A. SAAD

REINFORCEMENT LEARNING FOR INTRUSION DETECTION

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

AHMED MOHAMED SAAD EMAM SAAD

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

APRIL 2021

ATILIM UNIVERSITY 2021

REINFORCEMENT LEARNING FOR INTRUSION DETECTION

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

BY

AHMED MOHAMED SAAD EMAM SAAD

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

APRIL 2021

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

Prof. Dr. Ender KESKİNKILIÇ
Director

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

Assoc. Prof. Dr. Gökhan ŞENGÜL
Head of Department

This is to certify that we have read the thesis **"Reinforcement Learning for Intrusion Detection"** submitted by **Ahmed Mohamed Saad Emam Saad** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Beytullah YILDIZ
Supervisor

Examining Committee Members:

Prof. Dr. Ali YAZICI
Software Engineering, Atilim University

Asst. Prof. Dr. Beytullah YILDIZ
Computer Engineering, Hacettepe University

Asst. Prof. Dr. Adnan ÖZSOY
Software Engineering, Atilim University

Date: April 30, 2021

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ahmed Mohamed Saad Emam Saad

ABSTRACT

Reinforcement Learning for Intrusion Detection

Saad, Ahmed Mohamed Saad Emam

M.S., Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Beytullah YILDIZ

April 2021, 60 pages

Network-based technologies such as cloud computing, web services, and Internet of Things systems are becoming widely used due to their flexibility and preeminence. On the other hand, the exponential proliferation of network-based technologies exacerbated network security concerns. Intrusion takes an important share in the security concerns surrounding network-based technologies. Developing a robust intrusion detection system is crucial to solve the intrusion problem and ensure the secure delivery of network-based technologies and services. In this thesis, a novel approach was proposed using deep reinforcement learning to detect intrusions to make network applications more secure, reliable, and efficient. As for the reinforcement learning approach, Deep Q-Learning is used alongside a custom-built Gym environment that mimics network attacks and guides the learning process. A supervised deep learning solution using a Long-Short Term Memory architecture is implemented to serve as a baseline. The NSL-KDD dataset is used to create the reinforcement learning environment and to train and evaluate the baseline model. The performance results of the proposed reinforcement learning approach show great superiority over the baseline model and the other relevant solutions from the literature.

Keywords: Reinforcement learning, Network Security, Intrusion Detection System,

Long Short-Term Memory, Deep Q-learning

ÖZ

Saldırı Tespiti için Takviyeli Öğrenme

Saad, Ahmed Mohamed Saad Emam

Yüksek Lisans, Bilgisayar Mühendisliği

Tez Yöneticisi : Dr. Öğr. Üyesi Beytullah Yıldız

Nisan 2021, 60 sayfa

Bulut bilişim, web servisleri ve Nesnelerin İnterneti sistemleri gibi ağ tabanlı teknolojiler, esneklikleri ve üstünlükleri nedeniyle yaygın olarak kullanılmaktadır. Öte yandan, ağ tabanlı teknolojilerin katlanarak büyümesi, ağ güvenliği sorunlarının büyüklüğünü artırmaktadır. İzinsiz giriş, ağ tabanlı teknolojilerin güvenliğinin önemli bir parçasıdır. Sağlam bir saldırı tespit sistemi uygulamak, izinsiz giriş sorununu çözmek ve ağ tabanlı teknolojilerin ve hizmetlerin güvenli bir şekilde sunulmasını sağlamak için çok önemlidir. Bu tezde, izinsiz girişleri tespit etmek ve ağ uygulamalarını daha güvenli, güvenilir ve verimli hale getirmek için pekiştirmeli öğrenmeyi kullanan yeni bir yaklaşım öneriyoruz. Takviye öğrenme yaklaşımı olarak, ağ trafiği saldırılarını taklit eden ve öğrenme sürecine rehberlik eden, özel olarak uyarlanmış bir Gym ortamının yanında kullanılan derin Q-öğrenme kullanılmaktadır. Uzun-Kısa Süreli Bellek kullanan denetimli bir derin öğrenme çözümü, karşılaştırma için temel yaklaşım alarak uygulanmıştır. NSL-KDD veri kümesi, takviye öğrenme ortamını oluşturmak için kullanılmakta olup temel modeli eğitmek ve değerlendirmek için de kullanılır. Önerilen pekiştirmeli öğrenme yaklaşımının performans sonuçları, temel modele ve literatürdeki diğer çözümlere göre büyük bir üstünlük göstermektedir.

Anahtar Kelimeler: Takviyeli Öğrenme, Ağ Güvenliği, Saldırı Tespit Sistemi, Uzun Kısa Süreli Bellek Sinir Ağı, Derin Q-öğrenme

To my dear mother

ACKNOWLEDGMENTS

I am grateful to my supervisor, mentor, and friend, Asst. Prof. Dr. Beytullah YILDIZ without whom this thesis and getting this degree will not be possible. His knowledge, ideas, motivation, help, and patience kept me going even in my most desperate times. It was a pleasure working with him and being one of his students.

I would like to show my gratitude and appreciation to my mother and father for the unconditional support throughout my educational journey and my life, without whom getting this degree will not be possible.

I want to show my appreciation to my dear friend Hisham for helping in proofreading this thesis.

Finally, I would like to thank everyone from the academic and administrative staff at Atilim University for the work they do for international students to make the educational process as convenient as possible.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

RL	:	Reinforcement Learning
IDS	:	Intrusion Detection System
DoS	:	Denial of Service
U2R	:	User to Root
R2L	:	Remote to Local
TP	:	True Positive
TN	:	True Negative
FP	:	False Positive
FN	:	False Negative
TPR	:	True Positive Rate
AWS	:	Amazon Web Service
LSTM	:	Long Short-Term Memory
QL	:	Q-Learning
MDP	:	Markov Decision Process
DQN	:	Deep Q-Network
DDQN	:	Double Deep Q-Network
PG	:	Policy Gradient
AE-DQN	:	Adversarial/Multi Agent Reinforcement Deep Q-Learning
AE-RL	:	Adversarial Environment using Reinforcement Learning
A3C	:	Asynchronous Advantage Actor Critic
Dueling DDQN	:	Dueling Double Deep Q-Network

CHAPTER 1

INTRODUCTION

Network-based computer systems and technologies like web services, cloud computing, and Internet of Things (IoT) systems are becoming more popular. The most famous network-based technology is cloud services. This popularity is due to the growing number of cloud service providers and the astonishing technological achievement in these services over the past few years. These technological achievements opened the doors to alternative possibilities and many changes in the way companies and different organizations plan to build their Information Technology (IT) infrastructure. The flexibility and the state-of-the-art services that cloud service providers offer make it easy for the different organizations to get the services needed without worrying about the hardware side of the infrastructure. For example, the hardware infrastructure requires utility bills to be paid and a suitable environment for maintaining its peak performance. Another reason for its popularity is the easy-to-use models that technology giants like Microsoft, Google, Amazon, IBM, and Alibaba provide to their customers. Services like Amazon Web Service (AWS), IBM Cloud, Microsoft Azure, Alibaba Cloud, and Google Cloud provide are becoming the ultimate replacement and solution for handling and providing the infrastructure of computer systems, storage space, and computational power that is needed by agencies, companies, organizations, institutions, and governments. Those services can be acquired at a remarkably affordable cost, and that is the primary reason for their fame among contemporary era organizations and government agencies.

Network-based technologies like cloud computing services are prone to intrusion, and the growing popularity of network-based systems made the intrusion issue worse. We can get an estimate on the expansion magnitude of network-based technologies and

in return the intrusion problem by examining the market of specific network-based technologies such as cloud computing services. It is evident from the massive increase in the market and the revenue of cloud services from the year 2017 in which the revenue estimate was around 54.9 billion US dollars [1] to the year 2020 in which the revenue estimate was around 129 billion US dollars [2] that network-based technologies are getting a lot of attraction, which increases the scale of the intrusion issue. This increasing scale comes with significant economic costs, which has been confirmed by two studies carried out by McAfee cyber-security firm. The two studies were conducted over a 6-year period and show the alarming increase in the economic cost of cyber-crimes. The first study shows that in 2014 the cost of cyber-attacks was around 475 billion US dollars [3], and the second study shows that in 2020 the cost of cyber-attacks was almost 1 trillion US dollars [4].

Since most of the network-based technologies and resources are obtained from a remote service provider that is not locally present through a medium, this raises the question of how to secure the medium used to obtain these services from intrusion, which in this case is the network system. The answer will significantly reduce the economical cost caused by intrusions and cyber-crimes. Securing that medium requires an absolute necessity of a modern solution to detect intrusions using a particular intrusion detection system (IDS). Traditional rule-based intrusion detection systems like snort intrusion detection system [5] were used to tackle and solve the intrusion problem. However, with the increase in novel attacks and the continuous change in the attack types and styles, rule-based intrusion detection systems are vulnerable. Even with rapid updates to their rules, they can not keep up with the continuous change in malicious attacks. Therefore, creating a novel, scalable, and adaptive approach for detecting intrusions in network systems that copes with the new malicious attacks is a necessity. This is where machine learning comes into place with its adaptability and flexibility. It can provide a solution for the intrusion issue addressed before and solve the limitation of rule-based intrusion detection systems. Although machine learning offers a solution, not all machine learning approaches are created equal. Machine learning can be categorized into three major topics: the first is reinforcement learning, the second is unsupervised learning and the third is supervised learning. Many solutions were developed for intrusion detection using the three different machine

learning approaches.

The aim of this thesis is to provide an answer to the mentioned problem by introducing and applying a novel reinforcement learning approach and solution for intrusion detection in network systems. Another objective is to gain an insight into the performance of the novel approach compared to a traditional approach and other relevant work and to achieve better results.

In this thesis, we propose a novel approach using reinforcement learning because, in theory, it is superior to other machine learning approaches in intrusion detection for the following reasons. First, it can go beyond the dataset by solving the labeling issue. Second, it can generalize and approximate when dealing with large observation space or features. Third, it can scale and adapt to numerous attack patterns, and it is not volatile to changes. The novel machine learning approach introduced utilizes a reinforcement learning-based algorithm called Deep Q-Network. Reinforcement learning utilizes an environment to train an agent. We use OpenAI Gym library to build the environment. The Gym environment guides the learning process through positive and negative rewards and makes use of the NSL-KDD intrusion detection dataset as a source of network traffic. In other words, the reinforcement learning agent learns from its previous actions by observing the states and rewards from the environment, so it can perform better actions in the future by maximizing the reward it gets from the environment. The reinforcement learning agent uses deep neural networks as a function approximator for Q-values associated with decision (action) making. For the NSL-KDD dataset, it is the largest and the most diverse in terms of attack types, and it fits the eleven criteria for an appropriate IDS dataset [6]. It is well-suited and serves the goal of this research. The dataset goes through preprocessing stage before being used as the source of encoded network traffic.

The proposed approach will be compared with a supervised learning approach that uses Long-Short Term Memory (LSTM) neural network which will serve as a baseline. This approach is based on the idea of recognizing attacks from old network traffic attack data upon which it can detect relatively similar attacks. The same dataset used for reinforcement learning is used for the baseline supervised learning approach after some preprocessing and encoding to get the most relevant features from the dataset

and to improve the overall performance of the approach.

The main contribution of this study is first our novel reinforcement learning approach that offers a significant improvement in terms of metrics, such as accuracy, recall, and precision compared to the baseline solution and other relevant works in the literature. Second, we create a customized environment that is more expressive about intrusions and has more resolution. This is one of the key elements for the success of the reinforcement learning model. Finally, we perform several hyper-parameter tunings to improve intrusion detection.

This thesis is organized as follows. The second chapter provides a brief background and an overview of the related work that uses machine learning techniques to detect intrusions and some other work of different applications of reinforcement learning applied in different areas. The third chapter explains the methodology, including the proposed model, the dataset used, and the baseline model. The fourth chapter shows the experimental results and the overall evaluation. The fifth and final chapter provides conclusions and suggested future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter presents a background and literature review to provide the important concepts as well as previous related studies in the area covered by this thesis. The chapter also reviews some existing solutions for intrusion detection and investigates their performance and the various used techniques. Moreover, this chapter concisely introduces an overview of intrusion, different types of attacks, and detection systems. Finally, an overview of machine learning main aspects is also presented.

2.1 Background

The increasing adoption of network-based technologies leads to the increase in networks as part of every modern technology. This gave network systems a vital role in every computer system application. The increase in network-based applications can be seen from the change in the market share of network-based technologies like cloud services from 2017 as shown in Figure 2.1 [1] to the year 2020 as shown in Figure 2.2 [2]. This implies that network-based technologies are being adopted more frequently over time. Security can be defined as the protection of information in an automated way to ensure its confidentiality, availability, and integrity. Networks are known to be targeted by various intrusion attacks, which threaten the network and the security of its applications. The trend of network and cyber threats keeps growing each year as shown by a report done by McAfee cyber-security firm [7]. In the next sections, intrusion detection systems, intrusion types, and solutions will be explained.

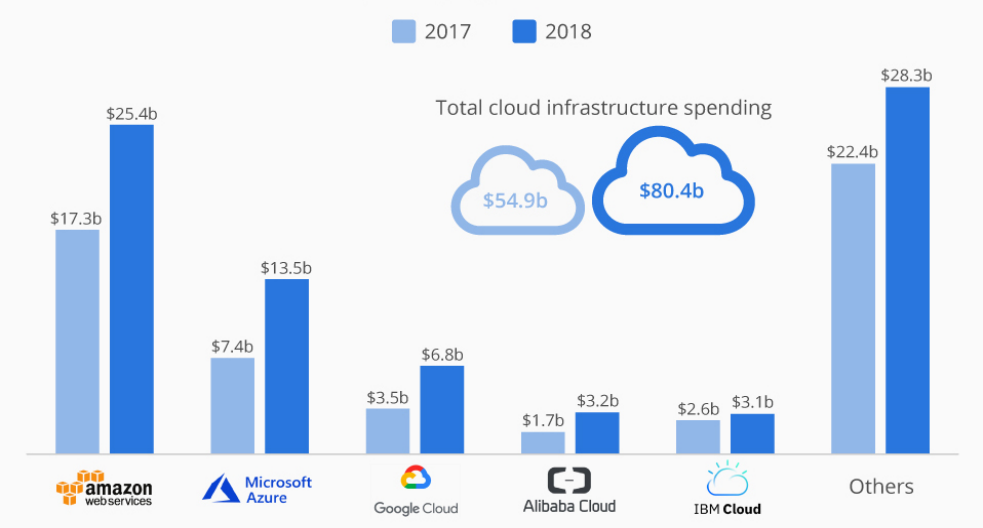

Figure 2.1 Cloud Market Share 2017 and 2018 [1]

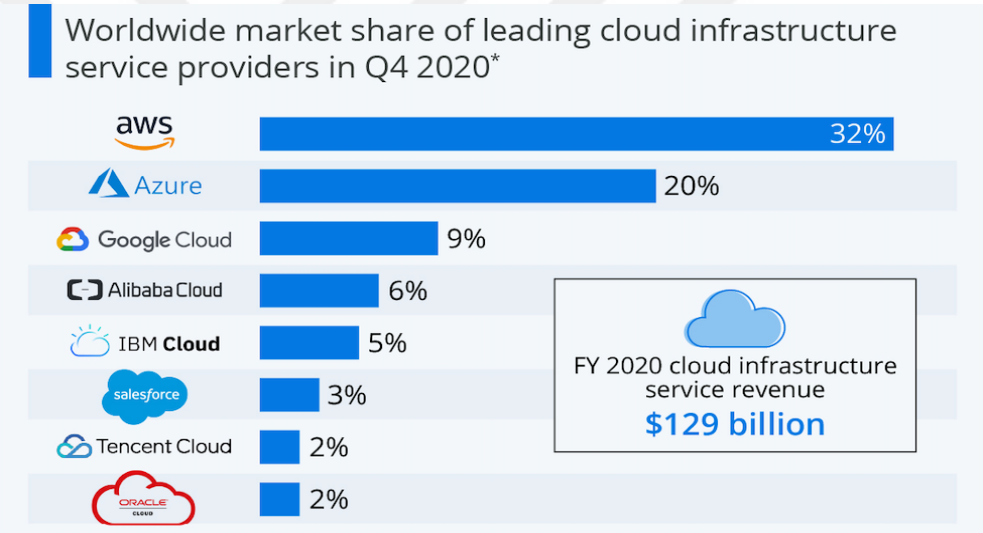

Figure 2.2 Cloud Market Share 2020 [2]

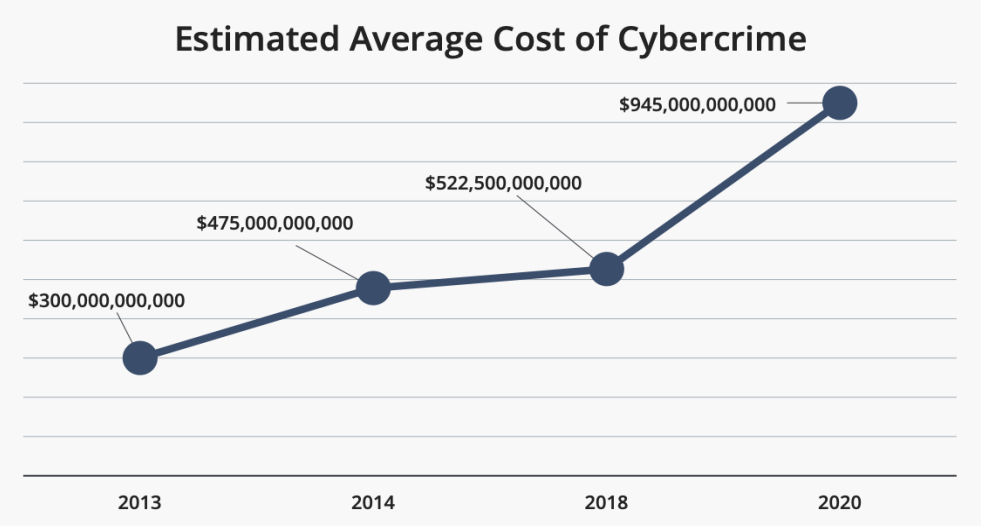

Figure 2.3 Economic Cost of Cybercrimes 2013 to 2020 [7]

2.1.1 Intrusions

Network intrusion refers to the abnormal or suspicious activity on a network system. Intrusions can vary in a long range but most of them fall under four main categories. These categories are represented in the NSL-KDD dataset used in this thesis.

- Intrusion Main Categories
  - Probe
  - Denial of Service (DoS)
  - User to Root (U2R)
  - Remote to Local (R2L)

2.1.1.1 Probe

It is a type of intrusion attack with the goal of stealing relevant information that will help in a future attack. Most attacks start with Probe attack.

2.1.1.2 DoS

DoS attack aims to overflow the network system with unnecessary traffic that it can not handle in order to stop actual users from reaching the target network.

2.1.1.3 U2R

U2R attack is the attempt of a normal user to acquire admin privileges over a network system.

2.1.1.4 R2L

In R2L attack, the intruder tries to gain normal user privileges to a remote network that he/she does not have permission to access. It is considered the attack preceding U2R.

2.1.2 Network Security Systems

Network security systems that deal with intrusions can be categorized into three main defense mechanisms: intrusion prevention, intrusion response, and intrusion detection.

2.1.2.1 Intrusion Prevention

Intrusion prevention system mechanism works to stop the attack from happening or make the targeted system cope with the attack without affecting the system users.

2.1.2.2 Intrusion Response

Intrusion response system attempts to mitigate the damage caused by the intruders to the network system.

2.1.2.3 Intrusion Detection

An intrusion detection system works to monitor network traffic for suspicious or abnormal activities and sends an alarm if suspicious activity is detected.

2.1.3 Intrusion Detection Systems

Intrusion detection systems work in different ways and have many architectures. The most common two types of network intrusion detection systems are signature (misuse) based intrusion detection system and anomaly-based intrusion detection system.

2.1.3.1 Signature-Based Intrusion Detection System

Signature-based intrusion or misuse detection system works similarly to a rule-based system. It can recognize attacks' fingerprints that were previously introduced to it. However, it fails when detecting unintroduced or novel attacks.

2.1.3.2 Anomaly-Based Intrusion Detection System

Anomaly-based intrusion detection system attempts to deal with a novel attack by learning the normal pattern of network traffics, and any deviation from that normal pattern is considered as an intrusion. The downside of this system is its high sensitivity that leads to a high false positive rate.

2.2 Machine Learning Approaches

Machine learning is building a model or an algorithm that improves, makes decisions, and automatically learns without being explicitly programmed through experiences and exposure to data. It is categorized into three main topics: supervised learning, unsupervised learning, and reinforcement learning.

2.2.1 Supervised Learning

In supervised learning, the labeled dataset is presented to the algorithm in the training process. After learning from the dataset, the model can make predictions. In the case of an IDS, using a supervised learning approach will work similarly to a signature-based intrusion detection system.

2.2.2 Unsupervised Learning

In unsupervised learning, the algorithm is trained on unlabeled data, and in the training process, it tries to detect patterns from the data. The disadvantage of using unsupervised learning in an IDS is the high rate of false positives and negatives.

2.2.3 Reinforcement Learning

Reinforcement learning is a type of machine learning that has the ability to learn through trial-and-error interaction with a dynamic environment. Reinforcement learning consists of two main components: the agent and the environment. The learning starts when a state is observed by the agent, and an action is taken, which is then rewarded. This learning process occurs recursively by focusing on maximizing the reward until learning is terminated.

2.2.3.1 Model-Free vs. Model-Based

Reinforcement learning requires an environment for agents to interact with. Depending on the environment, different methods can be chosen. Model-based is a method used when predictions can be made based on expected values of the environment. On the other hand, Model-free is a method used when the system is constantly storing past experiences.

2.2.3.2 Monte Carlo Method

Monte Carlo Method is a model-free method. This is an experience-based method in which we just focus on the value functions directly from the interactions with the environment. It depends on calculating the value function of a certain policy directly from the episodes (group of states) of experience.

2.2.3.3 Temporal Difference Learning

Temporal-Difference learning is a model-free method. The difference between the Monte Carlo method and the Temporal-Difference learning is the frequency of values update. In Temporal-Difference, the value updates at each state, unlike the Monte Carlo method which updates the value at the end of an episode.

2.2.3.4 Q-Learning

Q-learning is a model-free reinforcement learning and Temporal-Difference learning method. It introduces a new function which is called Q-function. The value function exploits each state quality and in the Q-function assigns a value to every action taken by the agent at all states.

2.2.3.5 Deep Q-Learning

The mentioned Q-learning is a powerful algorithm, but it can not estimate values of unknown states. Hence, some Q-values can not be calculated. This is where another algorithm called Deep Q-Network (DQN) comes into place. It deals with this problem of lack of generality. A neural network is implemented, where the states are the input and the estimates of Q-values for each action are the output. Deep Q-Learning algorithm is used in this thesis and will be explained in detail in the next chapter.

2.3 Related Work

Bouzida and Cuppens [8] compared two supervised learning approaches. The first approach was decision tree and the second one was neural network. Both of the approaches were tested on the KDD 99 dataset and over real-time network traffic. It was concluded that both neural networks and decision tree performed well in generalization while decision tree performed better in detecting novel attacks.

An approach that merges an unsupervised learning algorithm (K-means) for clustering alongside a supervised learning one (support vector machine or SVM) for classification was introduced by Aljamal et al. [9]. The K-means algorithm clustered the data from network traffic into 64 clusters, and then the SVM algorithm used to categorize these clusters into two categories: normal and abnormal. The evaluation was done using the benchmark UNSW-NB15 dataset. The overall performance was low when compared with other researches, and the false positive rate was very high.

A deep learning approach was proposed by Parampottupadam and Moldovann [10]. The approach is making use of two deep learning models: binomial classification and multinomial model. The binominal method is used to identify intrusion, while the multinominal model is used to categorize the attack according to its type. The models were built using H2O and Deeplearning4j libraries, and they used only the NSL-KDD dataset to evaluate the proposed approach.

An intrusion detection approach for IoT based on an artificial neural network was proposed by Hodo et al. [11]. This research was based on gathering and analyzing data from a real normal IoT network and then training the artificial neural network using the internet packet traces. If the incoming network traffic was unrecognizable by the neural network, it sends a warning to the responsible personnel. This research was only used to detect DDoS and DoS attacks.

Bostani and Sheikhan [12] suggested a multicomponent intrusion detection system with a centralized module that uses an unsupervised machine learning approach (Optimum Path Forest) based on a MapReduce approach. This intrusion detection system was designed to detect only two types of attacks, which are selective-forwarding and sinkhole attacks. Moreover, it can expand to detect wormhole attacks. The overall

performance was 76.19% true positive rate.

A proposal included combining feature selection alongside a supervised learning classifier (SVM) was introduced by Pervez and Farid [13]. They evaluated their model using only the NSL-KDD dataset. They only used one algorithm and one dataset to test their approach.

A two-tier intrusion detection system for differentiating between normal and abnormal traffic patterns using a supervised machine learning approach (KNN) as a classifier combined with reinforcement learning that was used to decrease the false-positive rate was introduced by Divyatmika and Sreekesh [14]. Additionally, Multilayer Perceptron (MLP) algorithm was used for misuse detection, and an unsupervised learning approach was used to build a database which can be utilized to detect new attacks.

Mukkamala et al. [15] compared the use of a neural network and supervised learning technique namely SVM to build an intrusion detection solution. First, they intended to build a classifier using patterns or features from user behavior in order to identify anomalies. They used the KDD dataset as a benchmark, and they achieved high accuracy on the test data of nearly 99%. The time taken for the support vector machine to train was shorter than that of the neural network, but both approaches were not tested further on data other than the test data.

Sinclair et al. [16] suggested an attempt to automate the process of anomaly and intrusion detection. They used a genetic algorithm and decision tree to generate rules to be used later as a network traffic classifier which detects anomalies. This approach, however, was never truly tested on a real dataset.

Lee and Stolfo [17] attempted to identify patterns and features that describe normal user behavior. They then created a classifier to identify anomalies through a detection agent that uses data mining techniques such as association rules algorithm and frequent rule algorithm to recognize anomalies. As in [16], this approach was also never tested on real data.

Sommer and Paxson [18] stated some guidelines for the future of using ML techniques to detect intrusions under four main principles. These principles are understanding the threat model, keeping the scope narrow, reducing the costs, and evaluating the model.

They also stated what researchers should do in order to contribute to this ongoing research area.

An arbitrary model consisting of two parts: an unsupervised clustering algorithm and a traditional anomaly detection solution was proposed by Zanero and Savaresi [19]. The former attempted to reduce traffic payload to a traceable size, and the latter's efficiency depended on the data available. Their model was tested on data generated by a MIT laboratory. They compared the performance of various candidate algorithms (S.O.M., principal direction, and K-means) to validate their proposed model. However, the second part on the model was tested on preliminary results from previous studies, and the full architecture was not present for a full test.

Chen et al. [20] compared the performance of two machine learning techniques for intrusion detection: the first one used a neural network, and the second one used SVM. Both approaches were tested on 1998 DARPA BSM audit data. The SVM solution's performance was superior to the ANN one, but the two models were not deployed on a real-time system for evaluation.

A supervised learning technique to detect intrusions based on Bayes' theorem, which deals with intrusions as probabilities was proposed by Scott [21]. The model was not tested on a dataset, and it is not fully automated and functional as it requires human interaction to help the system identify intrusions.

Li and Guo [22] implemented a supervised machine learning approach (KNN) using fewer data and features for training than other approaches, achieving higher accuracy and lower FP rate. The researchers used the KDD Cup 1999 dataset for proving the effectiveness of their proposal. They tested different K values, and each K value achieved nearly the same accuracy of more than 99% on the test dataset. However, it was not evaluated based on data other than the test dataset.

Tsai et al. [23] provided a survey of the researchers that implemented machine learning techniques for intrusion detection from the year 2000 to the year 2007. The survey reviewed 55 research papers based on the proposed model under three clusters (hybrid, single, and ensemble classifiers).

A deep reinforcement learning-based model to detect intrusions in the cloud environ-

ment with a low FP rate was suggested by Sethi et al. [24]. The researchers used the UNSW-NB 15 dataset as a benchmark and the Chi-square algorithm for feature selection. The proposed model consisted of three parts: administrator network, host network, and agent network. The researchers also combined their model with some other classifiers, achieving an accuracy of 83% and a false positive rate of 2.6%. The model, on the other hand, was not deployed or evaluated on a real cloud environment.

Liang et al. [25] suggested a hybrid approach of a multi-agent reinforcement learning model consisting of three parts: data management, analysis and response modules, and data collection. The analysis modules are based on deep learning to detect anomalies from the transport layer in the network. The dataset used for evaluation was the NSL-KDD dataset. The anomaly detection accuracy of 98% was achieved in an IoT environment. The ability for the proposed model to classify different types of attacks was accurate by 97%. Nevertheless, the model was only tested in an IoT environment where the types of attacks are very limited.

A distributed deep learning model to detect attack patterns after training on a dataset to differentiate the attacks from the normal traffic was proposed by Diro and Chilamkurti [26]. The NSL-KDD dataset was used after being encoded for training and evaluation. They also compared the performance of centralized and distributed models with some other shallow learning algorithms. They achieved an accuracy of more than 99%. The paper, however, only introduced the distributed nature of an intrusion detection system, which consists of a master node and cooperative nodes that can propagate updates through the master node. This distributed nature can reduce computation overhead and overfitting, although this approach was not proven effective and was not investigated thoroughly.

Du et al. [27] suggested an anomaly detection approach using Long Short Term Memory. They called it DeepLog, which depends on analyzing the normal logs as a natural language sequence (structured language) so that it can recognize the normal pattern and then which pattern deviated from the normal one. They also demonstrated how it can be updated so it can adapt to novel attacks. They used Blue Gene/L dataset, which is composed of supercomputer logs. One of the downsides of the research is the high computational overhead and the massive hardware it needs to function.

A reinforcement learning-based model which is using an off-policy approach (actor-critic) for intrusion detection was introduced by Sagha et al. [28]. They also used Temporal Difference learning to learn the rules and the appropriate behavior from a control system, and they claim it can adapt to novel attacks.

A reinforcement-based model to defend users against network attacks was proposed by Xia et al. [29]. The proposed model consists of two agents: one that works as a defending agent and the other works as an attack generating agent. The paper's approach was new but it was theoretical and was not tested on any dataset. The functionality of the proposed model and the availability of datasets make the success of this model questionable.

Koduvely [30] proposed making a Gym environment based on the OpenAI Gym environment concept to detect network intrusions using reinforcement learning and policy gradient model, which will be used later for comparison purposes with the approach suggested in this thesis. The proposed approach works by solving the environment, and for evaluation a ROC curve is used. The proposed solution's performance was not evaluated, and the FP and FN rates were unknown. The research also suggests implementing other techniques such as deep neural network and deep and wide neural network, but there was no continuation on this research.

A new feature selection from the NSL-KDD dataset approach called SFSDT, which utilizes two different approaches for detection of intrusions in network traffic: forward selection algorithm and decision tree model was proposed by Le et al. [31]. Then, it feeds those features to three different neural networks that use different architectures (Gated Recurrent Unit, Recurrent Neural Network, and Long Short-Term Memory). The LSTM had the best performance when tested on some types of attacks compared to other models. The authors claim that their approach reduced the time required for the overall computation.

A novel approach using deep reinforcement learning to detect attacks in a network without requiring to solve an environment by directly using batches from two datasets (NSL-KDD an AWID) separately was suggested by Lopez-Martin et al. [32]. This approach proposes a new reward method which is rewarding the model in the training process, whether the detection was correctly performed or not. They used four

different approaches to implement their proposal, which are Policy Gradient, Double Deep Q-Network (DDQN), Actor-Critic, and Deep Q-Network (DQN). They claim that the top performance was achieved by the Double Deep Q-Network approach and that they were able to decrease the overall computational time required compared to traditional machine learning approaches.

Tang et al. [33] suggested a novel intrusion detection model utilizing the LightGBM algorithm as a feature selector, which chooses features based on an assigned score to each feature from the NSL-KDD dataset. It then feeds the selected feature to several encoders, which are autoencoder, variation autoencoder, and denoising autoencoder for the training process and anomaly detection. The research also compared several traditional machine learning approaches (XGBoost, Decision Tree, Random Forest, K-Nearest Neighbor and Gradient Boosting decision tree) to the proposed model. The performance was evaluated using the F1-Score evaluation approach. The performance of the auto encoder was the highest with almost 90% accuracy.

An approach called reinforcement in-game learning was used by Jeerige et al. [34]. To get a deep understanding of reinforcement learning, they re-implemented research papers' approaches and compared the performance of each one. They created an agent which can solve an environment which is the Atari 2600 game (Breakout), using two reinforcement learning algorithms (Actor-Critic and Deep Q-Learning). The actor-critic model performed better than the Deep Q-Learning Model.

Coronato et al. [35] suggested the use of reinforcement learning in medical software systems, which they claim can put the risk of processes ran by medical software at minimum. They implemented a use case of Nuclear Medicine where the entire process was run by a reinforcement learning agent without any human involvement for full automation and without exposing medical staff to radiation.

Naderi et al. [36] suggested using deep reinforcement learning in robotics, where they worked on handling the movement of the multiple limbs of a climbing humanoid robot.

A deep reinforcement learning solution, namely Double Q-Learning for speed control in autonomous vehicles was proposed by Zhang et al. [37]. The agent decides the

appropriate speed based on solving a Gym environment that uses real data.

CHAPTER 3

METHODOLOGY

This thesis proposes a novel approach for intrusion detection in network systems, which utilizes a deep reinforcement learning algorithm and network traffic in the NSL-KDD intrusion detection dataset. This approach can be later deployed into different network intrusion detection systems that are more flexible, resilient, and efficient. One of our goals is to pave the way for other researchers to implement their approaches using the custom-built environment. Another aim is to gain insight into the performance of the developed deep reinforcement learning approach in intrusion detection.

The reinforcement learning approach used for learning in this thesis is Deep Q-Learning. This approach includes building an RL agent using a reinforcement learning algorithm known as Deep Q-Network. It is used as part of the agent's structure to detect intrusions. Moreover, the approach includes creating and setting up an environment to guide the agent's learning process. The environment created uses the network traffic in the NSL-KDD dataset and was inspired by an environment introduced by Hari Koduvely [30]. The environment was custom-built to suit the RL approach developed in this thesis.

The baseline solution implemented in this thesis is a supervised deep learning approach, namely Long-Short Term Memory neural network structure. This solution is trained on the same dataset used in the reinforcement learning approach for performance comparison purposes.

3.1 Reinforcement Learning

Reinforcement learning is an evolutionary machine learning approach that simulates the learning process in living organisms [38]. Learning is a process in which living organisms increase their knowledge in the scope of different tasks by accumulating knowledge that enhances their capability on how to perform certain tasks better as their knowledge increases. Similarly, machine learning algorithms can emulate living organisms' learning process through their exposure to data, which in this scenario simulates the accumulative knowledge living organisms acquire overtime.

In reinforcement learning, the algorithm does not need to have prior domain knowledge and can learn over time by trial and error [39]. The algorithm in that scenario represents the brain in living organisms. The main purpose in reinforcement learning is trying to develop that brain, which in this case is called an agent, and this is where the algorithm resides.

The main purpose of the agent in reinforcement learning is to sum up the reward over time and get as many positive rewards as possible. This process simulates doing correct actions in certain tasks given the prior knowledge accumulated in the brain of a living organism.

In reinforcement learning, the agent interacts with an observation acquired from the environment (see Figure 3.1), which is called a state. In living organisms, such state represents the interaction with the real world whereby experience or knowledge is gained. Building a well-structured reinforcement learning environment is paramount for optimizing the agent's learning process and performance in a specific domain.

In the context of this research, the environment includes the network intrusion detection dataset and evaluates the agent's actions (right or wrong), the observation represents the network traffic, and the agent represents the detection mechanism that decides whether the network traffic is an intrusion or not.

In the upcoming sections, we will explain some reinforcement learning concepts and its four main elements, which are agent, environment, state, and reward, in the scope of this research.

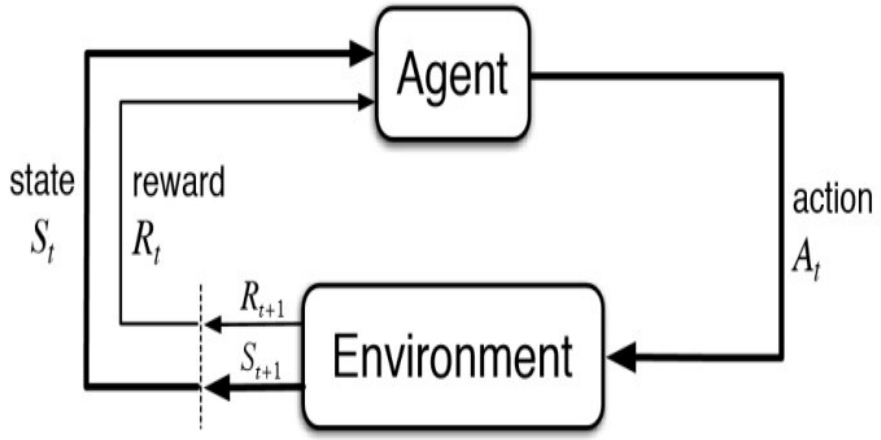

Figure 3.1 Reinforcement Learning [40]

3.2 Markov Decision Process

Markov Decision Process, which can also referred to as MDP, is a sequential decision-making process that depends on several factors. In the context of reinforcement learning, it depends on five factors: the environment, the agent, all the states provided by the environment, all possible actions that the agent can perform, and all the rewards that the agent acquires from the environment after performing an action.

The sequential process starts by the agent observing a state from the environment. Depending on the state, the agent selects an action to perform. Afterwards, the agent gets a reward from the environment, and then another state is initiated. This entire process can be optimized in order for the agent to get the maximum accumulative reward, not just the immediate reward.

In other words, we are trying to map state-action pairs to rewards as represented in the following Equation 3.1:

$$f(S_t, A_t) = R_{t+1} \tag{3.1}$$

3.3 Q-Function and Q-Values

Q-Function is an indication of the quality of an action taken by the agent given a certain state. Therefore, it is referred to as the action-value function. The output of the function for any state-action is known as the Q-value. In Q-Learning, a Q-Table is built containing the state-action pairs and their corresponding Q-Values. In simple words, it tells us which actions to take in which states to get the highest reward. In order to calculate the Q-values, we use an equation called the Bellman equation which is shown in the following Equation 3.2:

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right] \tag{3.2}$$

The Bellman equation consists of two parts: the immediate reward and the future Q-Values. This equation can estimate the total reward we can get from some states onwards if we are going to follow this particular behavior. Simply put, it is used to optimize behavior by looking one step ahead.

In deep reinforcement learning, which is the approach used in this thesis, we follow another approach called Deep Q-Learning, in which we use neural networks to work as a function approximator to obtain the optimal Q-values as shown in Figure 3.2:

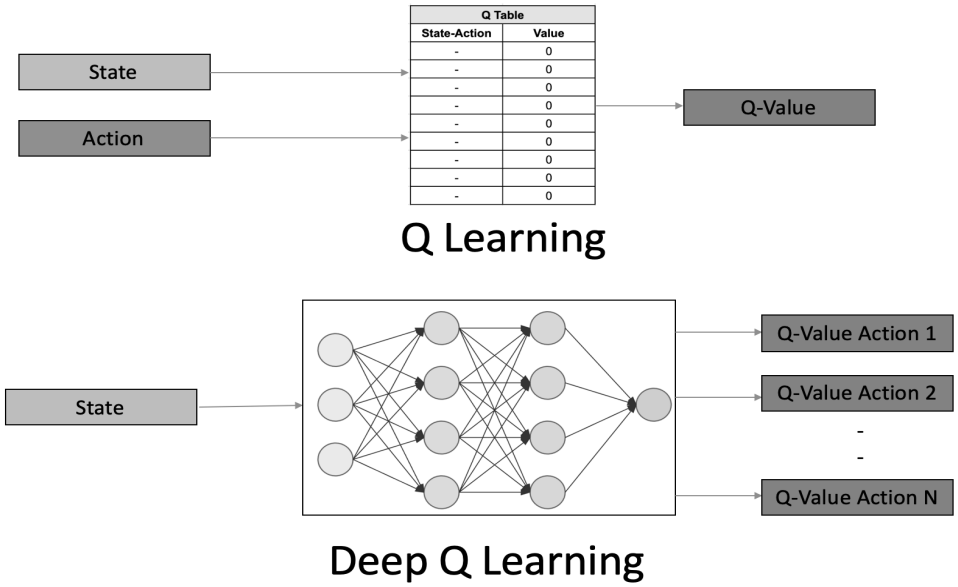

Figure 3.2 Q-Learning Vs Deep Q-Learning [41]

3.4 Deep Reinforcement Learning Agent

A reinforcement learning agent is the brain where the algorithm resides. The agent's purpose is to select the appropriate action given a certain state to maximize the overall reward. This indicates that it is taking the best action available. Deep Q-learning is the chosen approach used in this research because using Q-learning to estimate the Q-values by constructing a Q-table of state-action pairs will be unsuitable given the large number of observations or states observed by the agent. In other words, we use neural networks to map states to action-Q-value pairs instead of mapping state-action pairs to Q-values.

Deep Q-learning uses two separate neural networks: the first one is called the main network or the policy network, and the second one is called the target network. The reason for using two separate neural networks is that when we feed a state to the neural network in order to obtain the appropriate action-Q-value pairs, the weights of the network are updated. With that constant change, this mapping process will be impossible because we are chasing a dynamic target that contradicts the whole purpose of mapping states to action-Q-value pairs. Instead, we use a second neural network with initially the same weights as those of the main network. The target

network is used to obtain the target Q-values as shown in Figure 3.3, and then, the weights will be updated to match the main network ones every certain number of iterations to ensure that the learning process is functioning properly [42].

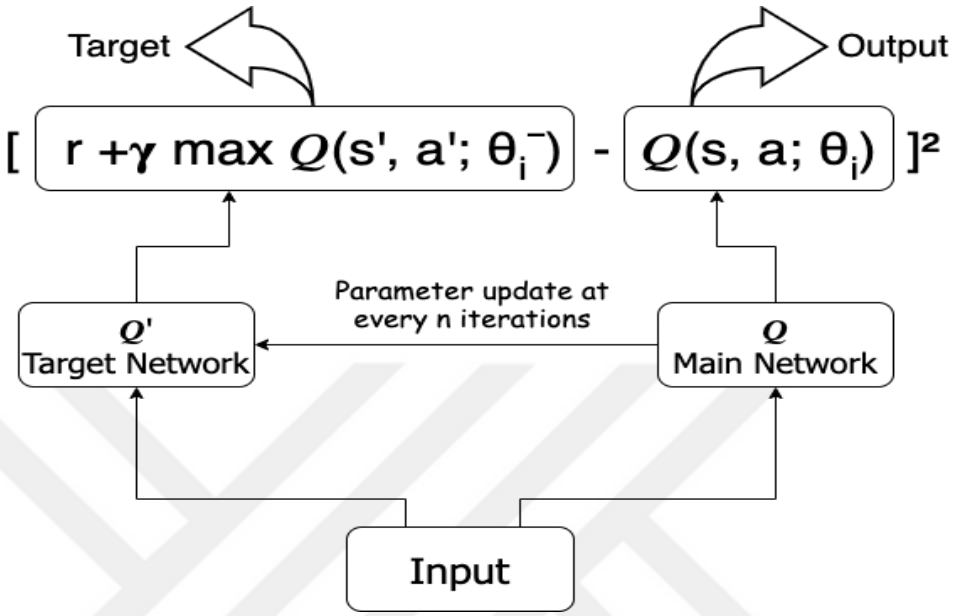

Figure 3.3 Target Q-Values

3.4.1 Exploration Vs Exploitation

For the reinforcement learning agent to take any action, it follows a strategy called an epsilon greedy strategy because it needs to explore the environment by initially performing random actions, and later on it decides the best actions required in the long term. This introduces us to the exploration-exploitation trade-off, which is when the agent goes through the environment, and later, it gets to a point when it does not require to explore the environment anymore. We start by setting an exploration rate which is initially $\epsilon = 1$ and will decay by a certain rate at the beginning of each training loop, or in this context called episode. Then, a random number between ”0” and ”1” will be produced as a threshold. If that threshold is greater than ϵ, the agent will start exploitation to select its next action [43].

In this thesis, we have two actions between which the agent can choose. The actions

are represented by the numerical values “1” and ”0”, which indicate the state of the observed network traffic whether being an intrusion or not respectively.

3.4.2 Experience Replay

In deep Q-learning, a new concept called replay memory comes into place, where the agent’s experience e_t at time t is composed of the state, action, reward, and next state as shown in Equation 3.3 and will be stored in a dataset known as replay memory.

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1}) \tag{3.3}$$

The experience replay concept makes it flexible to sample from less correlated data and optimally utilize the data. This process helps train the neural network in the most efficient way and theoretically produce better results. In machine learning, a group of samples are known as a batch, and each sample represents an experience. A batch has a specific fixed sample size which is stated at the beginning of the training process. This concept can be visualized in Figure 3.4:

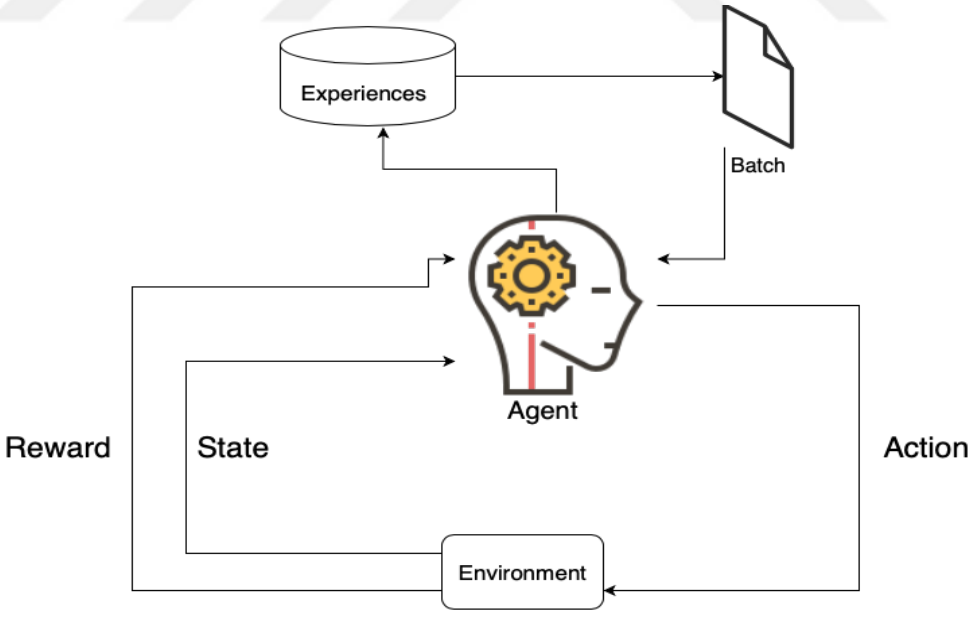

Figure 3.4 Experience Replay

In Table 3.1 a detailed explanation of the deep Q-learning algorithm is shown.

Table 3.1 Deep Q-learning Algorithm

Step	Detail
	1. Initialize replay memory D to capacity N. 2. Initialize the Main Network (policy network) with random weights. 3. Replicate the Main Network and call it the target network. 4. **For** each training iteration get initial observation from the environment. Preprocess the observation **For** each episode: Select an action via exploration or exploitation Execute selected action Observe reward and next state. Preprocess the next state Store experience in replay memory. Sample random batch from replay memory. Feed batch to Main network. Calculate loss between output Q-values and target Q-values. Update weights in the policy network to minimize loss. After *n* time steps, weights in the target network are updated to the weights in the policy network

PyTorch open-source machine learning framework [44] was used to implement the reinforcement learning agent. Some preprocessing was required for the raw observation provided by the Gym environment to fit the PyTorch operated deep neural network. The Gym environment structure and work flow will be discussed in the upcoming section. The entire operational structure and workflow of the reinforcement learning agent can be observed below in Figure 3.5:

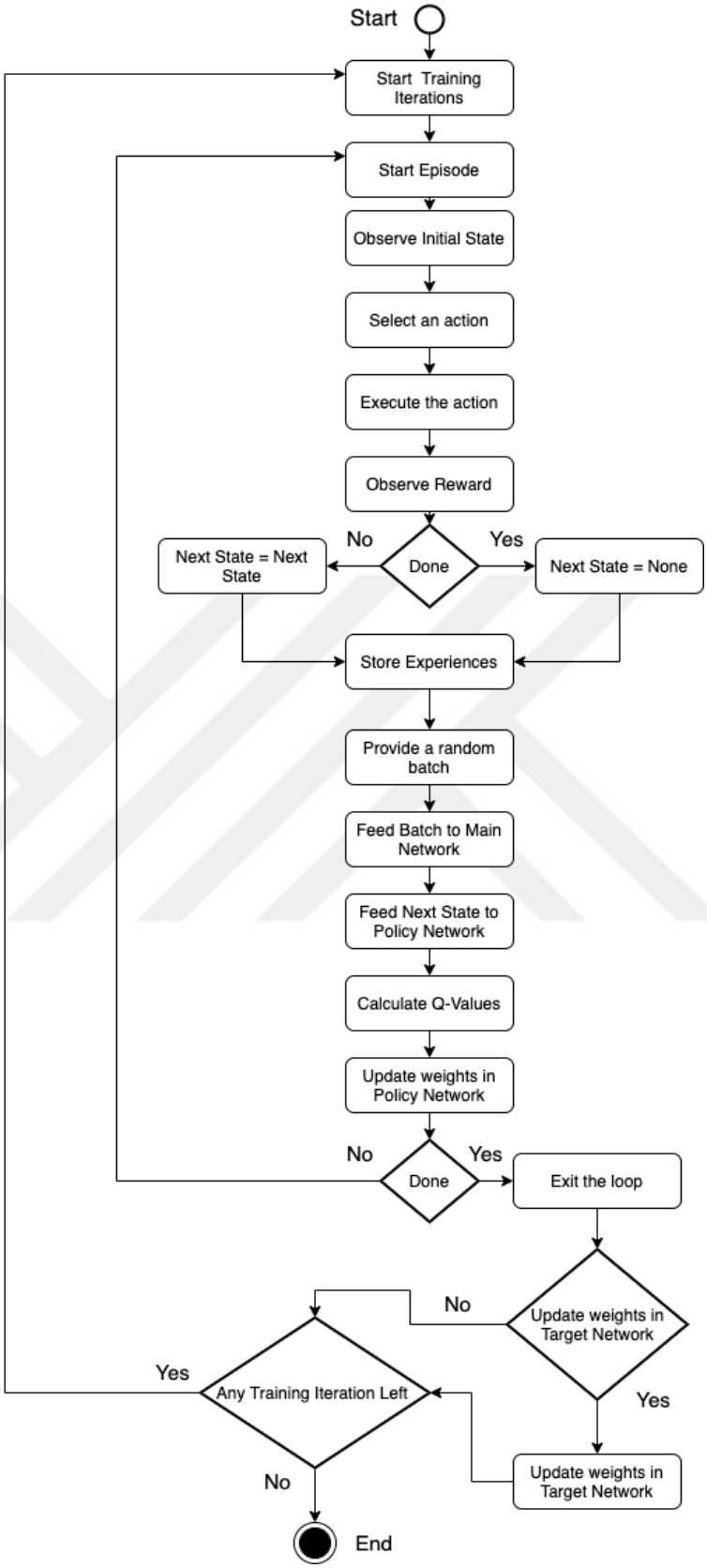

Figure 3.5 Activity Chart of Reinforcement Learning Agent

3.5 Gym Environment

The environment is a very important element of the reinforcement learning process. Building an expressive environment is a necessity for the agent to be properly trained and perform the assigned task efficiently. It can also be used to train different reinforcement learning models. In order to implement this environment concept, a specifically developed toolkit called Gym is used to create reinforcement learning environments, which is referred to as OpenAI Gym toolkit [45]. A custom-built environment was introduced to suit the approach developed in this thesis, which is Deep Q-Learning. The proposed Gym environment provides network traffic, and the agent's duty is to interact with this environment by selecting an action after being exposed to different network traffics and deciding whether this network traffic is an intrusion or not. In the upcoming sections, we will dive deeper into the structure of the Gym environment.

3.5.1 Gym Environment Structure

The gym environment structure is based on four different functions that are all in the same class: initialization, step, reset, and render. The environment will be referred to as an object called *env*.

The initialization function is composed of many parameters. The two most relevant parameters are the action space, which is the available actions for the agent to take, and the observation space, which includes network traffic's features that will be observed by the agent. The step function will observe the action taken by the agent and return four different parameters: next state, reward, done, and info. These parameters represent the next state that the agent will observe, the current reward, a Boolean value indicating whether this episode is over or not, and some additional diagnostic information. The reset function provides a new random initial state. The render function outputs a graphical representation of the current situation in the environment. In this context, however, it is irrelevant because we do not have a graphical representation for the Gym environment. A code snippet is shown in Figure 3.6 for further explanation. Moreover, all the possible actions, state labels, and rewards are provided in Table 3.2

to deepen the understanding of the previous concepts.

```
ob0 = env.reset() # sample environment state, return first observation
a0 = agent.act(ob0) # agent chooses first action
ob1, rew0, done0, info0 = env.step(a0) # environment returns observation,
# reward, and boolean flag indicating if the episode is complete.
a1 = agent.act(ob1)
ob2, rew1, done1, info1 = env.step(a1)
...
a99 = agent.act(o99)
ob100, rew99, done99, info2 = env.step(a99)
# done99 == True  =>  terminal
```

Figure 3.6 Single Episode with 100 Timesteps Provided by G. Brockman et al. [45]

Table 3.2 State, Action and Reward as Represented in the Environment

State Label	Action	Reward
Normal	0	1
Normal	1	-1
Intrusion	0	-1
Intrusion	1	1

First, the reset function is used to obtain the initial observation from the environment, which is then passed to the agent. Second, the agent will choose an action from the available ones. Third, the action chosen by the agent is passed to the environment step function. The step function will return the next observation, the reward for the agent's past action, a Boolean value indicating whether the training episode is over or not, and some diagnostic information. This training process continues until it is terminated. In Figure 3.7, a more realistic code snippet is shown to give a clearer understanding. The only difference is that the custom-built Gym environment is initiated at the beginning of the code. Table 3.2 explains the reward system given by the environment in correspondence with the action taken by the agent. When the agent observes a state provided by the environment, it responds by choosing an action from the available actions. The agent can either choose "0" or "1" which indicates whether

this network traffic is a normal or an intrusion respectively. Given this action, the environment judges the agent behavior by matching it with the label associated with each network traffic in the dataset. Thereupon, the environment assigns either a positive reward "1" or a negative reward "-1" in response to the correct or incorrect action of the agent respectively.

```
env = gym.make('network_intrusion_env')
obs = env.reset() #Enviroment returns intial observation
action1 = agent.select_action(obs) #Agent choosing its first action
next_state, reward, done, info = env.step(action1) #Enviroment returns
the next state or observation, reward, a boolean value and additional info
...
```

Figure 3.7 Single Episode with 1 Timestep of Our Intrusion Detection RL Code

3.5.2 NSL-KDD Dataset

The NSL-KDD dataset, which is used by the Gym environment to provide observations to the agent, was obtained from the website of the Canadian Institute of Cyber Security, University of New Brunswick [46]. The dataset is a modified version of the original KDDCUP'99 dataset which was originally introduced in a competition with the goal of gathering network traffic data. This dataset is one of the largest and most researched dataset, and it serves as a benchmark for today's modern intrusion detection systems. Its training set contains 125,973 records of which 67,343 represent normal network traffic and 58,630 represent intrusions. Its testing set contains 22,544 records of which 9,711 represent normal network traffic and 12,833 represent intrusions [47].

The dataset contains four main attack categories (DoS, Probe, U2R, and R2L) and 39 different subcategories as described in Table 3.3.

Table 3.3 Main Attack Types and their Subcategories in NSL-KDD Dataset

	Attack Types	**Main Attack Category**
1	Back	dos
2	buffer_overflow	u2r
3	ftp_write	r2l
4	guess_passwd	r2l
5	imap	r2l
6	ipsweep	probe
7	land	dos
8	loadmodule	u2r
9	multihop	r2l
10	neptune	dos
11	nmap	probe
12	perl	u2r
13	phf	r2l
14	pod	dos
15	portsweep	probe
16	rootkit	u2r
17	satan	probe
18	smurf	dos
19	spy	r2l
20	teardrop	dos
21	warezclient	r2l
22	warezmaster	r2l

Moreover, each record of the dataset represents 41 features of network traffic and a label as described in Table 3.4 below.

Table 3.4 The Different 42 Feature in the NSL-KDD Dataset

	Feature	**Type**
1	duration	Continuous
2	protocol_type	Categorical
3	service	Categorical
4	flag	Categorical
5	src_bytes	Continuous
6	dst_bytes	Continuous
7	land	Binary
8	wrong_fragment	Discrete
9	urgent	Discrete
10	hot	Continuous
11	num_failed_logins	Continuous
12	logged_in	Binary
13	num_compromised	Continuous
14	root_shell	Binary
15	su_attempted	Discrete
16	num_root	Continuous
17	num_file_creations	Continuous
18	num_shells	Continuous
19	num_access_files	Continuous
20	num_outbound_cmds	Binary
21	is_host_login	Binary
22	is_guest_login	Binary
23	count	Discrete
24	srv_count	Discrete
25	serror_rate	Discrete
26	srv_serror_rate	Discrete
27	rerror_rate	Discrete
28	srv_rerror_rate	Discrete
29	same_srv_rate	Discrete
30	diff_srv_rate	Discrete
31	srv_diff_host_rate	Discrete
32	dst_host_count	Discrete
33	dst_host_srv_count	Discrete
34	dst_host_same_srv_rate	Discrete
35	dst_host_diff_srv_rate	Discrete
36	dst_host_same_src_port_rate	Discrete
37	dst_host_srv_diff_host_rate	Discrete
38	dst_host_serror_rate	Discrete
39	dst_host_srv_serror_rate	Discrete
40	dst_host_rerror_rate	Discrete
41	dst_host_srv_rerror_rate	Discrete
42	Label	Discrete

Given the multiple features in the NSL-KDD dataset and the different categorical values, it will be infeasible to use this data in its raw form. Therefore, data preprocessing is performed by using the one-hot encoding approach. This approach is based on turning the categorical value into a binary value. The first step includes mapping the categorical values to integer values. Then, a binary value of "1" will be assigned for each relevant category and a "0" for the irrelevant categories.

For further clarification, an example will be given using the data features introduced in the previous section. The "Protocol Type" feature has three possible categorical values: 'icmp', 'tcp', and 'udp'. It can be mapped to three different integer values, and a binary value "1" will be assigned to the existing protocol in the network traffic. Then, a binary value "0" will be assigned to the non-existing protocol as depicted in Figure 3.8.

Protocol		icmp	tcp	udp
1	Integer value	0	1	2
2	Part of the encoded network traffic that has icmp protocol	1	0	0

Figure 3.8 One Hot Encoding of Protocol Type Feature

To sum up what has been explained in this section, the acquired observation from the environment is a binary representation of the network traffic encoded using the one-hot vector encoding approach.

3.6 Deep Learning Baseline Solution Using LSTM

A supervised learning approach such as deep learning using Long Short-Term Memory neural network is a necessity for improving rule-based intrusion detection systems. Generally, unsupervised learning approaches are more feasible to implement. However, due to the enormous amount of features and changes in the network traffic, the unsupervised learning approaches are ineffective. Therefore, for intrusion detection supervised learning is a more suitable approach than unsupervised learning.

The Long Short-Term Memory neural network architecture, which was introduced in [48], is the most suitable one because of its unique architecture that showed success when dealing with time-series data such as handwriting, speech, and text recognition, Natural Language Processing (NLP), and weather forecasting as shown in Figure 3.9 [49].

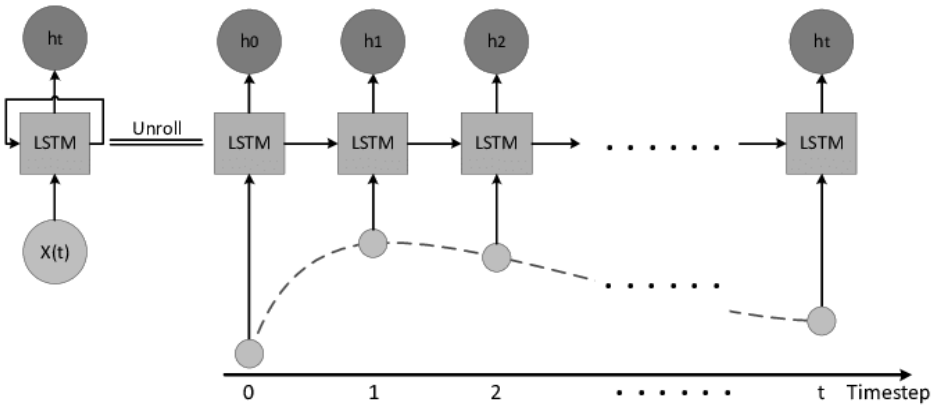

Figure 3.9 LSTM Handling time-series Data [49]

The baseline model implemented was trained on labeled training data using this architecture to detect intrusions in network traffic and was then evaluated using the unlabeled testing data. The dataset used is the NSL-KDD dataset, which was previously introduced.

3.6.1 LSTM Baseline Solution Data Preprocessing

For the deep learning approach, using the data in its raw form will be impossible because feeding categorical values to the deep neural network is not applicable. Instead, we will encode the raw data by using an approach called label encoding. Similar to the one-hot encoder approach introduced in the previous section, it assigns an integer numerical value instead of the categorical value. The dataset was processed to label all attack types as "intrusion", which are assigned the same integer numerical value regardless of the attack type. In the encoding process, the intrusion network traffic was assigned the value "0" and normal network traffic was assigned the value "1" as shown in Figure 3.10.

Network Traffic Label	Normal	Intrusion
Raw	Normal	Neptune
Preprocessed	Normal	Intrusion
Preprocessing + Encoding	1	0

Figure 3.10 Label Encoding

Since the data has different integral numerical values range, it should be normalized to make all the numerical values fit into a common scale without changing the difference between the values. This normalization process will enhance the training, as no particular feature in the data regardless of its large numerical value will further influence the training process. The last preprocessing step required is reshaping the dataset to the appropriate size and dimension to fit the neural network.

3.6.2 LSTM Model Architecture

The unique architecture of the LSTM neural network allows the data to be stored in a memory called a memory cell for a certain period of time, which is composed of gates that control three aspects: when data enters the memory (input gate), when it forgets the data (forget gate), and when it outputs the data (output gate). This can be seen clearly in Figure 3.11 [50].

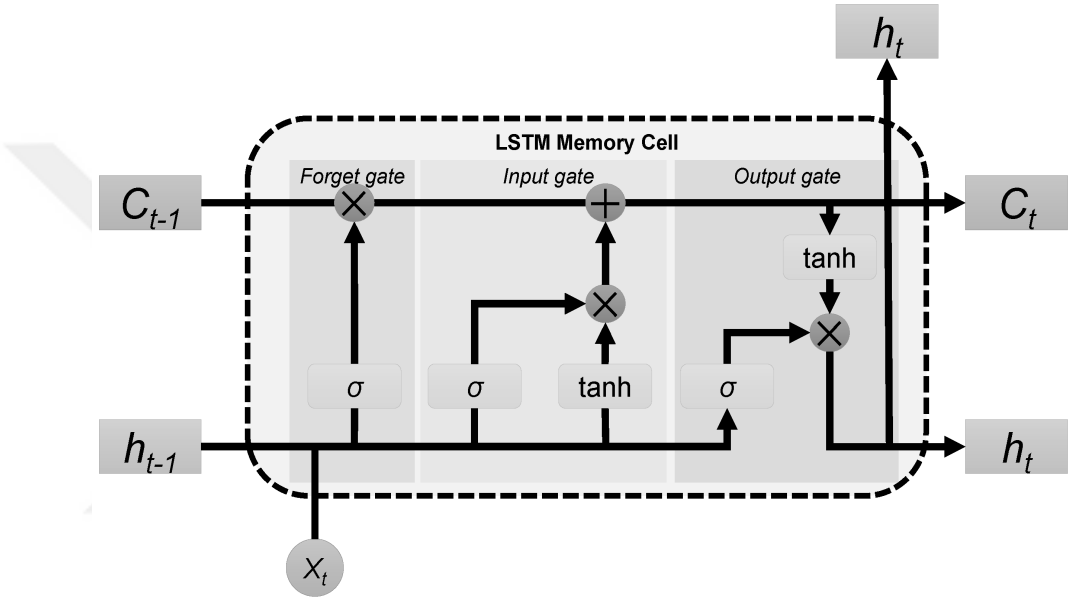

Figure 3.11 LSTM Architecture [50]

X_t represents the input, C_t represents the cell state, and h_t, known as the state, represents the output in Equation 3.4 that will be fed alongside the new input to the next step.

$$h_t = f(h_{t-1}, x_t) \tag{3.4}$$

The input gate has two functions: the sigmoid function and the tanh function. The sigmoid function makes a binary decision of whether the input values should pass or not, and the tanh function assigns weights to the successfully passed inputs to determine their importance. The forget gate has one sigmoid function which decides

which of the previous outputs and the current inputs should be kept and which should be discarded. Similarly, the output gate has the two sigmoid and tanh functions that perform the same mentioned tasks in the input gate. Then, the output from the tanh function will be multiplied by the output of the sigmoid function.

The LSTM neural network architecture was implemented using the TensorFlow open-source machine learning framework [51] alongside Scikit-Learn and Keras.

CHAPTER 4

EVALUATION AND RESULTS

In this chapter, the evaluation approach, metric used, and the the results of the experiments conducted using the new approach will be discussed. The experiments were conducted using different number of training iterations, various neural network structures including different numbers of hidden layers and neurons, and multiple batch sizes. Additionally, we used 50% of the data in some experiments and 100% in others to prove that reinforcement learning can handle attack types and function properly even when there is an increase in the number of attack types represented in the dataset, leading to the conclusion that RL can generalize. The baseline LSTM solution's experiments were conducted using different neural network structures. The reason for the variation proposed in the experiments is that obtaining an improvised model is not an exact formula but rather based on a trial-and-error approach. At the end of the chapter, a general comparison with other reinforcement learning approaches implementing the NSL-KDD dataset from the literature is discussed in order to demonstrate the performance comparison with the approach introduced in this thesis.

4.1 Experimental Settings

Reinforcement learning experiments were conducted using PyTorch machine learning framework and OpenAI Gym library. The baseline supervised learning solution's experiments were conducted using TensorFlow machine learning framework integrated with Scikit-Learn and Keras.

- Hardware
  - GPU: Radeon Pro 555X/4GB/GDDR5
  - RAM: 16GB/2400MHz/DDR4
  - CPU: Intel Core i7-8750H 2.2GHz x 6
- Software Versions
  - OS: macOS Big Sur 11.2
  - IDE: Visual Studio Code 1.53.0
  - Python: 3.7.4
  - PyTorch: 1.6.0
  - OpenAI Gym: 0.18.0
  - TensorFlow: 2.2.0
  - Scikit-learn: 0.23.1
  - Keras: 2.4.3

4.2 Evaluation Metrics

One of the best evaluation methods used for classification prediction models is confusion matrix metrics including F1 score (Equation 4.1), accuracy (Equation 4.2), precision (Equation 4.3), and recall (Equation 4.4) [52].

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{4.1}$$

$$Recall = \left(\frac{TruePositive}{TruePositive + FalseNegative} \right) \tag{4.2}$$

$$Accuracy = \left(\frac{TruePositive + TrueNegative}{TP + FN + FP + TN} \right) \tag{4.3}$$

$$Precision = \left(\frac{TruePositive}{TruePositive + FalsePositive} \right) \tag{4.4}$$

The confusion matrix [52] shown in Figure 4.1 can be used for classification problems to show the actual classification (label) and the predicated one. Since all of the implemented approaches in this research use binary classification, confusion matrix metrics are considered the optimal option for evaluation. Although using confusion matrix metrics is not suitable for reinforcement learning model evaluation, we can utilize the confusion matrix metrics for our reinforcement learning approach since the data is labeled. The previous four equations were derived from the confusion matrix, which represents F1 score, precision, recall (sensitivity), and accuracy of the evaluated model in terms of True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN).

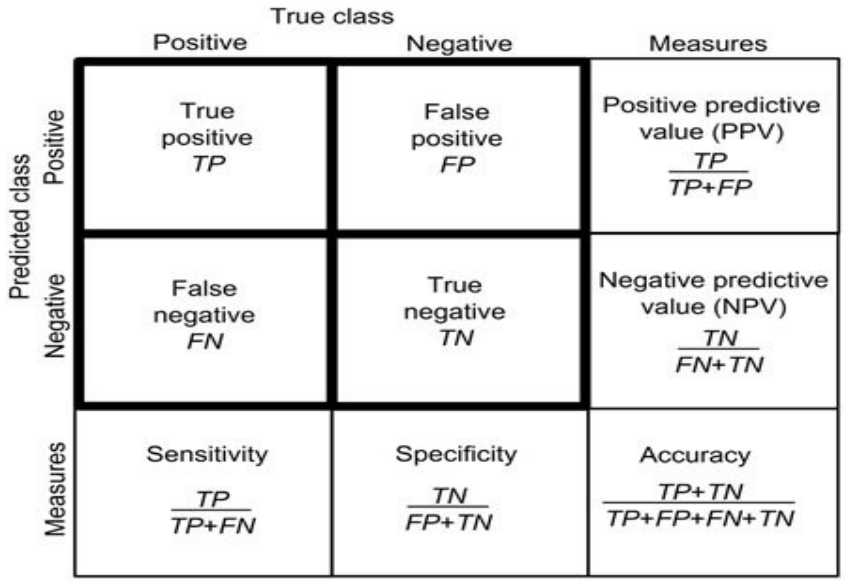

Figure 4.1 Confusion Matrix [52]

In the context of this research, TP means that the model evaluated the network traffic as an intrusion, and it is actually an intrusion. On the other hand, FP means that the model evaluated the network traffic as an intrusion, but it is not an intrusion. The TN that the model evaluated the network traffic as not an intrusion, and it is truly not an intrusion. Finally, FN indicates that the model evaluated the network traffic as not intrusion, however, it is actually an intrusion.

Each one of the four metrics can answer a question about the model performance. For example, the precision answers the question of how many of all the network traffic evaluated as an intrusion that were actually intrusions. The recall or True Positive Rate (TPR) answers the questions of how many intrusions were evaluated correctly and what will be the cost or the consequence if an actual intrusion was not detected. Higher recall value indicates that the model is evaluating more accurately. The accuracy is the ratio between the correctly predicted intrusions and the total number of intrusions. F1 shows the overall performance of the model and if the FN and FP predictions are affecting the model performance. The higher the F1 value, the better the performance.

4.3 Reinforcement Learning Model Evaluation

Reinforcement learning model evaluation differs from other machine learning approaches in its unique structure. In order to evaluate a reinforcement learning model, we have two approaches: the performance of the policy estimated by the model and the learning curve that indicates the improvement of the agent's actions over time. The learning curve approach was followed in this thesis. It can be identified as the accumulative reward as a function of the number of episodes. In other words, we are plotting the accumulative reward acquired by the RL agent over time and showing if it is increasing or decreasing. If the accumulative reward is increasing, it indicates that the RL agent's actions are improving during the learning process, which is the desired output.

4.4 Reinforcement Learning Experimental Results

In Figure 4.2 below, the learning curve of the RL agent is plotted. It is showing a great improvement in the agent's performance over time, which validates the model. This means that the accumulative reward that the agent is receiving is increasing over time. This indicates that the agent's ability to choose the most appropriate action by maximizing its gain is improving over time.

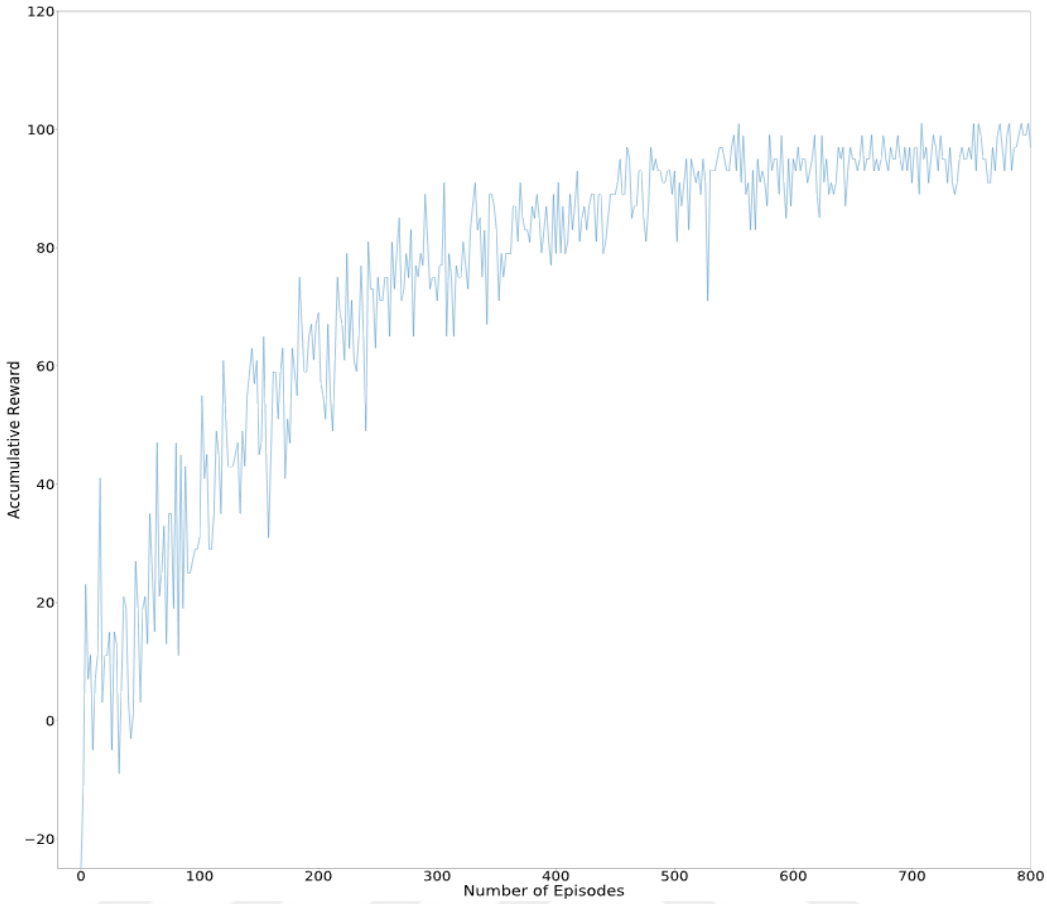

Figure 4.2 RL Agent Learning Curve

The results of the Deep Q reinforcement learning model are shown below. Although many experiments were conducted, only the top-performing ones are expressed. The other experiments were attempts to reach the optimal hyper-parameters. An experiment with only 50% of the dataset is performed to show the generalization property of RL when compared to the full dataset experiment. Another experiment is shown using different hyper-parameters to prove that the RL model's hyper-parameters and neural network structure can affect its performance drastically.

The first conducted experiment in RL was utilizing the novel reinforcement learning model developed in this research. We started by using 50% of the data with 400 training iterations, 100 steps per episode, and a batch size of 64. We used one hidden layer of size 50*10, and the results can be shown in Figure 4.3. The accuracy reached 86.80%.

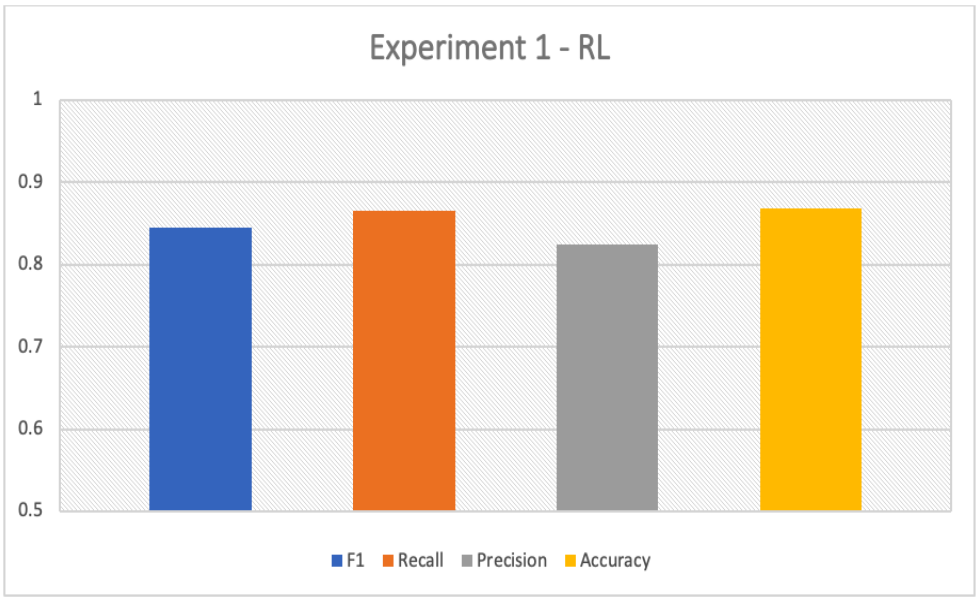

Figure 4.3 Results of Experiment 1 of Deep Q-Learning RL Model

In the second conducted RL experiment, we used 100% of the data with 800 training iterations, 100 steps per episode, and a batch size of 64. We used one hidden layer of size 50*10, and the results can be shown in Figure 4.4. The accuracy was about 93.12%.

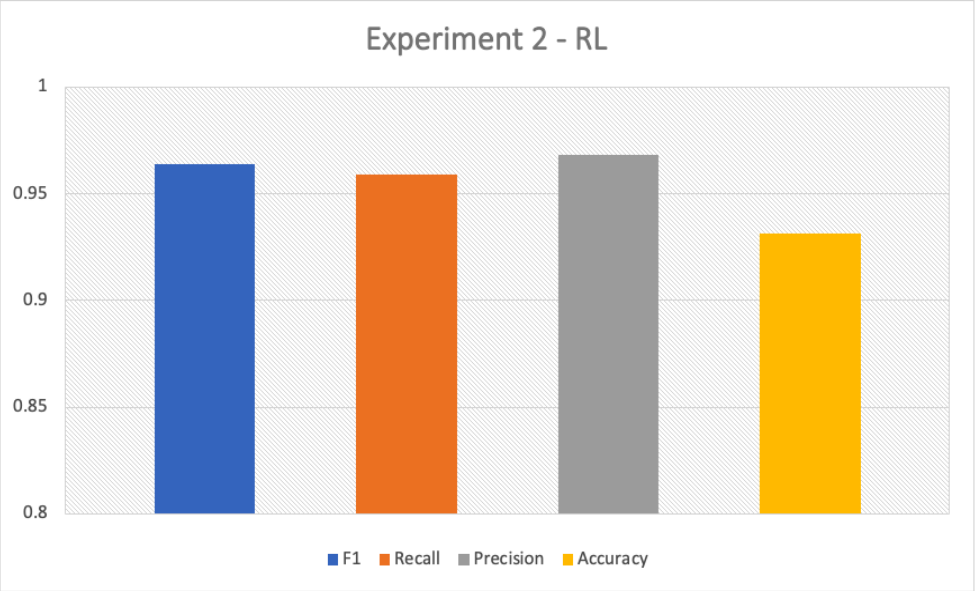

Figure 4.4 Results of Experiment 2 of Deep Q-Learning RL Model

In the third conducted RL experiment, we used 100% of the data with 800 training iterations, 100 steps per episode, and a batch size of 64. We used two hidden layers of size 150*10, and the results can be shown in Figure 4.5. The accuracy was about 91.63%. This experiment is shown to prove that RL performance can be affected by its hyper-parameters and its neural network structure.

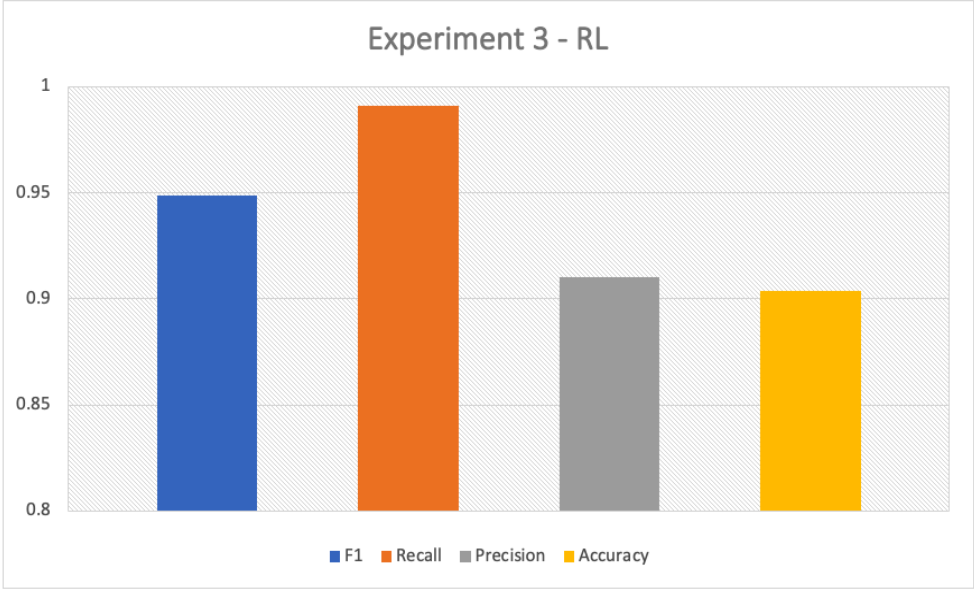

Figure 4.5 Results of Experiment 3 of Deep Q-Learning RL Model

4.5 LSTM Experimental Results

In this section, the conducted experiments on the LSTM are expressed. The idea of including the LSTM experimental results is to compare a supervised learning approach that serves as a baseline with the proposed reinforcement learning approach. As mentioned earlier, the other experiments were attempts to reach the optimal hyper-parameters.

In the top-performing experiment conducted on the LSTM solution (baseline), we used 100% of the data with 30 epochs, four LSTM layers, four Dropouts, and one dense layer of unit size 2. The results can be shown in Figure 4.6. The accuracy reached 76.81%.

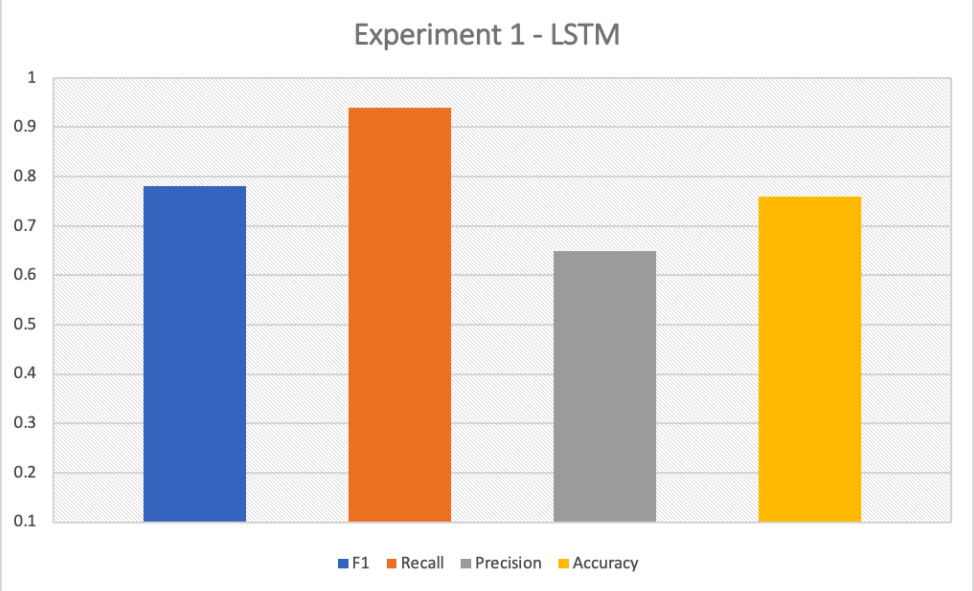

Figure 4.6 Results of Experiment 1 of LSTM Model

4.6 Experimental Results

An aggregation of all the conducted experiments' results can be shown in Table 4.1 to give an overview.

Table 4.1 Aggregated Results Comparison

Experiment	F1 (%)	Recall (%)	Precision (%)	Accuracy (%)
RL – Experiment 1	84.45	86.55	82.45	86.80
RL – Experiment 2	96.36	95.90	96.83	93.12
RL – Experiment 3	94.88	99.11	91.01	90.39
LSTM – Experiment 1	78.3	97.3	65.5	76.8

4.7 Performance Comparison With the Baseline Approach

Table 4.2 is an aggregation of the top-performing experimental results, including the developed RL approach and the implemented LSTM baseline solution. The table below shows the superiority of the RL approach in every aspect when compared to the LSTM approach.

Table 4.2 Aggregated Results Comparison

Experiment	F1 (%)	Recall (%)	Precision (%)	Accuracy (%)
RL – Experiment 2	96.36	95.90	96.83	93.12
LSTM – Experiment 1	78.3	97.3	65.5	76.8

4.8 Deep Dive Into the Results

From the results provided by the experiments, we can draw some conclusions. By comparing Experiment 1-RL and Experiment 2-RL in which we used 50% and 100% of the dataset respectively, we can see that the accuracy increased slightly. This shows that RL can generalize when dealing with many intrusion types and forms. Another conclusion was made by comparing Experiment 2-RL with Experiment 3-RL. We noticed a drop in the accuracy after changing the hyper-parameters, implying that the RL model's performance can be affected by the structure of its neural network and its hyper-parameters.

By comparing the recall and precision metrics of the Deep Q-Learning model and LSTM model, we can draw further conclusions. A high recall means that the number of FN predictions is low and high precision means that the FP rate is low, and vice versa. In intrusion detection systems, a high FP rate will render the system unrealistic and will cause unnecessary panic, and if the FN rate is high, a lot of intrusions will go undetected. A well-balanced system will have a ratio of a low FP rate to a low FN rate, which means that both precision and recall must be high. By observing the results of Experiment 2-RL, it is obvious from the recall (95.90%) and precision (96.83%) metrics that the model has a good balance between the FN rate and the FP rate, and both are low, which is preferred. However, looking at the results of Experiment 1-LSTM, the recall (97.3%) and precision (78.3%) have a large gap between them, and the model has a high FP rate and a low FN rate, which is undesirable in an intrusion detection system.

4.9 Performance Comparison With Others Work

Table 4.3 is an aggregation of the top-performing experimental results of the RL agent alongside other results collected from related researches in the literature. The related researches implemented RL approaches such as Adversarial/Multi Agent Reinforcement Learning using Deep Q-Learning (AE-DQN) [53], Deep Q-Network (DQN) [54], Double Deep Q-Network (DDQN) [55], Adversarial Environment Reinforcement Learning (AE-Rl) [56], and Asynchronous Advantage Actor Critic (A3C) [56] alongside the same NSL-KDD dataset. The table also expresses the value added by this research in enhancing and improving the reinforcement learning agent's performance in intrusion detection as shown.

Table 4.3 Aggregated Results Comparison

Approach	F1 (%)	Recall (%)	Precision (%)	Accuracy (%)
Reinforcement Learning (developed in this thesis)	96.36	95.9	96.83	93.12
DQN (Manuel Lopez-Martin et al. [32])	89.35	89.37	89.33	87.87
DDQN (Manuel Lopez-Martin et al. [32])	91.2	93.03	89.44	89.78
Policy Gradient (Manuel Lopez-Martin et al. [32])	79.09	70.67	89.80	78.73
Actor Critic (Manuel Lopez-Martin et al. [32])	81.11	72.51	92.03	80.78
AE-DQN (E. Suwannalai et al. [53])	79	-	-	80
DQN (Ying-Feng Hsu et al. [54])	-	90.2	92.8	91.4
DDQN (Xiangyu Ma et al. [55])	82.43	82.09	84.11	82.09
AE-RL (Caminero et al. [56])	79.4	80.16	79.74	80.16
DDQN (Caminero et al. [56])	68.98	73.72	66.61	73.72
A3C (Caminero et al. [56])	76	80	81	80
Dueling DDQN (Caminero et al. [56])	73.58	77.88	82.06	77.88

CHAPTER 5

CONCLUSION

Due to the massive shift to network-based technologies, cloud computing-based services are becoming the ultimate replacement and solution for handling and providing the infrastructure of computer systems, storage space, and computational power needed by agencies, companies, organizations, institutions, and governments. This transition introduced major concerns regarding the medium or network systems used to deliver these services and resources. Securing the network systems poses a challenge to customers and users of cloud computing services and other network-based technologies. The new attack types and continuous change in attack patterns introduced frequently pose a threat which make it very difficult for traditional cyber-security and intrusion detection systems to keep up with those developments in attacks and the increasing scale of network systems. With those challenges on the rise, machine learning comes into place by introducing a new way to build cyber-security and network intrusion detection systems that are flexible, resilient, and more efficient. Machine learning can be used as the heart of the network intrusion detection systems by learning from relevant data, categorizing, and detecting intrusions in network systems. The use of machine learning can overcome the growing challenges by keeping up with the continuous change in attack patterns or development of new attacks, which is aligned with the objective of this thesis.

In this thesis, we provided an answer to the intrusion problem in modern network systems by introducing and applying a novel reinforcement learning approach and solution for intrusion detection. We also got an insight into the performance and aspects of the novel approach and how this approach has performed compared to other solutions.

Using deep reinforcement learning, we developed a novel approach for intrusion detection, and this approach had two key parts. The first part was the agent, which was built using Deep Q-Learning algorithm and a deep neural network. The second part was the custom-built Gym environment that guided the learning process and provided network traffic from the dataset to the reinforcement learning agent. The dataset used to fuel the learning process was the NSL-KDD dataset, as it is used as a benchmark for most modern intrusion detection systems. A supervised learning solution that uses LSTM was implemented to serve as a baseline. The reinforcement learning approach was implemented using PyTorch open-source machine learning framework, and the baseline supervised deep learning solution was implemented using TensorFlow open-source machine learning framework. Several experiments were carried out using several hyper-parameters and configurations by trial and error to obtain the optimal results. The experiments performed were reported, classified according to their configurations, and compared to other relevant researches.

The superiority of the novel approach, which is our main contribution, was validated and confirmed by comparing its results with the baseline supervised learning solution and other relevant work from the literature. Additionally, the custom-built environment, which is the second contribution, contributed to the model's superiority by having expressive and inclusive features of intrusions. The proposed Deep Q-Learning reinforcement solution achieved the highest performance with an accuracy of 93.1% and it appears to be the most efficient solution among those tested and compared. This high accuracy was achieved as a result of using an expressive Gym environment and a well-tuned model. The LSTM solution which was used as a baseline performed less accurately with an accuracy close to 77%. The obtained results show that the LSTM solution is not sufficiently effective as a modern intrusion detection solution and has a high FP rate, limitations, and inability to generalize. Another conclusion drawn from the observation of RL-Experiments 1, 2, and 3 is that reinforcement learning can generalize detection of the various intrusion types, and its performance can be affected by changing its hyper-parameters and the structure of its neural network.

To sum up, the results obtained by the proposed RL approach were satisfactory and promising, showing that it is superior to other approaches. Therefore, the developed RL solution was accounted as the most suitable for intrusion detection.

As future work, an intrusion detection system can be developed by using the RL approach integrated with the different cloud computing services as they offer humongous data, metrics, and computing power that can be utilized to make the network system more secure.

REFERENCES

[1] "Cloud Market Share," Internet: https://www.statista.com/chart/7994/cloud-market-share/ [Jan. 15, 2021].

[2] "Cloud Infrastructure Market," Internet: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ [Feb. 6, 2021].

[3] J. Armin, B. Thompson, D. Ariu, G. Giacinto, F. Roli, and P. Kijewski, "2020 Cybercrime Economic Costs: No Measure No Solution," in *2015 10th International Conference on Availability, Reliability and Security*. IEEE, 2015, pp. 701–710.

[4] "Net Losses: Estimating the Global Cost of Cybercrime," Internet: https://www.csis.org/analysis/net-losses-estimating-global-cost-cybercrime [Jan. 15, 2021].

[5] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[6] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2018, no. 1, pp. 177–200, 2018.

[7] "The Hidden Costs of Cybercrime," Internet: https://www.csis.org/analysis/hidden-costs-cybercrime [Jan. 15, 2021].

[8] Y. Bouzida and F. Cuppens, "Neural networks vs. decision trees for intrusion detection," in *IEEE/IST Workshop on "Monitoring, Attack Detection and Mitigation (MonAM)*, vol. 28. Citeseer, 2006, p. 29.

[9] I. Aljamal, A. Tekeoğlu, K. Bekiroglu, and S. Sengupta, "Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments," in *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2019, pp. 84–89.

[10] S. Parampottupadam and A.-N. Moldovann, "Cloud-based Real-time Network Intrusion Detection Using Deep Learning," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 2018, pp. 1–8.

[11] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of IoT networks using artificial neural network intrusion detection system," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2016, pp. 1–6.

[12] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach," *Computer Communications*, vol. 98, pp. 52–71, 2017.

[13] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*. IEEE, 2014, pp. 1–6.

[14] M. Sreekesh *et al.*, "A two-tier network based intrusion detection system architecture using machine learning approach," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 42–47.

[15] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, vol. 2. IEEE, 2002, pp. 1702–1707.

[16] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*. IEEE, 1999, pp. 371–377.

[17] W. Lee and S. Stolfo, "Data Mining Approaches for Intrusion Detection," 1998.

[18] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 305–316.

[19] S. Zanero and S. M. Savaresi, "Unsupervised Learning Techniques for an Intrusion Detection System," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, 2004, pp. 412–419.

[20] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of SVM and ANN for intrusion detection," *Computers Operations Research*, vol. 32, no. 10, pp. 2617–2634, 2005.

[21] S. L. Scott, "A Bayesian paradigm for designing intrusion detection systems," *Computational statistics data analysis*, vol. 45, no. 1, pp. 69–83, 2004.

[22] Y. Li and L. Guo, "An active learning based TCM-KNN algorithm for supervised network intrusion detection," *Computers Security*, vol. 26, no. 7-8, pp. 459–467, 2007.

[23] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[24] 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)*. IEEE, 2020, pp. 1–6.

[25] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. De Boer, and G. Narayansamy, "Intrusion Detection System for Internet of Things based on a Machine Learning approach," in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*. IEEE, 2019, pp. 1–6.

[26] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[27] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1285–1298.

[28] H. Sagha, S. B. Shouraki, H. Khasteh, and M. Dehghani, "Real-Time IDS Using Reinforcement Learning," in *2008 Second International Symposium on Intelligent Information Technology Application*, vol. 2. IEEE, 2008, pp. 593–597.

[29] S. Xia, M. Qiu, and H. Jiang, "An adversarial reinforcement learning based system for cyber security," in *2019 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2019, pp. 227–230.

[30] H. Koduvely, "Anomaly Detection through Reinforcement Learning," presented at Ottawa Artificial Intelligence and Machine Learning MeetUp group, Ottawa, ON, Canada, Jan. 29, 2018.

[31] T.-T.-H. Le, Y. Kim, and H. Kim, "Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks," *Applied Sciences*, vol. 9, no. 7, p. 1392, 2019.

[32] 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.

[33] C. Tang, N. Luktarhan, and Y. Zhao, "An Efficient Intrusion Detection Method Based on LightGBM and Autoencoder," *Symmetry*, vol. 12, no. 9, p. 1458, 2020.

[34] A. Jeerige, D. Bein, and A. Verma, "Comparison of Deep Reinforcement Learning Approaches for Intelligent Game Playing," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0366–0371.

[35] G. Paragliola, A. Coronato, M. Naeem, and G. De Pietro, "A Reinforcement Learning-Based Approach for the Risk Management of e-Health Environments: A Case Study," in *2018 14th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. IEEE, 2018, pp. 711–716.

[36] K. Naderi, A. Babadi, S. Roohi, and P. Hämäläinen, "A Reinforcement Learning Approach To Synthesizing Climbing Movements," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–7.

[37] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1251–1256.

[38] D. F. Hougen and S. N. H. Shah, "The Evolution of Reinforcement Learning *," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 1457–1464.

[39] W. Qiang and Z. Zhongli, "Reinforcement learning model, algorithms and its application," in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. IEEE, 2011, pp. 1143–1146.

[40] "5 Things You Need to Know about Reinforcement Learning," Internet: https://www.kdnuggets.com/2018/03/5-things-reinforcement- learning.html [Feb. 2, 2021].

[41] "Reinforcement learning Deep-Q Networks." Internet: https://blogs.oracle.com/datascience/reinforcement-learning-deep-q-networks [Feb. 2, 2021].

[42] "A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python," Internet: https://medium.com/analytics-vidhya/a-hands-on-introduction-to-deep-q-learning-using-openai-gym-in-python- b15d7d8597d [Feb. 2, 2021].

[43] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[44] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv preprint arXiv:1912.01703*, 2019.

[45] G. Brockman *et al*, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[46] "NSL-KDD dataset," Internet: https://www.unb.ca/cic/datasets/nsl.html [Nov. 2, 2020].

[47] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.

[48] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, "Short-Term Photovoltaic Power Forecasting Based on Long Short Term Memory Neural Network and Attention Mechanism," *IEEE Access*, vol. 7, pp. 78 063– 78 074, 2019.

[50] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, "Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation," *Water*, vol. 12, no. 1, p. 175, 2020.

[51] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[52] M. Vihinen, “How to evaluate performance of prediction methods? Measures and their interpretation in variation effect analysis,” in *BMC Genomics*, vol. 13, no. 4. BioMed Central, 2012, pp. 1–10.

[53] E. Suwannalai and C. Polprasert, “Network Intrusion Detection Systems Using Adversarial Reinforcement Learning with Deep Q-network,” in *2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE, 2020, pp. 1–7.

[54] Y.-F. Hsu and M. Matsuoka, ”A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System,” *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, Piscataway, NJ, USA, 2020, pp. 1-6.

[55] X. Ma and W. Shi, “AESMOTE: Adversarial Reinforcement Learning with SMOTE for Anomaly Detection,” *IEEE Transactions on Network Science and Engineering*, 2020.

[56] G. Caminero, M. Lopez-Martin, and B. Carro, “Adversarial environment reinforcement learning algorithm for intrusion detection,” *Computer Networks*, vol. 159, pp. 96–109, 2019.