

- [22] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 31-36.
- [23] R. R. dos Santos, E. K. Viegas, A. Santin, and V. V. Cogo, "A Long-Lasting Reinforcement Learning Intrusion Detection Model," in International Conference on Advanced Information Networking and Applications, 2020, pp. 1437-1448.
- [24] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020.
- [25] K. Sethi, R. Kumar, N. Prajapati, and P. Bera, "Deep reinforcement learning based intrusion detection system for cloud infrastructure," in 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), 2020, pp. 1-6.
- [26] P. Y. Glorennec, "Reinforcement learning: An overview," in Proceedings European Symposium on Intelligent Techniques (ESIT-00), Aachen, Germany, 2000, pp. 14-15.
- [27] D. Acemoglu, A. Malekian, and A. Ozdaglar, "Network security and contagion," *Journal of Economic Theory*, vol. 166, pp. 536-585, 2016.
- [28] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of Microservices-enabled fog applications," *Concurrency and Computation: Practice and Experience*, p. e4436.
- [29] S. Henningsen, S. Dietzel, and B. Scheuermann, "Challenges of Misbehavior Detection in Industrial Wireless Networks," in *Ad Hoc Networks*, ed: Springer, 2018, pp. 37-46.
- [30] N. Pandeewari and G. Kumar, "Anomaly detection system in cloud environment using fuzzy clustering based ANN," *Mobile Networks and Applications*, vol. 21, pp. 494-505, 2016.
- [31] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484-497, 2017.
- [32] J. Jabez and B. Muthukumar, "Intrusion detection system (IDS): anomaly detection using outlier detection approach," *Procedia Computer Science*, vol. 48, pp. 338-346, 2015.

- [33] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical machine learning tools and techniques: Morgan Kaufmann, 2016.
- [34] K. Cup, "Dataset," available at the following website <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>, vol. 72, 1999.
- [35] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," Knowledge-based systems, vol. 78, pp. 13-21, 2015.
- [36] S. Jaiswal, K. Saxena, A. Mishra, and S. K. Sahu, "A KNN-ACO approach for intrusion detection using KDDCUP'99 dataset," in Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, 2016, pp. 628-633.
- [37] N. G. Relan and D. R. Patil, "Implementation of network intrusion detection system using variant of decision tree algorithm," in Nascent Technologies in the Engineering Field (ICNTE), 2015 International Conference on, 2015, pp. 1-5.
- [38] V. D. Mane and S. Pawar, "Anomaly based ids using backpropagation neural network," International Journal of Computer Applications, vol. 136, 2016.
- [39] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security (TISSEC), vol. 3, pp. 262-294, 2000.
- [40] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on, 2014, pp. 1-6.
- [41] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, pp. 446-452, 2015.
- [42] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on, 2015, pp. 92-96.

- [43] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in Military Communications and Information Systems Conference (MilCIS), 2015, 2015, pp. 1-6.
- [44] M. Belouch, S. El Hadaj, and M. Idhammad, "A Two-Stage Classifier Approach using RepTree Algorithm for Network Intrusion Detection," INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, vol. 8, pp. 389-394, 2017.
- [45] H. Gharaee and H. Hosseinvand, "A new feature selection IDS based on genetic algorithm and SVM," in Telecommunications (IST), 2016 8th International Symposium on, 2016, pp. 139-144.
- [46] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, "Intrusion detection system for NSL-KDD data set using vectorised fitness function in genetic algorithm," Adv. Comput. Sci. Technol., vol. 10, pp. 235-246, 2017.
- [47] N. Moustafa and J. Slay, "A hybrid feature selection for network intrusion detection systems: Central points," arXiv preprint arXiv:1707.05505, 2017.
- [48] R. Primartha and B. A. Tama, "Anomaly detection using random forest: A performance revisited," in Data and Software Engineering (ICoDSE), 2017 International Conference on, 2017, pp. 1-6.
- [49] M. Al-Zewairi, S. Almajali, and A. Awajan, "Experimental Evaluation of a Multi-layer Feed-Forward Artificial Neural Network Classifier for Network Intrusion Detection System," in 2017 International Conference on New Trends in Computing Sciences (ICTCS), 2017, pp. 167-172.
- [50] I. U. Din, S. Mahooz, and M. Adnan, "Performance evaluation of different Ethernet LANs connected by Switches and Hubs," European Journal of Scientific Research, vol. 37, pp. 461-470, 2009.
- [51] A. E. Retana, A. C. Lindem III, and R. White, "Dynamically configuring and verifying routing information of broadcast networks using link state protocols in a computer network," ed: Google Patents, 2015.
- [52] Y. Song, S. Gao, A. Hu, and B. Xiao, "Novel attacks in OSPF networks to poison routing table," in 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1-6.
- [53] K. Gould and A. Danforth, "Method to block unauthorized network traffic in a cable data network," ed: Google Patents, 2016.

- [54] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, p. 200892, 2020.
- [55] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics: Review, taxonomy, and open challenges," *Journal of Network and Computer Applications*, vol. 66, pp. 214-235, 2016.
- [56] J. M. Kizza, *Guide to computer network security*: Springer, 2017.
- [57] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Computing Surveys (CSUR)*, vol. 51, pp. 1-36, 2018.
- [58] R. Dabestani, A. Shahin, and M. Saljoughian, "Evaluation and prioritization of service quality dimensions based on gap analysis with analytic network process," *International journal of quality & reliability management*, vol. 34, pp. 530-548, 2017.
- [59] T. P. Fries, "Classification of Network Traffic Using Fuzzy Clustering for Network Security," in *Industrial Conference on Data Mining*, 2017, pp. 278-285.
- [60] A. Sahasrabuddhe, S. Naikade, A. Ramaswamy, B. Sadliwala, and P. Futane, "Survey on Intrusion Detection System using Data Mining Techniques," 2017.
- [61] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Information Networking (ICOIN)*, 2016 International Conference on, 2016, pp. 63-68.
- [62] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report*, James P. Anderson Company, 1980.
- [63] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, pp. 222-232, 1987.
- [64] H. Orman, "The Morris worm: A fifteen-year perspective," *IEEE Security & Privacy*, vol. 99, pp. 35-43, 2003.
- [65] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Research in Security and Privacy*, 1990. *Proceedings.*, 1990 IEEE Computer Society Symposium on, 1990, pp. 296-304.
- [66] D. Dasgupta, "Immunity-based intrusion detection system: a general framework," in *Proc. of the 22nd NISSC*, 1999, pp. 147-160.

- [67] M. Hayashi, M. Owari, G. Kato, and N. Cai, "Secrecy and robustness for active attack in secure network coding," in Information Theory (ISIT), 2017 IEEE International Symposium on, 2017, pp. 1172-1176.
- [68] S.-T. Park, G. Li, and J.-C. Hong, "A study on smart factory-based ambient intelligence context-aware intrusion detection system using machine learning," Journal of Ambient Intelligence and Humanized Computing, vol. 11, pp. 1405-1412, 2020.
- [69] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, "Intrudtree: a machine learning based cyber security intrusion detection model," Symmetry, vol. 12, p. 754, 2020.
- [70] M. M. Hassan, A. Gumaiei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," Information Sciences, vol. 513, pp. 386-396, 2020.
- [71] P. Sengar and N. Bhardwaj, "A Survey on Security and Various Attacks in Wireless Sensor Network," International Journal of Computer Sciences and Engineering, vol. 5, pp. 78-84, 2017.
- [72] D. Leon, F. Mayorga, J. Vargas, R. Toasa, and D. Guevara, "Using of an anonymous communication in e-government services: in the prevention of passive attacks on a network," in 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018, pp. 1-4.
- [73] S. Pareek, A. Gautam, and R. Dey, "Different Type Network Security Threats and Solutions, A Review," International Journal of Computer Science, vol. 5, 2017.
- [74] S. Latha and S. J. Prakash, "A survey on network attacks and Intrusion detection systems," in Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on, 2017, pp. 1-7.
- [75] M. E. Aminanto and K. Kim, "Detecting active attacks in Wi-Fi network by semi-supervised deep learning," in Conference on Information Security and Cryptography, 2017.
- [76] G. Nguyen, S. Dlugolinsky, V. Tran, and Á. L. García, "Deep learning for proactive network monitoring and security protection," IEEE Access, vol. 8, pp. 19696-19716, 2020.
- [77] E. M. Tzanakou, Supervised and unsupervised pattern recognition: feature extraction and computational intelligence: CRC Press, 2017.

- [78] S. Liu and M. d'Aquin, "Unsupervised learning for understanding student achievement in a distance learning setting," in Global Engineering Education Conference (EDUCON), 2017 IEEE, 2017, pp. 1373-1377.
- [79] E. Hancer and D. Karaboga, "A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number," *Swarm and Evolutionary Computation*, vol. 32, pp. 49-67, 2017.
- [80] S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, pp. 70-73, 2014.
- [81] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 59-68.
- [82] S. Suthaharan, "Support vector machine," in *Machine learning models and algorithms for big data classification*, ed: Springer, 2016, pp. 207-235.
- [83] R. Lior, *Data mining with decision trees: theory and applications* vol. 81: World scientific, 2014.
- [84] H.-R. Zhang and F. Min, "Three-way recommender systems based on random forests," *Knowledge-Based Systems*, vol. 91, pp. 275-286, 2016.
- [85] R. J. Schalkoff, *Artificial neural networks* vol. 1: McGraw-Hill New York, 1997.
- [86] M. Valipour, "Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms," *Meteorological Applications*, vol. 23, pp. 91-100, 2016.
- [87] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, pp. 436-444, 2015.
- [88] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514-525, 2014.
- [89] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, ed: Elsevier, 1994, pp. 157-163.
- [90] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330-337.

- [91] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279-292, 1992.
- [92] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
- [93] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [94] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, et al., "Deep q-learning from demonstrations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [95] K. Gurney, *An introduction to neural networks*: CRC press, 2014.
- [96] O. Niculaescu, "Neural networks and how machines learn meaning," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 25, pp. 64-66, 2019.
- [97] O. Eluyode and D. T. Akomolafe, "Comparative study of biological and artificial neural networks," *European Journal of Applied Engineering and Scientific Research*, vol. 2, pp. 36-46, 2013.
- [98] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117-1125.
- [99] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013, pp. 1058-1066.
- [100] S. Sharma, "Activation functions in neural networks," *towards data science*, vol. 6, 2017.
- [101] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
- [102] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, pp. 1137-1149, 2017.
- [103] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, pp. 111-122, 2011.

- [104] M. Sheinfeld, S. Levinson, and I. Orion, "Highly accurate prediction of specific activity using deep learning," *Applied Radiation and Isotopes*, vol. 130, pp. 115-120, 2017.
- [105] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*, 2019, pp. 1675-1685.
- [106] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, et al., "Learning to learn by gradient descent by gradient descent," *arXiv preprint arXiv:1606.04474*, 2016.
- [107] J. He, J. Rexford, and M. Chiang, "Design for optimizability: Traffic management of a future internet," in *Algorithms for Next Generation Networks*, ed: Springer, 2010, pp. 3-18.
- [108] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [109] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249-256.
- [110] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, ed: Elsevier, 1992, pp. 65-93.
- [111] I. K. Sethi and A. K. Jain, *Artificial neural networks and statistical pattern recognition: old and new connections* vol. 11: Elsevier, 2014.
- [112] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85-117, 2015.
- [113] Z. Zainuddin and O. Pauline, "Function approximation using artificial neural networks," *WSEAS Transactions on Mathematics*, vol. 7, pp. 333-338, 2008.
- [114] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Eötvös Loránd University, Hungary*, vol. 24, p. 7, 2001.
- [115] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137-2145.
- [116] Y. Chen and E. Kula, "A Deep Q-Network with Experience Optimization (DQN-EO) for Atari's Space Invaders," in *Workshops of the International Conference on Advanced Information Networking and Applications*, 2019, pp. 351-361.

- [117] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, 2015, pp. 2863-2871.
- [118] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529, 2015.
- [119] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.
- [120] R. Sahani, C. Rout, J. C. Badajena, A. K. Jena, and H. Das, "Classification of intrusion detection using data mining techniques," in *Progress in computing, analytics and networking*, ed: Springer, 2018, pp. 753-764.
- [121] Z. Shen, Y. Zhang, and W. Chen, "A Bayesian Classification Intrusion Detection Method Based on the Fusion of PCA and LDA," *Security and Communication Networks*, vol. 2019, 2019.
- [122] S. Bhattacharya, R. Kaluri, S. Singh, M. Alazab, and U. Tariq, "A Novel PCA-Firefly based XGBoost classification model for Intrusion Detection in Networks using GPU," *Electronics*, vol. 9, p. 219, 2020.
- [123] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525-41550, 2019.
- [124] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & security*, vol. 21, pp. 439-448, 2002.
- [125] B. B. Rao and K. Swathi, "Fast kNN classifiers for network intrusion detection system," *Indian Journal of Science and Technology*, vol. 10, pp. 1-10, 2017.
- [126] A.-U.-H. Qureshi, H. Larijani, N. Mtetwa, A. Javed, and J. Ahmad, "RNN-ABC: A new swarm optimization based technique for anomaly detection," *Computers*, vol. 8, p. 59, 2019.
- [127] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210-42219, 2019.
- [128] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850-50859, 2018.

- [129] H. Liu, B. Lang, M. Liu, and H. Yan, "CNN and RNN based payload classification methods for attack detection," *Knowledge-Based Systems*, vol. 163, pp. 332-341, 2019.
- [130] J. Cui, J. Long, E. Min, Q. Liu, and Q. Li, "Comparative study of CNN and RNN for deep learning based intrusion detection system," in *International Conference on Cloud Computing and Security*, 2018, pp. 159-170.
- [131] N. Neha, M. G. Raman, N. Somu, R. Senthilnathan, and V. S. Sriram, "An Improved Feedforward Neural Network Using Salp Swarm Optimization Technique for the Design of Intrusion Detection System for Computer Network," in *Computational Intelligence in Pattern Recognition*, ed: Springer, 2020, pp. 867-875.
- [132] R. Shalini and A. Kumaravel, "Deep Reinforcement Learning for Joint Object Search," *International Journal of Pure and Applied Mathematics*, vol. 119, pp. 1747-1751, 2018.
- [133] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *Journal of the Franklin Institute*, vol. 355, pp. 1752-1779, 2018.
- [134] A. Mehmood, A. Khanan, M. M. Umar, S. Abdullah, K. A. Z. Ariffin, and H. Song, "Secure knowledge and cluster-based intrusion detection mechanism for smart wireless sensor networks," *IEEE Access*, vol. 6, pp. 5688-5694, 2017.
- [135] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Applied Sciences*, vol. 9, p. 238, 2019.
- [136] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96-109, 2019.
- [137] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera, and Y. V. Madhav, "A context-aware robust intrusion detection system: a reinforcement learning-based approach," *International Journal of Information Security*, pp. 1-22, 2019.
- [138] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology*, vol. 7, pp. 479-482, 2018.

- [139] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, et al., "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," arXiv preprint arXiv:1803.11347, 2018.
- [140] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," arXiv preprint arXiv:1704.03952, 2017.
- [141] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, et al., "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," IEEE Transactions on Vehicular Technology, vol. 68, pp. 4192-4203, 2019.
- [142] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," Autonomous Agents and Multi-Agent Systems, vol. 33, pp. 750-797, 2019.