**NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY**



**Faculty of Applied Science**

**Department of Computer Science**

Distributed Denial Of Service Attack Detection using Convolutional Neural Networks

**STUDENT NAME : TAFADZWA N DINGWA**

**STUDENT NUMBER : N02017099Y**

**SUPERVISOR : Dr S Nleya**

**This report is submitted in Partial Fulfilment of the requirements of the Bachelor of Science Honours Degree in Computer Science at the National University Of Science And Technology**

# **Declaration of Originality**

****

# **Acknowledgements**

# **Abstract**

In the realm of cybersecurity, combatting Distributed Denial-of-Service (DDoS) attacks remains a crucial challenge. These attacks aim to overwhelm a network or server with excessive traffic, rendering it inaccessible to legitimate users. This study proposes a novel system leveraging the power of neural networks to effectively detect DDoS attacks. The proposed system extracts relevant features from network traffic data, encompassing characteristics like packet size, flow rate, and source-destination information. This data serves as input for the meticulously designed neural network architecture. The architecture which identifies subtle anomalies indicative of DDoS attacks, is then trained using a comprehensive dataset encompassing both normal and attack traffic patterns. Following the training phase, the system is evaluated on its ability to classify incoming network traffic accurately. The evaluation assesses the system's performance metrics, including detection accuracy, loss, and F1 score. This work sheds light on the promising potential of neural networks in the fight against DDoS attacks. By harnessing the power of machine learning, the proposed system offers a robust and effective solution for safeguarding networks against these malicious attempts.

Table of Contents

[**Declaration of Originality** i](#_Toc164338338)

[**Acknowledgements** i](#_Toc164338339)

[**Abstract** ii](#_Toc164338340)

[**List of Figures** v](#_Toc164338341)

[**List of Tables** vi](#_Toc164338342)

[**List of Abbreviations** vi](#_Toc164338343)

[**CHAPTER 1 INTRODUCTION** 1](#_Toc164338344)

[1.1 Introduction 1](#_Toc164338345)

[1.2 Problem Description 1](#_Toc164338346)

[1.3 Background and Motivation 1](#_Toc164338347)

[1.4 Aim 2](#_Toc164338348)

[1.5 Objectives 2](#_Toc164338349)

[1.6 Proposed Approach 2](#_Toc164338350)

[1.7 Highlights of what has been achieved 3](#_Toc164338351)

[1.8 Report organization 3](#_Toc164338352)

[**CHAPTER 2 LITERATURE REVIEW** 4](#_Toc164338353)

[2.1. Introduction 4](#_Toc164338354)

[2.2 DDOS Definition 5](#_Toc164338355)

[2.2.1 Volume Based Attacks 6](#_Toc164338356)

[2.2.2 Protocol Attacks 7](#_Toc164338357)

[2.2.3 Application Layer Attacks 7](#_Toc164338358)

[2.3 General DDOS Detection Approaches 9](#_Toc164338359)

[2.3.1 Statistical and information theory methods 9](#_Toc164338360)

[2.3.2 Detection of DDOS traffic based on soft computing methods 10](#_Toc164338361)

[2.3.3 Detection of DDOS traffic based on Knowledge methods 10](#_Toc164338362)

[2.3.4 Clustering based techniques 11](#_Toc164338363)

[2.4 Neural network architectures for DDoS detection 11](#_Toc164338364)

[2.4.1 Artificial Neural Networks (ANNs) 11](#_Toc164338365)

[2.4.2 Neural Network Architecture 12](#_Toc164338366)

[2.4.3 Main Components of Neural Network Architecture 13](#_Toc164338367)

[2.4.4 Types of Neural Network Architectures 13](#_Toc164338368)

[2.5 Related work 15](#_Toc164338369)

[2.5.1 Detecting DDoS attacks using adversarial neural network 15](#_Toc164338370)

[2.5.2 Early detection of DDoS attacks using photonic neural networks 15](#_Toc164338371)

[2.5.3 DDoS detection and mitigation using machine learning 15](#_Toc164338372)

[2.5.4 An approach to detect DDoS attacks using neural networks 16](#_Toc164338373)

[2.5.5 Refined LSTM based intrusion detection for denial of service attack in internet of things 16](#_Toc164338374)

[2.6 Conclusion 16](#_Toc164338375)

[**CHAPTER 3 METHODOLOGY** 17](#_Toc164338376)

[3.1. Introduction 17](#_Toc164338377)

[3.2 Research Methodology 17](#_Toc164338378)

[3.3 Software methodology 18](#_Toc164338379)

[3.4 Neural Network to be used 22](#_Toc164338380)

[3.4.1 Architecture of Convolutional Neural Networks 22](#_Toc164338381)

[3.5 Data Acquisition Techniques 23](#_Toc164338382)

[3.6 Data Preprocessing Techniques 24](#_Toc164338383)

[3.7 Tools/Technology 25](#_Toc164338384)

[3.8 Conclusion 25](#_Toc164338385)

[**CHAPTER 4 SYSTEM ANALYSIS AND DESIGN** 26](#_Toc164338386)

[4.1 Introduction 26](#_Toc164338387)

[4.2 Requirements specification 26](#_Toc164338388)

[4.2.1 Functional Requirements 27](#_Toc164338389)

[4.2.2. Non-Functional Requirements: 27](#_Toc164338390)

[4.3 Feasibility study 28](#_Toc164338391)

[4.3.1 Technical Feasibility 28](#_Toc164338392)

[4.3.2 Economic Feasibility 28](#_Toc164338393)

[4.3.3 Operational Feasibility 29](#_Toc164338394)

[4.4 System Flowchart 29](#_Toc164338395)

[4.5 Use Case Diagram 31](#_Toc164338396)

[4.6 User Interface Design 32](#_Toc164338397)

[4.7 Database design 35](#_Toc164338398)

[4.8 Conclusion 37](#_Toc164338399)

[**CHAPTER 5 DESIGN AND IMPLEMENTATION** 38](#_Toc164338400)

[5.1 Introduction 38](#_Toc164338401)

[5.2 Hardware and Environment Setting 38](#_Toc164338402)

[5.3 Development Tools 39](#_Toc164338403)

[5.3.1 Integrated Development Environment (IDE) 39](#_Toc164338404)

[5.3.2 Deployment Tool 39](#_Toc164338405)

[5.3.4 API Development Tool 40](#_Toc164338406)

[5.3.5 Database Management Tool (DBMT) 40](#_Toc164338407)

[5.4 System Implementation 41](#_Toc164338408)

[5.4.1 Data Collection 41](#_Toc164338409)

[5.4.2 Data Preparation 42](#_Toc164338410)

[5.4.3 Feature Selection 43](#_Toc164338411)

[5.4.4 Convolutional Neural Network 44](#_Toc164338412)

[5.4.5 Compilation and Training 46](#_Toc164338413)

[5.4.6 Database 48](#_Toc164338414)

[5.4.7 User Interface 50](#_Toc164338415)

[5.5 Test Plan 50](#_Toc164338416)

[5.5.1 Testing Scope 50](#_Toc164338417)

[5.5.2 Test Cases 50](#_Toc164338418)

[5.6 Testing 51](#_Toc164338419)

[5.7 Conclusion 51](#_Toc164338420)

[**CHAPTER 6 RESULTS EVALUATION AND CONCLUSION** 52](#_Toc164338421)

[6.1 Introduction 52](#_Toc164338422)

[6.2 Evaluation Metrics 52](#_Toc164338423)

[6.3 Summary of findings 55](#_Toc164338424)

[6.4 Review of Project Aim and Objectives 55](#_Toc164338425)

[6.5 Problems Encountered 56](#_Toc164338426)

[6.5 Future Work 56](#_Toc164338427)

[6.7 Conclusion 56](#_Toc164338428)

[References 57](#_Toc164338429)

[Appendices 59](#_Toc164338430)

# **List of Figures**

[Figure 1 Major categories of DDoS attacks. 6](#_Toc164338431)

[Figure 2 Conventional structure of a volumetric-based DDoS attack 7](#_Toc164338432)

[Figure 3 Taxonomy of DDoS attack detection mechanisms. 9](#_Toc164338433)

[Figure 4 Umbrella of data science techniques 12](#_Toc164338434)

[Figure 5 Neuron in Artificial Neural Network 13](#_Toc164338435)

[Figure 6 Agile Methodology 19](#_Toc164338436)

[Figure 7 DevOps Methodology 19](#_Toc164338437)

[Figure 8 Waterfall Methodology 20](#_Toc164338438)

[Figure 9 RAD Methodology 21](#_Toc164338439)

[Figure 10 Scrum Methodology Lifecycle 21](#_Toc164338440)

[Figure 11 Project Timeline 29](#_Toc164338441)

[Figure 12 System Flowchart 29](file:///C:\Users\user\Desktop\CHAPTER%20ONE.docx#_Toc164338442)

[Figure 13 Use Case Diagram 32](#_Toc164338443)

[Figure 14 Dashboard UI 33](#_Toc164338444)

[Figure 15 Alerts UI 34](#_Toc164338445)

[Figure 16 Reports UI 35](#_Toc164338446)

[Figure 17 ERD 37](#_Toc164338447)

[Figure 19 Device Specifications 38](#_Toc164338448)

[Figure 20 Convolutional neural network Code 45](#_Toc164338449)

[Figure 21 Test Set Evaluation Matrix 47](#_Toc164338450)

[Figure 22 Docker Deployment 48](#_Toc164338451)

[Figure 23 MySql Code 49](#_Toc164338452)

[Figure 24 Confusion Matrix 53](#_Toc164338453)

[Figure 25 Classification Report 54](#_Toc164338454)

# **List of Tables**

[Table 1 Overview of the three major DDoS attack classification. 8](#_Toc164338455)

[Table 3 List of Features 41](#_Toc164338456)

[Table 4 Cicddos2019 dataset general statistics 42](#_Toc164338457)

[Table 5 List of the 70 Features in the dataset 43](#_Toc164338458)

[Table 6 Importance Scores of Features 44](#_Toc164338459)

[Table 7 Features of Dataset Used 44](#_Toc164338460)

[Table 8 Test Case Results 51](#_Toc164338461)

# **List of Abbreviations**

DDoS – Distributed Denial Of Service

IOT – Internet of Things

AI – Artificial Intelligence

OSI – Open Systems Interconnection

TCP/IP – Transmission Control Protocol/Internet Protocol

POD – Ping of Death

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

# **CHAPTER 1 INTRODUCTION**

## 1.1 Introduction

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic (cloudflare, 2023). DDoS attacks are carried out with networks of Internet-connected machines. These networks consist of computers and other devices (such as IoT devices) which have been infected with [malware](https://www.cloudflare.com/learning/ddos/glossary/malware/), allowing them to be controlled remotely by an attacker. These individual devices are referred to as [bots](https://www.cloudflare.com/learning/bots/what-is-a-bot/) (or zombies), and a group of bots is called a [botnet](https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/). Once a botnet has been established, the attacker is able to direct an attack by sending remote instructions to each bot. When a victim’s server or network is targeted by the botnet, each bot sends requests to the target’s [IP address](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/), potentially causing the server or network to become overwhelmed, resulting in a [denial-of-service](https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/) to normal traffic. Because each bot is a legitimate Internet device, separating the attack traffic from normal traffic can be difficult

DDoS detection is the process of distinguishing distributed denial of service attacks from normal network traffic in order to perform effective attack mitigation (kentik, 2023). DDoS attacks intentionally occupy resources such as computing power and bandwidth to deny the services to potential users so the automatic identification of DDoS attacks is very important. This project aims to contribute to the field of network security by developing an advanced DDoS detection model using neural networks. The outcome of this project has the potential to enhance the security posture of organizations and improve their ability to defend against DDoS attacks.

## 1.2 Problem Description

Many DDoS attacks happen every day and many remain unnoticed. The real challenge in detecting and defending the DDoS attack is its dynamic nature. The source of the attack is not just a single node or a system on the Internet, there can be many systems participating and often these systems are distributed over different regions of the Internet. The original attack source is changed in a spoofed data packet, which makes it harder to know the actual IP address of the system from where the attack has originated. In addition, oftentimes the source system itself is not aware that it is compromised, and it’s being used as a bot by an attacker to launch a DDoS attack. The destination may know that an attack is happening but to stop it happening it will have to block all the incoming traffics and analyzing large flow information. These attacks pose a significant threat to network security, causing downtime and financial losses for organizations. Detecting and mitigating DDoS attacks is crucial for ensuring the availability and reliability of network services.

## 1.3 Background and Motivation

## 1.4 Aim

To design and implement a neural network-based DDoS detection model capable of identifying DDoS attacks in real-time.

## 1.5 Objectives

* To distinguish distributed denial of service (DDoS) attacks from normal network traffic in real time
* To alert the network administrator regarding any attack
* To produce reports on detected attacks in a network

## 1.6 Proposed Approach

The research will employ the quantitative research methodology. Quantitative research aims to create a general understanding of behavior of the system. They are often fast, focused, scientific and relatable.

The software methodology to be used is Scrum software development. We chose scrum because it ensure that project deliverables are completed quickly and efficiently. It also divided the project into easily manageable sprints.

The researcher will collect and work with datasets from reputable online data repositories. Intrusion detection evaluation dataset CICDDoS2019 dataset Sharafaldin et al. [2019] has proved to be a dependable.

The steps below depict the stages to be followed to develop the system.

**Data Collection and Preparation:**

- To collect network traffic data from various sources, including simulated attack scenarios and real-world network traffic.

- Preprocessing and cleaning the data to ensure it is suitable for training and testing the neural network.

**Feature Selection:**

- Identifying relevant features for DDoS attack detection, such as packet rates, traffic patterns, and source-destination information.

- Selection and extraction of these features from the network traffic data.

**Neural Network Design and Training:**

- Designing neural network architecture suitable for DDoS detection, considering factors like network size, layers, and activation functions.

- Training the neural network using labeled datasets that include both normal and DDoS attack traffic.

**Evaluation and Testing:**

- Assessing the performance of the neural network-based DDoS detection model using test datasets and evaluation metrics (e.g., accuracy, false positives, false negatives).

**System Design:**

-Designing the system that detects DDoS attacks in real time.

## 1.7 Highlights of what has been achieved

## 1.8 Report organization

This report details the design and evaluation of a system that utilizes neural networks for DDoS attack detection. The report will be organized as follows:

**1. Introduction**

* Briefly introduce the concept of DDoS attacks and their impact.
* Introduce the project problem statement.
* Identify the project aim and objectives

**2. Literature Review**

* Summarize existing research on DDoS detection using various techniques, including machine learning and neural networks.
* Highlight the strengths and limitations of existing approaches.

**3. Methodology**

* Describe the research and software methodology used by the researcher.

**4. Systems Analysis and Design**

* .Description in detail of how the system is developed.

**5. Design and Implementation**

* Coding and testing
* Define the metrics used for evaluation, such as accuracy, precision, recall, and false positive rate.
* Present the evaluation results, including the system's performance compared to existing methods.

**6. Conclusion**

* Summarize the key findings of the study, emphasizing the effectiveness of the proposed system in detecting DDoS attacks.
* Discuss the limitations of the study and potential avenues for future research.

**7. References**

* List of all references cited throughout the report in Harvard style.

**8. Appendix**

* Include any supplementary information, such as detailed network architecture diagrams, additional evaluation data, and source code for the system.

# 

# **CHAPTER 2 LITERATURE REVIEW**

## 2.1. Introduction

This chapter covers past research that has been made by other researchers into detecting DDoS attacks using neural network. The researcher will also cover literature on the classification of DDoS attacks and general approaches to detect them. The aim of this chapter is to provide readers with an understanding of the different approaches that have been developed in the recent years to address the problem of DDoS detection.

By carefully reviewing and analyzing existing research, this chapter aims to:

* **Demonstrate the project's novelty and significance:** Position the proposed neural network-based approach within the broader context of DDoS detection research, showcasing its potential contribution to the field.
* **Draw upon established knowledge:** Leverage the insights and findings from previous studies to inform the project's methodology and design choices.
* **Identify research opportunities:** Highlight potential areas for further exploration and refinement based on the identified knowledge gaps and limitations of existing approaches.

## 2.2 DDOS Definition

DDoS stands for Distributed Denial of Service. DDoS attack is a malicious attempt to prevent legitimate users from accessing online services or resources by incapacitating the server that provides the service or bringing down the network connectivity between users and the server. The attacks accomplish this by flooding the target with traffic or sending specifically crafted packets that trigger a crash. In both instances, the DoS/DDoS attacks deprive legitimate users (i.e., employees, subscribers, or account holders) of the online services or network resources.

DoS/DDoS attackers often target web servers of high profile organizations such as banking, commerce, media companies, or government and trade organizations. Though DOS attacks do not typically result in the theft or significant loss of information or other assets, they can cost the victim a great deal of time and money to handle, its reputation, and lost opportunities.

The difference between DoS and DDoS attacks is in the number of attack sources involved. A single attacker triggers a DoS attack. In contrast, multiple attackers trigger a DDoS attack; and by taking advantage of distributed attacks via botnets, a DDoS attack is several times more destructive than DoS. (A. Ghaben, 2021)

DDoS attacks differ greatly in terms of how they are initiated and the impact they have on the target server. As shown in **Figure 1**, DDoS attacks can therefore be divided into three categories: volumetric base, protocol base, and application base. (ddos-attacks, 2023)

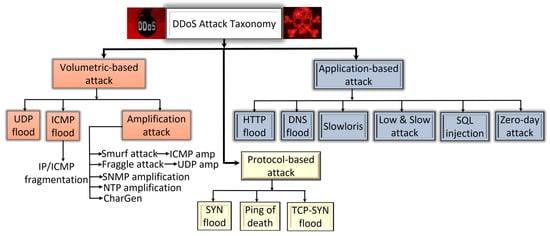


Figure 1 Major categories of DDoS attacks.

### ****2.2.1 Volume Based Attacks****

Volumetric attacks can also be called “floods” because an attack floods a target’s server with requests, like unwanted pings. Attacks are measured in bits per second (bps) or Gigabits per second (Gbps).

The concept of a volumetric attack is to send as much traffic as possible to a site to overwhelm the server’s bandwidth. Volumetric attacks are typically produced using amplification techniques. DNS amplification is one of the more common techniques attackers use to carry out a volumetric attack. The bad actor sends small DNS requests with the victim’s spoofed source IP address to a DNS server. When the server receives the request, it responds to the victim with a large response. Attackers also create volumetric attacks using botnets made up of exploited IoT devices. Connected devices usually lack basic security defences, but because they’re connected to the Internet and can execute code, they can be easily exploited. The first volumetric DDoS attack made headlines in the late 1990s and has since spawned an army of copycats. The generic structure of a conventional volumetric-based DDoS attack is illustrated below.

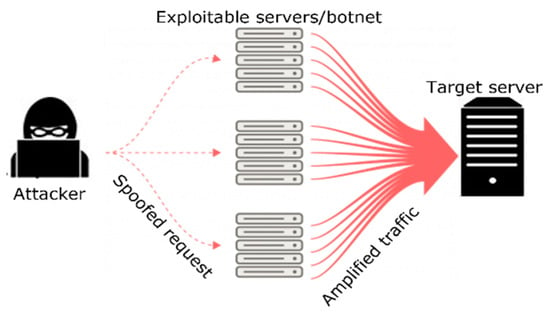


Figure 2 Conventional structure of a volumetric-based DDoS attack

### ****2.2.2 Protocol Attacks****

This type of attack consumes actual server resources, or those of intermediate communication equipment, such as firewalls and [load balancers](https://www.imperva.com/learn/availability/load-balancing-algorithms/), and is measured in packets per second (Pps). An internet protocol is a discrete set of rules for exchanging information across the internet. TCP/IP is one of the most well-known rules for exchanging requests and data. A bad actor can severely disrupt an online service by exploiting these rules. Protocol attacks often work at layers 3 and 4 of the OSI model on network devices like routers. Because they are on the network layer, they are measured in packets per second (pps).

An example of a Network Protocol DDoS Attack is the ping of death (POD**)**. The ping of death (POD) is an IP fragmentation attack that exploits the inherent size limitation of a packet. By manipulating parts of the packet or fragments, the exploit can overflow the memory buffers allocated to that packet and then deny service to legitimate packets.

### ****2.2.3 Application Layer Attacks****

These threats are harder to detect because attackers usually make requests like legitimate users. Consequently, these attacks often show up as smaller traffic spikes and do not require the assistance of a botnet. Application layer attacks are measured in requests per second (RPS) — the number of requests an application makes. It is considered a resource-based attack; therefore, it takes fewer requests to bring down an application because the attack is focused on overwhelming the CPU and memory.

An attack typically includes hitting the web server, running PHP scripts, and contacting the database to load web pages. A single HTTP request, which is simple to execute on the client side, can cause a server to execute many internal requests and load numerous files to fulfil the request, which slows the system.

The table below shows the comparison of the 3 major DDoS attack.

| **Attack Type** | **Features** | **Attack Magnitude** | **Effect on Target Server** | **Attack Complexity** | **Affected Layer** | **Frequency of Occurrence** |
| --- | --- | --- | --- | --- | --- | --- |
| Volumetric-based | The use of a huge amount of traffic to saturate the bandwidth of the target server | Bits per second (bps), Gbps, flood | Access to the target resources may be totally blocked by the attack’s sheer volume of traffic. | Easy to generate using simple amplification techniques | Network layer | Most common |
| Protocol-based | It exploits the weakness in layers 3 and 4 of the protocol stack to make the target server not accessible. | Packets per second (pps) | It disrupts service by consuming all the target server’s processing power or resources, including the firewall. | Less complex | Network and transport | More common |
| Application-based | It harnesses the flaws in layer 7 of the protocol stack to make the target server not accessible. | Requests per second (rps) | It creates a session with the target and then uses up its resources by completely dominating processes. | Complex and difficult to detect | Application | Less common |

Table 1 Overview of the three major DDoS attack classification.

## 2.3 General DDOS Detection Approaches

DDoS detection is the process of distinguishing distributed denial of service attacks from normal network traffic in order to perform effective attack mitigation. The primary goal of a DDoS attack is to either limit access to an application or network service, thereby denying legitimate users access to the services. (kentik, 2023)

Detecting an ongoing DDoS attack is challenging for several reasons. It is challenging to distinguish DDoS attacks from flash crowd events due to their network traffic similarity. Therefore, there is a need for an effective and accurate mechanism to detect DDoS attacks in the network.

.

There are many mechanisms have been proposed to detect DDoS attacks, such as machine learning and the mathematical based mechanisms. The difference between mathematical-based DDoS detection mechanism and machine learning-based DDoS detection mechanism is that machine learning-based mechanism aims to predict future events or classify an existing material based on training data, i.e., to classify the network traffic as normal traffic or DDoS attacks. On the other hand, mathematical-based DDoS detection mechanism aims to find the relationship between the data points, i.e., find the relationship between two or more features that represent the network traffic. (A. Ghaben, 2021) categorized DDoS attack defense mechanisms into four categories: statistical technique, soft computational technique, knowledge-based methods, and clustering, as shown below.

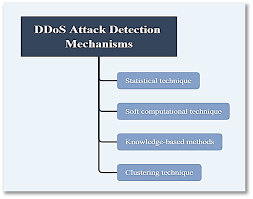


Figure 3 Taxonomy of DDoS attack detection mechanisms.

### 2.3.1 Statistical and information theory methods

Statistical traffic characteristics can be utilized to differentiate between normal and DDoS traffic. Statistical-based approaches are based on the use of statistical methods in determining the normal traffic model. After that, it can be statistically determined whether a new traffic instance (flow, packet or package set) corresponds to a defined model.

The commonly used DDoS traffic detection methods from statistics and information theory domain are deviation, cumulative sum, correlation, entropy, and covariance. Self-similarity and long-range dependence (LRD) of network traffic are often used in statistical processing and DDoS traffic detection. Data traffic under normal conditions maintains an LRD property which implies loss or reduction of LRD property in the event of anomalies in the communications network such as the occurrence of DDOS traffic. Therefore, by analyzing LRD property of the incoming traffic it is possible to detect DDoS traffic. Self-similarity and LRD are expressed by the Hurst parameter (H), also called the long-term dependence index, and it is measured by statistical estimators such as autocorrelation, variance aggregation, wavelet, R/S method and similar.

There are challenges in determining LRD property to determine the time period within traffic. If the time period is too short, the results of the analysis will not be valid due to the insufficient volume of traffic to determine the degree of LRD, while a too long time period will cause the inability to detect short-term anomalies. In addition, the disadvantage of this detection is predefined static limit value of the Hurst parameter which results in the detection of DDoS traffic only when its intensity causes a change in the value of the Hurst parameter above a defined threshold (A. Ghaben, 2021).

### 2.3.2 Detection of DDOS traffic based on soft computing methods

The advantages of soft computing methods compared to the previously described are tolerance on imprecision, uncertainty, incompleteness and partial authenticity of the input data. The robustness and efficiency of these methods have been proven in solving many complex problems like pattern matching. Soft computing approach is effective in solving problems where information about the problem is incomplete, and the possible problem solution is not exact. This is the reason for the frequent use of this group of methods in DDoS traffic detection, where artificial neural networks (ANN) and fuzzy logic are often used (A. Ghaben, 2021).

### 2.3.3 Detection of DDOS traffic based on Knowledge methods

The third category comprises DDoS attack mechanisms utilizing knowledge-based techniques. Knowledge-based techniques attempt to match the traffic or flow patterns against a set of predefined rules. The traffic or flow is flagged as an attack if it fit the rules. Otherwise, the pattern is considered normal. An example is the use of machine-learning methods is one of the approaches to DDoS traffic detection. The reason for their use is the advantage over the pattern-based detection method because the human factor's impact is significantly reduced in the overall DDoS traffic detection process. The machine-learning methods can be classified on supervised (existing knowledge is used to classify the future unknown instances) and unsupervised (attempts to determine the corresponding instance class without prior knowledge). Examples of supervised machine-learning methods commonly used in DDoS traffic detection are decision trees, k-nearest neighbor (kNN), vector machines (SVM) and naïve Bayes classifier. Unsupervised machine-learning methods commonly used in DDOS traffic detection are fuzzy C means and k-mean clustering (A. Ghaben, 2021).

### 2.3.4 Clustering based techniques

The fourth category of DDoS attack mechanisms is clustering-based techniques, a data mining technique known as unsupervised classification. It does not need training with a training dataset, and the strength of clustering is within the algorithm itself.

.

These approaches attempt to detect both either single point outliers or cluster-based outliers, or can assign each outlier a degree of being an outlier. LDBSCAN (local-density-based spatial clustering of applications with noise) is a cluster-based outlier detection algorithm. LDBSCAN randomly selects one core point which has not been clustered, and then retrieves all points that are local density reachable from the chosen core point to form a cluster. It does not stop until there is no unclustered core point. (A. Ghaben, 2021).

## 2.4 Neural network architectures for DDoS detection

### 2.4.1 Artificial Neural Networks (ANNs)

(Choi RY, 2020) defines ANNs as “a machine learning algorithm inspired by biological neural networks”. Each ANN consists of nodes that can be compared to cell bodies, which communicate with each other through connections, similar to axons and dendrites in biological neurons. Just as synapses between neurons are reinforced when their neurons produce correlated outputs in a biological neural network, according to the Hebbian theory, the connections between nodes in an ANN are weighted based on their ability to produce a desired outcome.

An external file that holds a picture, illustration, etc.
Object name is tvst-9-2-14-f001.jpg

Figure 4 Umbrella of data science techniques

Artificial intelligence (AI) falls within the realm of data science, and includes classical programming and machine learning (ML). ML contains many models and methods, including deep learning (DL) and artificial neural networks (ANN).

### 2.4.2 Neural Network Architecture

The architecture of neural networks is made up of an input, output, and hidden layer. Neural networks themselves, or artificial neural networks (ANNs), are a subset of machine learning designed to mimic the processing power of a human brain. Neural networks function by passing data through the layers of an artificial neuron.

The Neural Network architecture is made of individual units called neurons that mimic the biological behaviour of the brain. Below are the various components of a neuron. (Baheti, 2021)

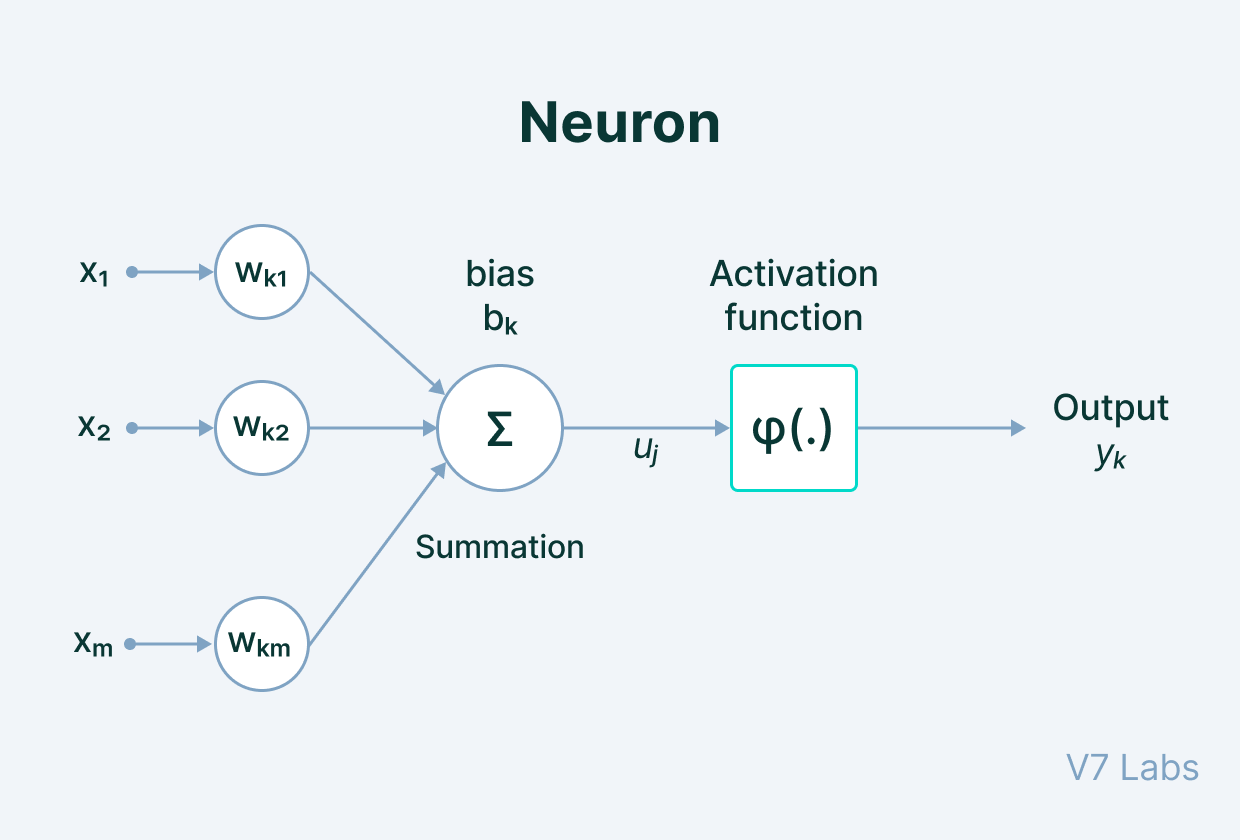


Figure 5 Neuron in Artificial Neural Network

### 2.4.3 Main Components of Neural Network Architecture

There are many components to neural network architecture. Each neural network has a few components in common:

Input - Input is data that is put into the model for learning and training purposes.

Weight - Weight helps organize the variables by importance and impact of contribution.

Transfer function - Transfer function is when all the inputs are summarized and combined into one output variable.

[Activation function](https://h2o.ai/wiki/activation-function) - The role of the activation function is to decide whether or not a specific neuron should be activated. This decision is based on whether or not the neuron’s input will be important to the prediction process.

Bias - Bias shifts the value given by the activation function. (Baheti, 2021)

### 2.4.4 Types of Neural Network Architectures

Neural networks are an efficient way to solve machine learning problems and can be used in various situations. Neural networks offer precision and accuracy.

#### 2.4.4.1 Standard neural networks

**Perceptron**

Perceptron is the simplest Neural Network architecture. It is a type of Neural Network that takes a number of inputs, applies certain mathematical operations on these inputs, and produces an output. It takes a vector of real values inputs, performs a linear combination of each attribute with the corresponding weight assigned to each of them. The weighted input is summed into a single value and passed through an activation function. These perceptron units are combined to form a bigger[Artificial Neural Network](https://www.v7labs.com/definitions/artificial-neural-network) architecture. (Baheti, 2021)

**Feed-Forward Networks**

A multi-layered neural network where the information moves from left to right, or, in a forward direction. The input values pass through a series of hidden layers on their way to the output layer.  In the forward pass, the information comes inside the model through the input layer, passes through the series of hidden layers, and finally goes to the output layer. This Neural Networks architecture is forward in nature—the information does not loop with two hidden layers. The later layers give no feedback to the previous layers. The basic learning process of Feed-Forward Networks remains the same as the perceptron. (Baheti, 2021)

**Residual Networks (ResNet)**

A deep feed-forward network with hundreds of layers. The core idea behind ResNet is that a deeper network can be made from a shallow network by copying weight from the shallow counterparts using identity mapping. The data from previous layers is fast-forwarded and copied much forward in the Neural Networks. This is what we callskip connections first introduced in Residual Networks to resolve vanishing gradients. (Baheti, 2021)

#### 2.4.4.2 Recurrent neural networks

Recurrent neural networks (RNNs) remember previously learned predictions to help make future predictions with accuracy.

* Long short term memory network (LSTM) - LSTM adds extra structures, or gates, to an RNN to improve memory capabilities.
* Echo state network (ESN) - A type of RNN hidden layers that are sparsely connected. (Baheti, 2021)

#### 2.4.4.3 Convolutional neural networks

Convolutional neural networks (CNNs) are a type of feed-forward network that are used for image analysis and language processing. There are hidden convolutional layers that form ConvNets and detect patterns. CNNs use features such as edges, shapes, and textures to detect patterns. Examples of CNNs include:

* AlexNet - Contains multiple convolutional layers designed for image recognition.
* Visual geometry group (VGG) - VGG is similar to AlexNet, but has more layers of narrow convolutions.
* Capsule networks - Contain nested capsules (groups of neurons) to create a more powerful CNN. (Baheti, 2021)

#### 2.4.4.4 Generative adversarial networks

Generative adversarial networks (GAN) are a type of [unsupervised learning](https://h2o.ai/wiki/unsupervised-machine-learning) where data is generated from patterns that were discovered from the input data. GANs have two main parts that compete against one another:

* Generator - creates synthetic data from the learning phase of the model. It will take random datasets and generate a transformed image.
* Discriminator - decides whether or not the images produced are fake or genuine.

GANs are used to help predict what the next frame in a video might be, text to image generation, or image to image translation. (Baheti, 2021)

#### 2.4.4.5 Transformer neural networks

Unlike RNNs, transformer neural networks do not have a concept of timestamps. This enables them to pass through multiple inputs at once, making them a more efficient way to process data. (WIKI, 2023)

## 2.5 Related work

The following are some past studies that have been done which I found through Google Scholar platform. This was done through keyword search for entries relating to “DDoS detecting using Neural Networks”. Descriptions are summarized from the abstracts and results of the research that were done.

### **2.5.1 Detecting DDoS attacks using adversarial neural network**

(Ali Mustapha, 2023) proposed a [DDoS](https://www.sciencedirect.com/topics/computer-science/distributed-denial-of-service) detection method based on the Long Short-Term Memory (LSTM) model, which is a type of [Recurrent Neural Networks](https://www.sciencedirect.com/topics/computer-science/recurrent-neural-network) (RNNs) capable of learning long-term dependencies. The detection scheme yields a high accuracy level in detecting attacks. Secondly, they tested the same technique against different types of adversarial DDoS attacks generated using GAN. The results show the inefficiency of the LSTM-based detection scheme. Finally, they demonstrate how to enhance the scheme to detect adversarial attacks. The experimental results show that the detection model is efficient and accurate in identifying GAN-generated adversarial DDoS traffic with a detection ratio ranging between 91.75% and 100%. Despite the use LSTM model, they didn’t investigate the vulnerabilities of ML-based IDS against adversarial attacks. It would have been necessary to evaluate the performance of our IDS on data generated by another model such as the auto-encoder.

### 2.5.2 Early detection of DDoS attacks using photonic neural networks

(Kirtas, 2022) proposed employing a photonic neuromorphic lookaside accelerator, aiming to perform real-time traffic inspection, enabling it to detect port-scanning attacks, which are indicative of DDoS attacks. They designed, trained, and evaluated a Photonic Neural Network (PNN) capable of detecting DDoS attacks and operating on such photonic neuromorphic lookaside accelerators. The experimental evaluation was performed on Transport Control Protocol (TCP) traces obtained by simulating a port scanning attack.

### 2.5.3 DDoS detection and mitigation using machine learning

(Gawande, 2018) discussed a way to detect DDoS attacks using machine learning tools at the routers, instead of setting up a centralized analysis system. They proposed a standard communication architecture which can be used across all the networking devices for mitigating DDoS attacks. They combined two machine learning algorithms (clustering and classification) to efficiently detect DDoS attacks at the router.

### 2.5.4 An approach to detect DDoS attacks using neural networks

(Rangapur, 2022) presented the detection of DDoS attacks using neural networks, that would flag malicious and legitimate data flow, preventing network performance degradation. The model achieved better results using a smaller value of dropout with a ReLU activation function for every layer. It was observed that the model achieved overall accuracy up to 99.7%, which was trained for 40 epochs.

### 2.5.5 Refined LSTM based intrusion detection for denial of service attack in internet of things

In Adefemi Alimi (Adefemi Alimi KO, 2022) , the authors proposed an IDS based on LSTM for detecting DOS attacks. They evaluated the proposed framework using the CICIDS-2017, and NSL-KDS Tavallaee datasets. The results obtained show that the LSTM can effectively detect DOS attacks with an accuracy of 99.2% with CICIDS-2017 and 98.6% with NSL-KDD. Their framework was tested and evaluated only on DOS attacks. The limitation here is that the model lacks generalization because of the use of some features with uneven distribution.

## 2.6 Conclusion

This chapter concluded the literature review by comprehensively examining existing research on DDoS detection methods and the application of neural networks in cybersecurity. It achieved the following key objectives:

* **Contextualized the problem:** Established the criticality of DDoS detection and the challenges associated with it.
* **Surveyed existing methods:** Analyzed various traditional and potentially applicable techniques for DDoS detection, highlighting their advantages and limitations.
* **Explored neural network applications:** Delved into the use of neural networks for cybersecurity, specifically focusing on their potential for DDoS detection and drawing insights from prior research.
* **Identified knowledge gaps:** Pinpointed areas where existing research lacks depth or innovation, paving the way for this project's potential contribution.

# **CHAPTER 3 METHODOLOGY**

## 3.1. Introduction

The preceding chapters established a comprehensive understanding of DDos attacks, neural networks and analysis of existing studies. This chapter focuses on the crucial aspect of **methodology**, delving into the strategies and processes employed to achieve the project's goals.

Here, we will shed light on the following key elements of the project's methodology:

* **Research Methodology**
* **Software Methodology**
* **Data Acquisition and Preprocessing**

By meticulously examining these aspects, this chapter aims to provide a transparent and informative overview of the **research and software methodology approach** adopted in the project. It will showcase the rationale behind the chosen techniques and the effectiveness of the implemented strategies in achieving the desired outcomes.

## 3.2 Research Methodology

A research methodology is defined by (Witwatersrand, 2023) as the specific procedures or techniques used to identify, select, process, and analyse information about a topic. It describes the techniques and procedures used to identify and analyze information regarding a specific research topic. It is a process by which researchers design their study so that they can achieve their objectives using the selected research instruments. It includes all the important aspects of research, including research design, data collection methods, data analysis methods, and the overall framework within which the research is conducted. (tiffin.edu, 2022).

Described below are the three major methodologies employed by researchers when conducting their studies (MEHTA, 2023):

**Quantitative research methodology**

Focuses on measuring and testing numerical data. This approach is good for reaching a large number of people in a short amount of time. This type of research helps in testing the causal relationships between variables, making predictions, and generalizing results to wider populations.

**Qualitative research methodology**

Examines the opinions, behaviours and experiences of people. It collects and analyzes words and textual data. This research methodology requires fewer participants but is still more time consuming because the time spent per participant is quite large. This method is used in exploratory research where the research problem being investigated is not clearly defined.

**Mixed-method research methodology**

Uses the characteristics of both quantitative and qualitative research methodologies in the same study. This method allows researchers to validate their findings, verify if the results observed using both methods are complementary, and explain any unexpected results obtained from one method by using the other method.

This research employs the quantitative research methodology as we will use a trained neural network architecture to understand the behaviour of the network and predict whether there is DDoS attack or not.

## 3.3 Software methodology

Software development methodology is a framework for structuring, planning, and controlling the process of developing an information system. The primary goal of these approaches is to ensure smooth software development in accordance with project specifications. (rikkeisoft, 2022). Following a methodology benefits developers because these processes lay out a structured sequence of steps that guide professionals through each stage of development. Additionally, development methodologies often follow a design philosophy, which can help developers align their process and the product's features with its functional goals. (Team I. E., 2023).

Described below are the three major methodologies employed by researchers when conducting their studies (Team, 2017)**.**

**Agile Methodology**

Teams use the agile development methodology to minimize risk (such as bugs, cost overruns, and changing requirements) when adding new functionality. In all agile methods, teams develop the software in iterations that contain mini-increments of the new functionality. There are many different forms of the agile development method, including scrum, crystal, extreme programming (XP), and feature-driven development (FDD).

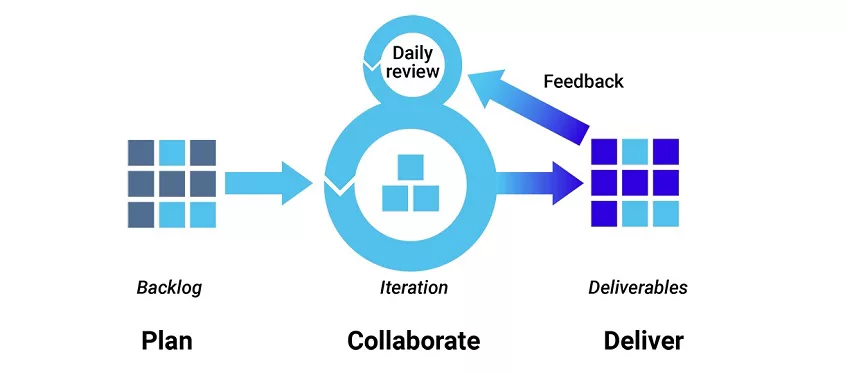


Figure 6 Agile Methodology

The primary benefit of agile software development is that it allows software to be released in iterations. Iterative releases improve efficiency by allowing teams to find and fix defects and align expectation early on. They also allow users to realize software benefits earlier, with frequent incremental improvements.

Agile development methods rely on real-time communication, so new users often lack the documentation they need to get up to speed. They require a huge time commitment from users and are labour intensive because developers must fully complete each feature within each iteration for user approval.

**DevOps Methodology**

[DevOps](https://www.synopsys.com/glossary/what-is-devops.html) is not just a development methodology but also a set of practices that supports an organizational culture. DevOps deployment centres on organizational change that enhances collaboration between the departments responsible for different segments of the development life cycle, such as development, quality assurance, and operations.

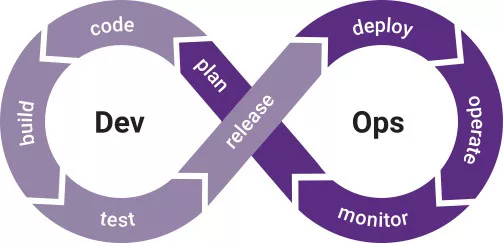


Figure 7 DevOps Methodology

DevOps is focused on improving time to market, lowering the failure rate of new releases, shortening the lead time between fixes, and minimizing disruption while maximizing reliability. To achieve this, DevOps organizations aim to automate [continuous deployment](https://www.synopsys.com/glossary/what-is-continuous-deployment.html) to ensure everything happens smoothly and reliably. Companies that use DevOps methods benefit by significantly reducing time to market and improving customer satisfaction, product quality, and employee productivity and efficiency.

**Waterfall Methodology**

Many consider the waterfall method to be the most traditional software development method. The waterfall method is a rigid linear model that consists of sequential phases (requirements, design, implementation, verification, maintenance) focusing on distinct goals. Each phase must be 100% complete before the next phase can start. There’s usually no process for going back to modify the project or direction.

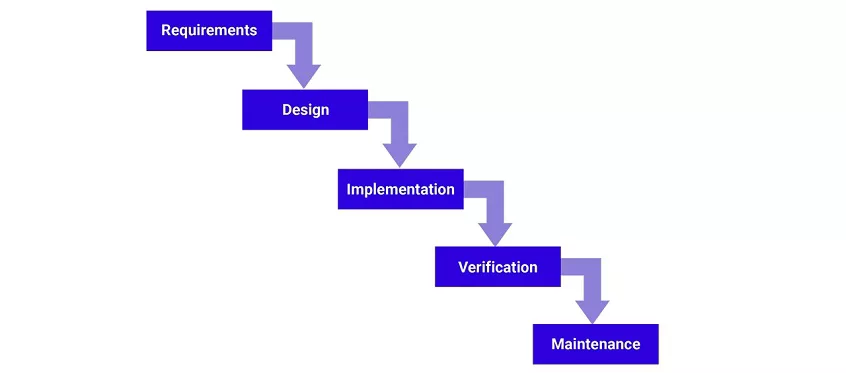


Figure 8 Waterfall Methodology

The linear nature of the waterfall development method makes it easy to understand and manage. Projects with clear objectives and stable requirements can best use the waterfall method. Less experienced project managers and project teams, as well as teams whose composition changes frequently may benefit the most from using the waterfall development methodology.

The waterfall development method is often slow and costly due to its rigid structure and tight controls. These drawbacks can lead waterfall method users to explore other software development methodologies.

**Rapid application development**

Rapid application development (RAD) is a condensed development process that produces a high-quality system with low investment costs. Scott Stiner, CEO and president of UM Technologies, [said in Forbes](http://www.forbes.com/sites/forbestechcouncil/2016/08/24/rapid-application-development-rad-a-smart-quick-and-valuable-process-for-software-developers/), “This RAD process allows our developers to quickly adjust to shifting requirements in a fast-paced and constantly changing market.” The ability to quickly adjust is what allows such a low investment cost.

The rapid application development method contains four phases: requirements planning, user design, construction, and cutover. The user design and construction phases repeat until the user confirms that the product meets all requirements.

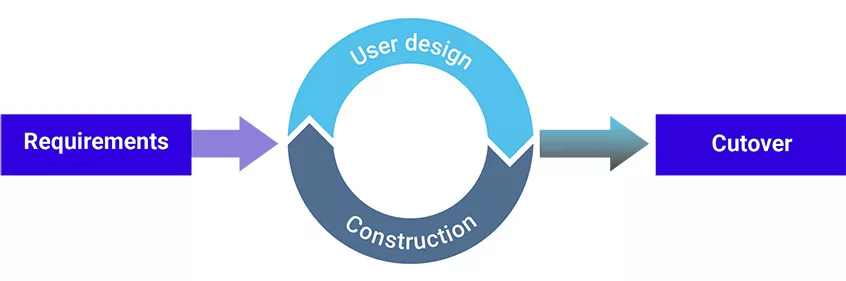


Figure 9 RAD Methodology

 Rapid application development is most effective for projects with a well-defined business objective and a clearly defined user group, but which are not computationally complex. RAD is especially useful for small to medium projects that are time sensitive.

Rapid application development requires a stable team composition with highly skilled developers and users who are deeply knowledgeable about the application area. Deep knowledge is essential in a condensed development timeline that requires approval after each construction phase. Organizations that don’t meet these requirements are unlikely to benefit from RAD.

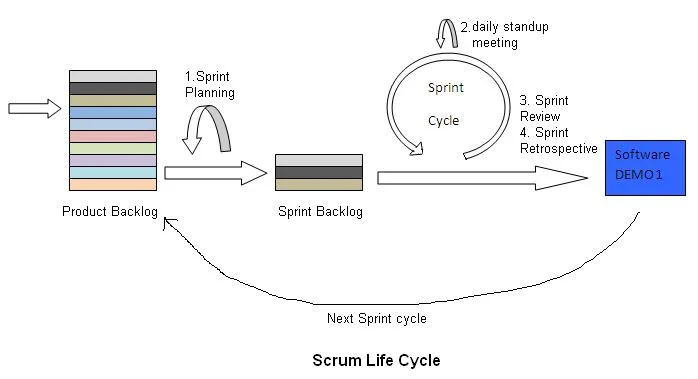
This research employs the Scrum methodology. Scrum is an agile framework that will help the researcher to work collaboratively with the team to build a high-quality system. It is a lightweight, iterative, and incremental methodology.

Figure 10 Scrum Methodology Lifecycle

**Product Backlog**

The Scrum lifecycle begins with the creation of a product backlog which contains the requirements, features, and enhancements that need to be implemented.

**Sprint Planning**

In this phase, the set of items from the product backlog to be implemented during the upcoming sprint are selected. The sprint goal is defined.

**Sprint**

A sprint is a time-boxed iteration, usually lasting between 1 and 4 weeks. During the sprint, the researcher works on implementing the selected items from the product backlog.

**Sprint Backlog**

The sprint backlog is a subset of the product backlog, containing the items selected for implementation in the current sprint. The researcher tracks its progress and updates the sprint backlog regularly.

**Sprint Review**

At the end of each sprint, a sprint review is conducted to review the work completed during the sprint. The researcher demonstrates the implemented features, gathers feedback, and discusses any potential changes or adaptations required.

**Backlog Refinement**

Throughout the Scrum lifecycle, backlog refinement activities are carried out to refine and adjust the product backlog items. This involves clarifying requirements, estimating effort, and re-prioritizing items based on changing needs and emerging insights.

## 3.4 Neural Network to be used

This research will use the Convolutional Network architecture because they can process large amounts of data and produce highly accurate predictions. CNN is a deep neural networks, and is multilayer perceptron, which means that the CNN network is a fully connected. In any layer, each neuron is connected to all neurons in the next layer. CNN employs a mathematical operation named as convolution, where convolution is a specialized kind of linear operation.

### 3.4.1 Architecture of Convolutional Neural Networks

The Convolutional Neural Network model typically handles two-dimensional input data in its input layer. However, intrusion detection data is usually represented as one-dimensional data. To address this, the intrusion detection data is subjected to convolution using a one-dimensional convolution method.

The CNN will consists of a total of 10 layers, including an input layer, three convolutional layers to extract features from the input data, three dropout layers to mitigate overfitting and improve generalization performance, one max-pooling layer to reduce spatial dimensions while preserving important information, one fully connected layer for classification purposes, and finally an output layer that provides the desired prediction or classification result.

**Input Layer**

Following the pre-processing steps, including data set cleaning and normalization, three distinct datasets are created. The input layer, the first layer of the model, is then filled with the processed data directly.

**Convolution layer**

Convolution layers present in the network architecture are Layer 2, Layer 4, and Layer 6. An approach of decreasing kernel size and rising kernel number is will be used to construct the convolution kernel sizes for these various layers. Specifically, starting from the top layer downwards, the kernel sizes are set to 3, 2, and 1 respectively. The number of kernels in each layer is set at 16, 32, and 64 respectively. In order to extract more local features in shallow convolutional layers, a larger convolutional kernel size has been utilized.

**The Dropout layer**

To address the issue of over-fitting in the CNN model during training, a layer called dropout is utilized. Overfitting occurs when there is an imbalance in the labels of the datasets, which negatively impacts the classification performance of the model. Our model will have dropout layers at specific points within layers 3, 5, and 7 with corresponding dropout values set to 0.6, 0.5, and 0.4 respectively,

**Pooling layer**

This particular layer is designed to extract both strong and fine features from the input data. It achieves this by utilizing a maximum pooling method that compresses and removes redundant features.. Our model has layer 8 as the max-pooling layer..

**Fully Connected Layer:**

A CNN model will include fully connected layers. The model will implement one fully connected layer with a specific number of neurons set at 64.

**Output Layer**

Batch Normalization and the Adam algorithm, will be used to speed up the model's convergence time when there are only so many computational resources available.

## 3.5 Data Acquisition Techniques

Data acquisition is the process of sampling signals that measure real-world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer (contributors, Data acquisition, 2024). In this context it involves gathering network traffic data that can be used to train and evaluate the neural network for DDoS attack detection.

**Techniques:**

Network capture tools

* Tools like Wireshark or tcpdump can capture network traffic on specific interfaces or networks.

Datasets

* Publicly available datasets that can provide realistic attack traffic data.

Collaboration with network security providers

* Partnering with security companies can provide access to anonymized real-world network traffic data containing both normal and attack patterns

The researcher is going to collect a dataset from public reputable repositories because:

**1. Efficiency and Time Saving**

* Public datasets eliminate the need to set up and maintain network capture infrastructure, saving time and resources.
* Public datasets often contain a broader range of attack scenarios and variations than what might be captured on a specific network, leading to a more robust and generalizable model.

**2. Ethical Considerations**

* Capturing network traffic potentially raises ethical concerns about user consent and privacy, especially if it involves capturing data from unsuspecting users. Public datasets are typically anonymized and avoid these issues.

**3. Cost-Effectiveness**

* Using publicly available data eliminates the costs associated with deploying and maintaining network capture infrastructure.

## 3.6 Data Preprocessing Techniques

**Pre-processing** is crucial to prepare the acquired data for effective neural network training.

**Data Pre-processing Techniques:**

**Data cleaning**

* Identifying and removing irrelevant, missing, or erroneous data points to ensure the quality and consistency of the dataset.

**Normalization**

* Scaling the data to a specific range (e.g., 0-1 or -1 to 1) to improve the training process and prevent features with larger scales from dominating the learning process.

**Feature engineering**

* Extracting relevant features from the raw network traffic data that are informative for DDoS detection. Examples include packet size, flow rate, source and destination IP addresses, protocol type, and inter-arrival time between packets.

**Data balancing**

* Addressing potential class imbalance issues, where normal traffic significantly outweighs attack traffic. Techniques like oversampling (duplicating minority class samples) or undersampling (removing majority class samples) can be employed.

**Data transformation**

* Converting categorical data (e.g., IP addresses, protocols) into numerical representations suitable for neural network processing. This can involve techniques like one-hot encoding or label encoding.

## 3.7 Tools/Technology

**Programming Language :** Python – a programming language with vast applications in software development and automation. Python is widely used for machine learning and neural network development due to its extensive libraries and frameworks.

**Development Environment:** IDLE (Integrated Development and Learning Environment) – a Python IDE with advanced coding environment including a built-in code debugger, variable-value viewing and code cleaning.

**Neural Network Development** : Tensorflow - These frameworks offer high-level APIs for designing and training neural networks.

**Deployment:** Flask and React for a web-based interface and containerization with Docker Desktop for scalability

## 3.8 Conclusion

This chapter has presented a comprehensive exploration of the **methodology** employed in the development and evaluation of the neural network-based DDoS detection system. It has detailed the data acquisition and preprocessing techniques.

By meticulously outlining the chosen methodologies, this chapter has achieved the following:

**Transparency:** It provides a clear understanding of the project's execution, allowing for replication and potential improvements by others in the field.

**Justification:** It demonstrates the rationale behind the chosen techniques and highlights their effectiveness in achieving the project's objectives.

# **CHAPTER 4 SYSTEM ANALYSIS AND DESIGN**

## 4.1 Introduction

The ever-increasing reliance on internet services makes them vulnerable to malicious attacks, particularly Distributed Denial-of-Service (DDoS) attacks. These attacks overwhelm systems with traffic, rendering them unavailable to legitimate users. To address this growing threat, this project proposes a **neural network-based system** capable of efficiently detecting and classifying DDoS attacks in real-time.

This chapter delves into the **system design** of the proposed solution. It will explore the following key aspects:

* **Requirements specifications**
* **Feasibility Study**
* **System Architecture.**

## 4.2 Requirements specification

A software requirements specification is defined by (contributors, 2024) as a description of a software system to be developed. A requirements specification (RS) is a crucial document that outlines the **specific needs and functionalities** of a system before its development begins. It acts as a blueprint for both developers and stakeholders, ensuring everyone is on the same page about what the system should achieve.

### 4.2.1 Functional Requirements

These define what the system must do in terms of its core functionalities.

Detection

* Continuously analyze incoming network traffic and feed it to the trained model for predictions.
* Classify network activity as normal or indicative of a DDoS attack based on the model predictions.

Reporting

* Log all relevant events for analysis, troubleshooting, and improvement.
* Generate reports and alerts on attack detections.

### 4.2.2. Non-Functional Requirements:

These define how the system should perform in terms of its broader qualities.

Performance:

* Low latency in data processing and attack detection to ensure timely response.
* Scalability to handle high traffic volumes and potential surges.
* Resource efficiency to minimize hardware and infrastructure requirements.

Security:

* Protect sensitive data and system access with strong authentication and authorization.

Reliability:

* Ensure high availability and uptime of the system with minimal downtime.

Maintainability:

* Use modular and well-documented code for easy updates and modifications.
* Design the system for ease of monitoring and performance optimization.

Usability:

* Design a user-friendly interface for system configuration, monitoring, and analysis.

**Real-time Processing:**

* Ensure real time analysis of incoming network traffic at all times

## 4.3 Feasibility study

This section examines the feasibility of developing a DDoS attack detection system using convolutional neural networks whose theory was explained in chapter 2 section 2.4, considering technical, economic, and operational factors.

### 4.3.1 Technical Feasibility

#### 4.3.1.1 Data Availability and Quality:

* The dataset to be used is CICDDoS2019 which is readily available on the Kaagle website [link here](https://www.kaggle.com/datasets/subhrajitmajumder92/cic-ddos2019-benighn-vs-ddos). (SUBHRAJITMAJUMDER92, 2023). The CIC-DDoS 2019 dataset is the most recent planned dataset released by the Canadian Centre for Cybersecurity (CIC); it was created in a realistic test setting and incorporates the results of real network traffic assessments.

4.3.1.2 Neural Network Architecture:

* The selected neural network is convolutional network architecture because they can process large amounts of data and produce highly accurate predictions.
* CNN is a deep neural network, and is multilayer perceptron, which means that the CNN network is a fully connected.
* The architecture of the neural network was fully explained in Chapter 3 section 3.4.1.

#### **4.3.1.3 Programming Language:**

* The selected programming language is python, the IDE to be used is IDLE which is available and the researcher is already familiar with the usage of the required tools. **The convolutional neural network (CNN) built with TensorFlow package as the backend support and Keras as the application layer. The trained model is deployed using Docker containers, while the user interface is developed using React for the front-end and Flask for the back-end.**

### 4.3.2 Economic Feasibility

#### 4.3.2.1 Development and Deployment Costs:

* All tools required are open source.
* Required computing power is already available, hence all costs are covered.

#### 4.3.2.2 Return on Investment (ROI):

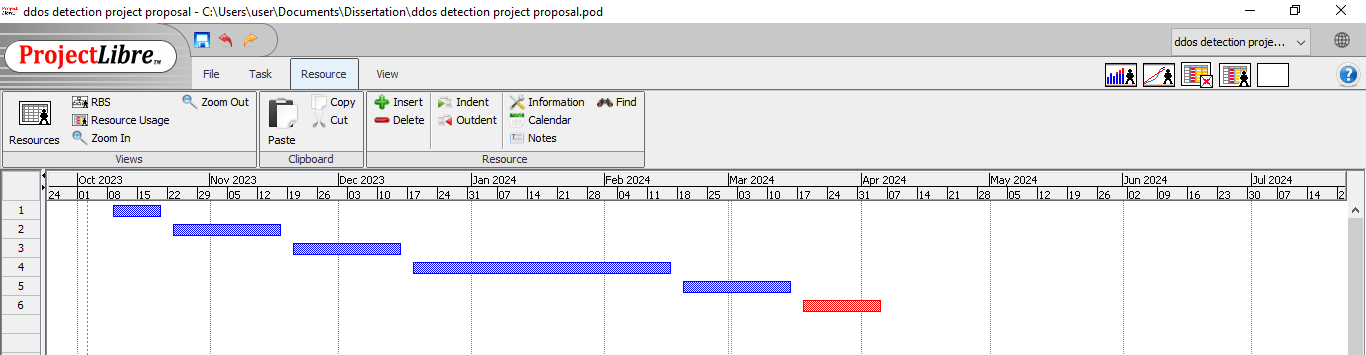
* The benefits of the system are real time network analysis, reduced downtime and improved security.

### 4.3.3 Operational Feasibility

#### 4.3.3.1 Project Timeline:

* Time allocated for development is enough to finish production and, skills are available for both the developer and end-user to fully develop and operate the system.

Figure 11 shows the project Ganttt Chart.



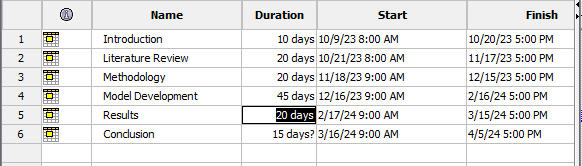


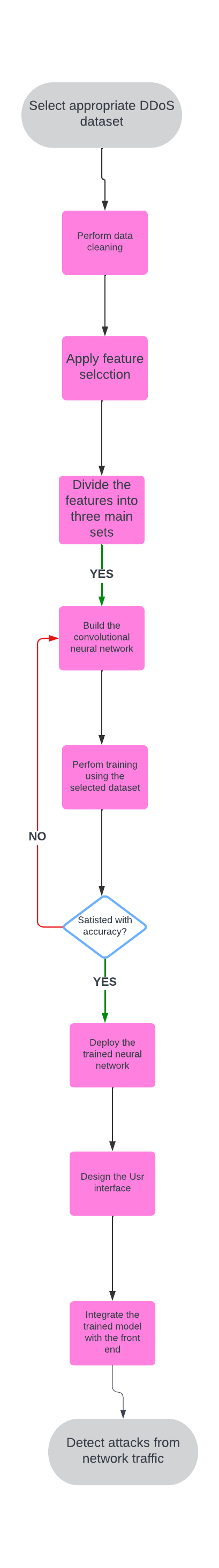
Figure 11 Project Timeline

#### 4.3.3.2 User Training and Adoption:

* There will be training needs for personnel responsible for operating and maintaining the system.
* Develop clear guidelines and procedures for using the system effectively.

Figure 12 System Flowchart

## 4.4 System Flowchart



**Data Collection**

* Selecting the appropriate dataset for model training

**Data Preprocessesing**

* Cleaning and normalization of the dataset to improve accuracy

**Feature Extraction**

* Extracts relevant features from the dataset.

**Neural Network Design**

* Building and training of the neural network responsible for DDoS detection.

**Database Design**:

* Stores relevant information:
  + **Attack Records**: Timestamps, attack types, affected resources.
  + **Model Parameters**: Neural network weights, architecture, hyperparameters.
  + **Administrative Data**: User accounts, system configuration.

**User Interface Design**

* Allows administrators to:
  + View attack records.
  + Monitor real-time alerts.

## 4.5 Use Case Diagram

**Actors:**

* **Network Administrator:** Monitors and manages the system and receives alerts.
* **Convolutional Neural Network:** Detects and generates attack reports

**Use Cases:**

**Detect DDoS Attack:**

* **Trigger:** New network traffic arrives.
* **Actor:** CNN.
* **Postcondition:** Attack is identified and classified.

**Generate Alert:**

* **Trigger:** Attack is detected.
* **Actor:** CNN.
* **Postcondition:** Network administrator receives notifications.

**Monitor System:**

* **Trigger:** Continuously.
* **Actor:** Network administrator.
* **Postcondition:** System performance and health are monitored, and issues are identified.

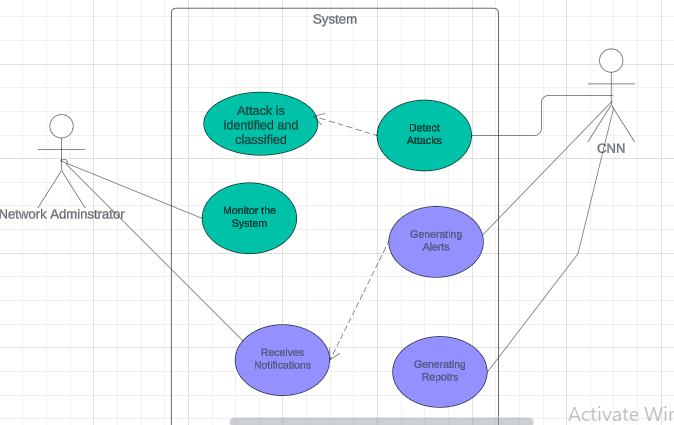


Figure 13 Use Case Diagram

The CNN continuously analyzes new network traffic, identifying and classifying attacks. If an attack is detected, the CNN triggers an alert for the network administrator who actively monitors system health and takes necessary actions, forming a comprehensive defense strategy.

## 4.6 User Interface Design

The user interface (UI) of your DDoS attack detection system plays a crucial role in its effectiveness and usability.

**Target Users:**

* **Network Administrators:** Responsible for monitoring the system's health and performance, configuring settings, and responding to alerts.
* **Security Analysts:** Investigate detected attacks and take necessary actions.
* **Data Scientists:** Manage and optimize the neural network model.

**UI Elements:**

* **Dashboard:**
* Overview of real-time attack activity, and key metrics (total devices connections, resource usage and number of DDoS attacks).
* Visualizations (charts, graphs) for detailed analysis.
* Drill-down options for attack reports and alerts.

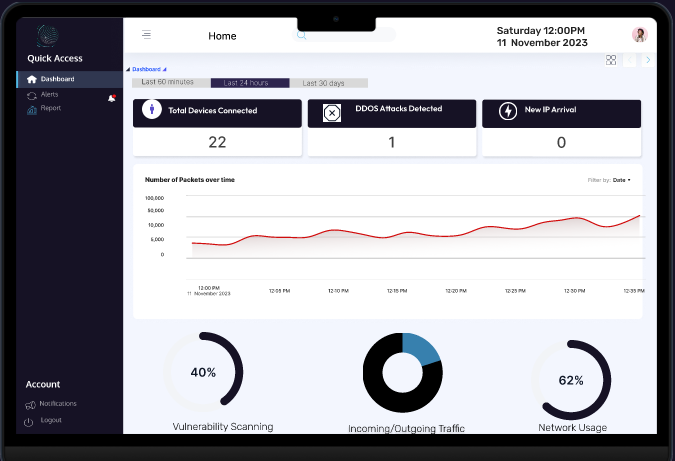


Figure 14 Dashboard UI

* **Attack Alerts:**
* List of on going and historical attacks with details like time, type, severity and origin IP.
* Filtering and search options to find specific attacks.
* Detailed view of each attack with visualizations, mitigation logs, and analyst notes.
* Filtering for records for example in the last 24hours

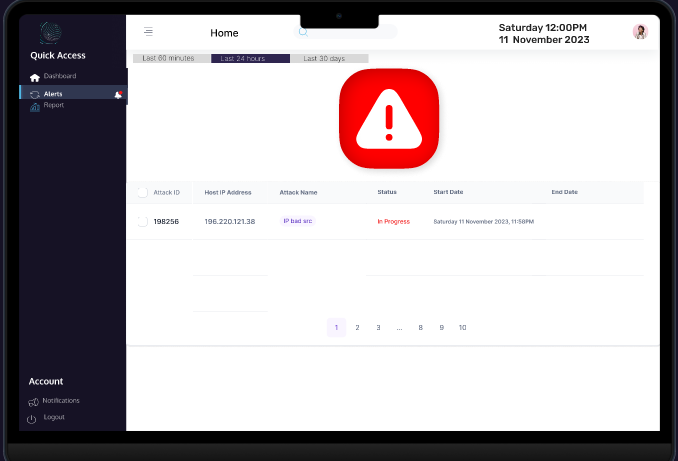


Figure 15 Alerts UI

* **Attack Reports:**
* List of all the attacks detected
* Option to filter for attacks in a specific time frame.
* Configuration of alerts and notifications.

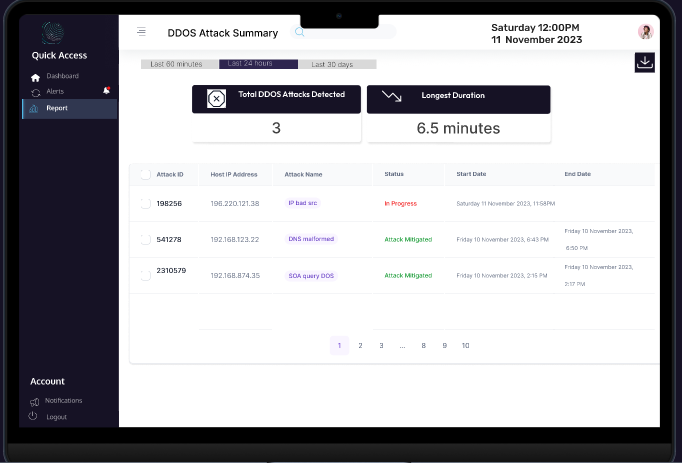


Figure 16 Reports UI

## 4.7 Database design

* Relational database to be used is MySQL for structured data like attack details and user information.

Entities:

**Attacks Records: Stores information about detected attacks:**

* Attack ID: Unique identifier for the attack.
* Start Timestamp: Time the attack was detected.
* Attack name Type: Type of attack detected (e.g., SYN flood, UDP flood, application-layer attack).
* Host IP Address: Origin IP address of the attack.

**Relationships**:

* **One-to-Many with Model Parameters**: Each attack record corresponds to a specific model configuration (e.g., neural network weights, architecture).
* **One-to-Many with Users Data**: Attack records may be associated with user accounts (e.g., the administrator who received the alert).

**Model Parameters**: Stores convolutional neural network weights and configuration:

* **Weights**: These are the learnable parameters that determine the strength of connections between neurons in different layers. During training, the neural network adjusts these weights to minimize the loss function.
* **Activation Functions**: Introduce non-linearity to the neural network.
* **Batch Size**: Determines how many samples are used in each iteration of training. Larger batch sizes can lead to faster convergence but require more memory.
* **Optimizer**: Update the model parameters based on gradients computed during back propagation.

**Relationships**:

* **Many-to-One with Attack Records**: Multiple attack records may share the same model parameters.
* **One-to-One with Administrative Data**: Model parameters are part of the system configuration.

**Users: Stores information about system users:**

* Username: Unique username for authentication.
* Password: Securely hashed password.
* Email address: Users’ email details

**Relationships**:

* **One-to-Many with Attack Records**: Administrative data helps identify responsible users during an attack.
* **One-to-One with Model Parameters**: Administrative data influences model configuration.

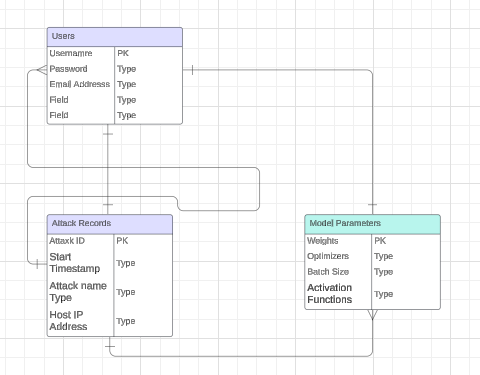


Figure 17 ERD

## 4.8 Conclusion

This chapter has presented a comprehensive design for the **system** capable of detecting and classifying DDoS attacks in real-time. By outlining the system architecture and neural network design, this chapter lays the foundation for developing a robust and effective defense against these malicious attempts.

The proposed system offers several potential **advantages**:

**Automated Detection:** The system can continuously analyze network traffic, reducing the burden on human administrators and enabling faster response times.

**Improved Accuracy:** Neural networks can learn complex patterns in data, potentially leading to more accurate attack detection compared to traditional methods.

**Scalability:** The system can be potentially adapted to handle diverse network environments and varying traffic volumes.

# **CHAPTER 5 DESIGN AND IMPLEMENTATION**

## 5.1 Introduction

The previous chapter established a comprehensive design for a convolutional **neural network-based system** capable of detecting and classifying DDoS attacks. This chapter takes the next crucial step by delving into the **design and implementation** aspects of the proposed system.

Here, we will bridge the gap between theoretical design and practical implementation by addressing the following key areas:

* **Hardware and Software Requirements**.
* **Implementation Details**
* **Testing Plan**

By meticulously addressing these aspects, this chapter aims to provide a clear and detailed roadmap for translating the conceptual design into a functional and robust system. Through careful implementation and rigorous testing, the objective is to create a reliable DDoS detection system that uses convolutional neural networks, contributing to a more secure and resilient internet landscape.

## 5.2 Hardware and Environment Setting

A personal PC with an Intel Core i3 processor running at 2.40 GHz was used to conduct the research as shown in Figure 1. The studies were also carried out in setups that used Windows 10, Python 3.11.6, Tensorflow 2.15.0, NumPy 1.23.5, Scikit-learn 1.3.2, Pandas 2.1.4, and Keras 2.15.0.

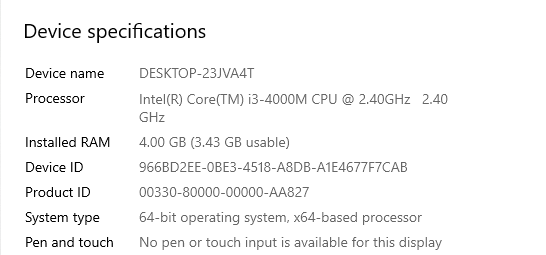


Figure 19 Device Specifications

## 

## 5.3 Development Tools

### **5.3.1 Integrated Development Environment (IDE)**

An integrated development environment (IDE) is a [software application](https://en.wikipedia.org/wiki/Application_software) that provides comprehensive facilities for [software development](https://en.wikipedia.org/wiki/Software_development). An IDE normally consists of at least a [source-code editor](https://en.wikipedia.org/wiki/Source-code_editor), [build automation](https://en.wikipedia.org/wiki/Build_automation) tools, and a [debugger](https://en.wikipedia.org/wiki/Debugger). (contributors, Integrated development environment, 2024). The researcher selected IDLE as the IDE.

**Justification of using IDLE**

**Simple and lightweight**

 IDLE comes bundled with Python installations, making it easy for the researcher to use and has a user-friendly interface

**Interactive shell**

 IDLE provides an interactive shell where the researcher can directly execute Python code line by line, experiment, and get immediate feedback.

**Basic debugging**

It offers basic debugging functionalities like stepping through code and inspecting variables, which can help identify errors in smaller scripts.

### **5.3.2 Deployment Tool**

Docker desktop is a one-click-install application for your Mac, Linux, or Windows environment that lets you build, share, and run containerized applications and microservices. (Docs, 2024)

**Justification for using Docker Desktop**

**Consistency and Isolation**

Docker creates containers that package the model, its code, and all its dependencies (libraries, frameworks) into a single unit. This ensures that the model runs consistently across different environments, regardless of the underlying operating system or pre-installed software. This isolation also prevents conflicts between the model's dependencies and other software on the system.

**Portability and Scalability**

Docker containers are portable across different computing environments, from local machines to cloud platforms. This makes it easy to deploy models on different systems without worrying about compatibility issues. Additionally, containers are lightweight and can be easily scaled up or down based on the model's resource needs.

### **5.3.4 API Development Tool**

API is the acronym for application programming interface which is a software intermediary that allows two applications to talk to each other. APIs are an accessible way to extract and share data within and across organizations.Flask APIs provide the researcher a way to build a web system that other applications can interact with using HTTP requests (GET, POST, PUT, DELETE).

**Justification for using Flask APIs**

**Lightweight and Easy to Learn**

Flask has a minimal core, making it a good choice for smaller projects

**Flexible**

Flask allows the researcher to build APIs tailored to specific needs.

**Scalable**

 Flask APIs can grow and adapt as your project's requirements evolve.

### **5.3.5 Database Management Tool (DBMT)**

MySQL is an open-source [Relational Database Management System](https://www.hostinger.com/tutorials/dbms) (RDBMS) that enables users to store, manage, and retrieve structured data efficiently (G, 2024). The system is using Mysql database to store user login details, model parameters and attack records.

**Justification for using MySql**

**Cost Effective**

MySQL is an open-source database, meaning the researcher can use and modify it freely without licensing fees since the project is running on a limited budget.

**Ease of Use**

MySQL is known for its user-friendly interface and simple query language (SQL), making it easy to learn and administer. This allows for quick set up, and management saving up on the project time.

**Robust and Reliable**

MySQL has a proven track record of reliability and stability, with a large community of users and developers contributing to its ongoing development and maintenance. This ensures consistent performance and reduces the risk of data loss or corruption.

## 5.4 System Implementation

### 5.4.1 Data Collection

Data collection is the process of gathering data relevant to your [AI project’s](https://labelyourdata.com/articles/lifecycle-of-an-ai-project-stages-breakdown) goals and objectives. You eventually obtain a [dataset](https://labelyourdata.com/articles/what-is-dataset-in-machine-learning), which is essentially your collection of data, all set to be trained and fed into a Machine Language model (Kniazieva, 2022).

The dataset to be used is CICDDoS2019 collected form Kaagle website [link here](https://www.kaggle.com/datasets/subhrajitmajumder92/cic-ddos2019-benighn-vs-ddos). The dataset was uploaded by SUBHRAJITMAJUMDER92 and was updated 6 months ago. The dataset is a subset of the CICDDoS2019 dataset by (Sharafaldin, 2019).

#### 5.4.1.1 Dataset Description

The CIC-DDoS 2019 dataset is the most recent planned dataset released by the Canadian Centre for Cybersecurity (CIC); it was created in a realistic test setting and incorporates the results of real network traffic assessments. CICDDoS2019 dataset is labelled with 70 network traffic features that were extracted and calculated for all benign and denial of service flows. Out of the several features in the dataset some of the features are displayed in Table 2.

Table 3 List of Features

|  |  |
| --- | --- |
| Protocol | Max Packet Length |
| Inbound | Timestamp |
| ACK Flag Count | Flow IAT Min |
| Active Max | Flow Duration |
| Protocol | Idle Min |
| Inbound | Active Min |
| Idle Max | Idle Std |
| Min Packet Length | Label |
| Flow Bytes/s | ECE Flag Count |
| Active Mean | Idle Max |
| Flow Packets/s | Packet Length Variable |
| Flow IAT Max | Active Std |

The dataset consists of 225460 records of which 112732 are benign and 112729 are ddos attacks.

|  |  |
| --- | --- |
| Parameter name | Total # |
| Total number of records | 225460 |
| Total number of features | 70 |
| Total number of labels | 2 |
| Total number of normal records | 112732 |
| Total number of attack records | 112729 |
| % of normal records | 50.1% |
| % of attack records | 49.9% |

Table 4 Cicddos2019 dataset general statistics

The CICDDoS2019 dataset will be divided into two subsets: the training subset, and the testing subset. The training subset contains samples of data used to fit the machine learning models, whereas the testing subset is to assess the performance of the trained machine learning model.

### 5.4.2 Data Preparation

Data preparation involves transforming raw data into a format where the machine learning algorithms can deal with, in order to uncover insights or make predictions.

#### 5.4.2.1 Data Cleaning

Data cleaning includes identifying and correcting errors or mistakes in the dataset that might affect the efficiency of the machine learning model.

As stated above, the dataset has 70 features. The table below shows the features.

|  |  |
| --- | --- |
| Name of the Feature | List of Features |
| Forward Packet  Backward Packet  Flow-based  Packet Header-based  Packet Payload-based | 'Protocol', 'Flow Duration', 'Total Fwd Packets',  'Total Backward Packets', 'Total Length of Fwd Packets',  'Total Length of Bwd Packets', 'Fwd Packet Length Max',  'Fwd Packet Length Min', 'Fwd Packet Length Mean',  'Fwd Packet Length Std', 'Bwd Packet Length Max',  'Bwd Packet Length Min', 'Bwd Packet Length Mean',  'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s',  'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min',  'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max',  'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std',  'Bwd IAT Max', 'Bwd IAT Min', 'Bwd PSH Flags', 'Fwd Header Length',  'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',  'Min Packet Length', 'Max Packet Length', 'Packet Length Mean',  'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',  'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',  'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio',  'Average Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size',  'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',  'Subflow Bwd Bytes', 'Init Win bytes forward',  'Init Win bytes backward', 'act data pkt fwd', 'min seg size forward',  'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean',  'Idle Std', 'Idle Max', 'Idle Min', 'Inbound', 'Label' |

Table 5 List of the 70 Features in the dataset

We noticed that several attributes (columns) contain zero values. For instance Bwd PSH Flags, FIN Flag Count and PSH Flag Count which were dropped. We noted an unnamed column and it was dropped. The data was checked for infinite or missing values using the inf() method and no rows were found. In total, 4 attributes (columns) were removed from the dataset.

### 5.4.3 Feature Selection

There were 70 features in the CICDDoS2019 dataset and after cleaning the data, 66 were left. Many features make the training and prediction tasks are very difficult. Therefore, it is important to minimize the number of features in the dataset. Minimizing the number of features may lead to several benefits, including accuracy improvement, speed up in the training process and data visualization.

For this stage, we selected XGBoost model for classification. We encode the labels in the Label column such that 0 represents benign and 1 DDoS attack. This was the target variable(y). The 65 features excluding ‘label’ were added into selected feature dataset which is the ‘data’(x).

The CICDDoS2019 dataset was divided into two subsets: training subset, and testing subset, with 80% for training and 20% for testing.  [The](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) model was fit on the dataset and the importanceswere retrieved for each feature. The table below shows the distribution of the top 10 importances.

|  |  |
| --- | --- |
| Feature | Importance |
| Min Packet Length | 0.697 |
| URG Flag Count | 0.189 |
| Inbound | 0.096 |
| Init Win bytes forward | 0.003 |
| SYN Flag Count | 0.003 |
| ACK Flag Count | 0.003 |
| Average Packet Size | 0.002 |
| Total Backward Packets | 0.001 |
| Fwd Packet Length Min | 0.001 |
| Fwd Packet Length Max | 0.001 |

Table 6 Importance Scores of Features

The table below provides a description of the important characteristics of the dataset.

|  |  |
| --- | --- |
| Feature | Description |
| Min Packet Length | Minimum length of a flow packet |
| Inbound | Indicates the direction of the network traffic, specifically that it is incoming. |
| Fwd Packet Length Min | Smallest possible packet size within the forward route (Information sent to slaves from masters in the attacks). |
| Fwd Packet Length Max | Packet size limit in the forward (outgoing) direction. |
| Average Packet | This is the average size of all packets in the flow. |
| URG Flag count | Number of packets with URG. It is used to indicate that the data contained in the packet should be prioritized and handled urgently by the receiver. |
| Init Win bytes forward | Number of bytes sent in the initial window in the forward direction |
| SYN Flag Count | Number of packets with SYN. used in first step of [connection establishment](https://www.geeksforgeeks.org/computer-network-tcp-connection-establishment/) phase or 3-way handshake process between the two hosts. |
| Total Backward Packets | Total packets in the backward direction |
| ACK Flag Count | This refers to the number of packets in the flow that have the ACK flag set, indicating that they are acknowledgment packets |

Table 7 Features of Dataset Used

### **5.4.4 Convolutional Neural Network**

**Importing the Necessary Libraries**

* numpy for numerical operations
* pandas for data manipulation
* tensorflow for building the neural network
* sklearn for data preprocessing and evaluation
* time for measuring execution time
* seaborn and matplotlib for visualizing results

**Loading and Preprocessing Data**

* Read the CSV file containing network traffic data using pd.read\_csv().
* Separated the target variable (Label) from the features.
* Applied label encoding to convert string labels to numerical values using LabelEncoder.
* Split the data into training, validation, and testing sets using train\_test\_split() with stratification to maintain class balance.
* Scaled the features to a standard range (0 to 1) using MinMaxScaler().

**Reshaping Data**

* Reshaped the features into a 3D array with shape (samples, timesteps, features) for compatibility with 1D convolutional layers.

The CNN model consists of a total of 10 layers, including an input layer, three convolutional layers, three dropout layers, one Global max-pooling, one dense layer and an output layer with sigmoid activation as explained in Chapter 3 Section 3.4.

Shown below is the code showing the convolutional neural network :

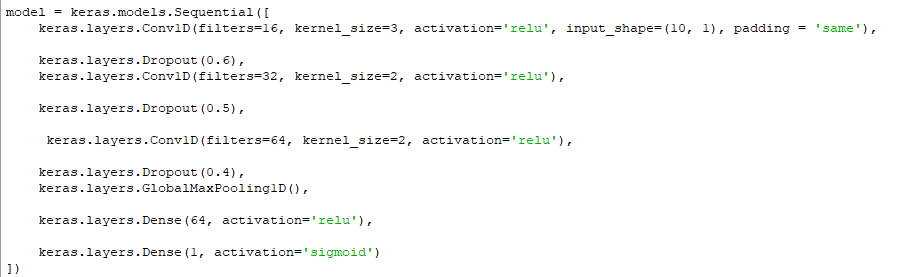


Figure 20 Convolutional neural network Code

#### 5.3.4.1 Justification of the used Python libraries

* **Powerful Combination**

TensorFlow provides a powerful and flexible backend for various deep learning tasks, offering low-level control and scalability for complex models. Keras acts as a high-level API built on top of TensorFlow, offering a user-friendly and concise interface for building, training, and evaluating models. This combination allows the researcher to leverage the strengths of both frameworks.

* **User-Friendliness and Ease of Use**

Keras is known for its simpler and more intuitive syntax, making it easier to learn and use.

* **Extensive Community and Resources**

Both TensorFlow and Keras benefit from large and active communities, offering abundant resources like tutorials, documentation, and forums. This makes it easier to find help, learn best practices, and troubleshoot any issues you might encounter.

* **Integration with Other Tools**

Both TensorFlow and Keras integrate well with other popular scientific computing libraries like NumPy, SciPy, and pandas, making it easy to combine them for data manipulation and analysis tasks within your machine learning workflow.

In conclusion, using Keras and TensorFlow together provides a powerful, user-friendly, and versatile toolkit for various deep learning and machine learning projects. Their combination offers ease of use, extensive resources, scalability, and the ability to handle a wide range of applications.

### 5.4.5 Compilation and Training

The training data for the model consisted of preprocessed dataset labelled as either normal or containing DDoS attacks. This data was split into two sets:

* **Training set:** Used to train the model and update its weights during training. This constituted the majority of the available data of 80%.
* **Validation set:** Used to monitor the model's performance during training and prevent overfitting. This constituted a smaller portion of the data of 20%.

**Optimizer**

The chosen optimizer for training the neural network was **Adam**. The Adam optimizer, short for **Adaptive Moment Estimation**, is an optimization algorithm commonly used in deep learning to efficiently update the weights of a neural network during training. Its goal is to minimize the loss function, ultimately leading the network to make better predictions. (Vashwa, 2023).

Adam Optimizer was selected because it employs an **adaptive learning rate** for each individual parameter (weight) in the network and requires **minimal hyper parameter tuning**.

**Loss Function**

The selected loss function was binary crossentropy. This function measures the discrepancy between the model's predictions and the true labels. Binary cross-entropy (BCE) is a loss function commonly used in **binary classification** problems, where the model's output is a probability between 0 and 1 representing the likelihood of an event belonging to one of two classes (Saxena, 2023).

**Formula:**

The mathematical formula for BCE is:

BCE = - (y \* log(p) + (1 - y) \* log(1 - p))

* **y**: True label (0 or 1)
* **p**: Predicted probability (between 0 and 1)

**Number of Training Epochs**

The model was trained for a total of **10 epochs**. An epoch represents one complete pass through the entire training dataset. We chose this number of epochs based on Training neural networks can be computationally expensive, especially for large datasets. The researcher selected 10 epochs as a **balance between achieving reasonable performance and training time.**

**Batch Size**

A batch size of 200 refers to the number of samples processed by the model during each **forward pass** and **backward pass** before updating its internal parameters (weights and biases). Using a smaller batch size like 200 requires less memory to store the input data and intermediate calculations during a single training step.

**Verbose**

The verbose was set at 2 because higher values (e.g., 1 or 2) provide more detailed information about the training progress, including loss and accuracy values.

**Evaluation Performance on Test Set**

model.evaluate() was used to calculate test loss and accuracy and the metrics are shown below



Figure 21 Test Set Evaluation Matrix

**Prediction on test set**

model.predict() was used to get predictions on the test set and the predictions were converted to binary classes using a threshold of 0.5.

#### 5.5.1.4 Deployment

The trained model was saved as a keras model along with its weights and deployed using Docker desktop.

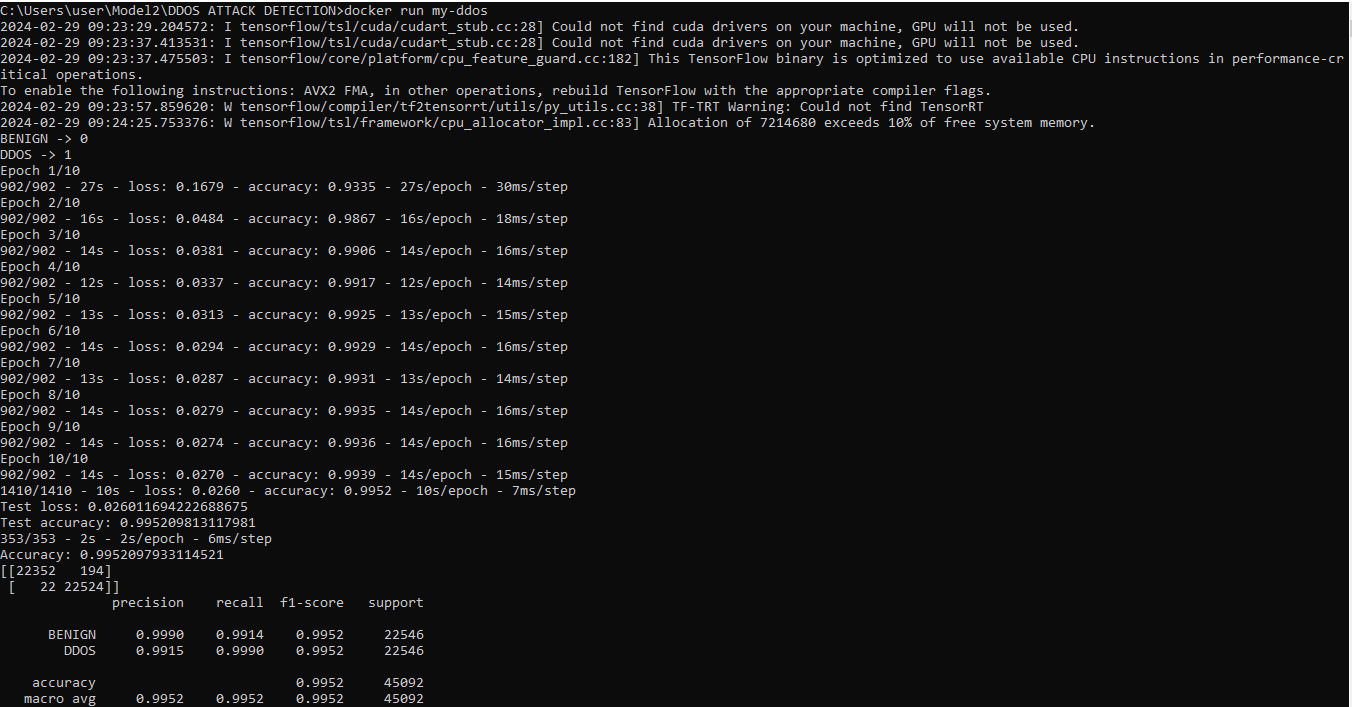


Figure 22 Docker Deployment

Figure 11 shows the deployed model on docker after executing the docker run command.

### 5.4.6 Database



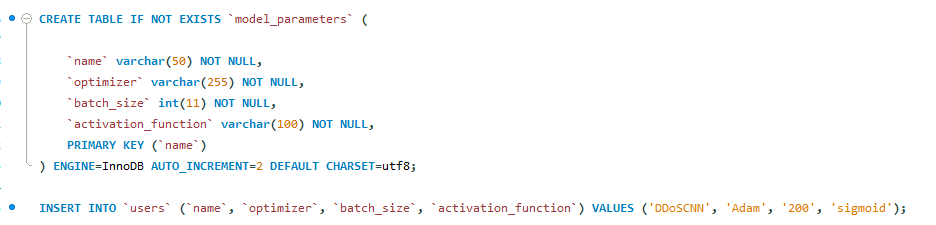


Figure 23 MySql Code

The code above initializes a database for the DDoS detection system. It creates tables to store user credentials, attack records, and model configuration details

**1. Database and Character Set**

* Created a database named DDoS if it doesn't already exist.
* Set the default character set to utf8 for text data within the database.

**2. User Table**

* Createed a table named users to store user information.
* The table has four columns:
  + id: An auto-incrementing integer primary key (unique identifier for each user).
  + username: A string to store the user's username (maximum length 50 characters).
  + password: A string to store the user's password (maximum length 255 characters).
  + email: A string to store the user's email address (maximum length 100 characters).

This table stores credentials for users who can access the DDoS detection system.

**3. Attack Records Table:**

* Created a table named attack\_records to store information about detected DDoS attacks.
* The table has five columns:
* id: An auto-incrementing integer primary key (unique identifier for each attack record).
* host\_ip\_address: A string to store the IP address of the suspected attacker (maximum length 50 characters).
* start\_timestamp: A string to store the timestamp when the attack started (maximum length 255 characters).
* status: A string to store the current status of the attack (e.g., "InProgress", "Mitigated") (maximum length 100 characters).
* end\_timestamp: A string to store the timestamp when the attack ended (maximum length 255 characters, currently empty).

This table gets populated by the DDoS detection system when it identifies potential attacks.

**4. Model Parameters Table:**

* Created a table named model\_parameters to store the configuration details of the DDoS detection model.
* The table has four columns:
* name: A string to store the name of the model (e.g., "DDoSCNN") (maximum length 50 characters, primary key).
* optimizer: A string to store the optimizer used during training (e.g., "Adam") (maximum length 255 characters).
* batch\_size: An integer to store the batch size used during training (number of samples processed together)
* activation\_function: A string to store the activation function used in the model (e.g., "sigmoid") (maximum length 100 characters)

This table helps track the configuration used for the deployed DDoS detection model.

### 5.4.7 User Interface

## 5.5 Test Plan

### 5.5.1 Testing Scope

* This test plan covers the functionality, performance, and robustness of the DDoS detection system.
* Specific areas of testing include:
  + Data preprocessing pipeline.
  + Model training and hyperparameter tuning.
  + Model prediction accuracy on various attack scenarios.

### 5.5.2 Test Cases

**Data Preprocessing**

* Test Case 1: Ensure proper feature engineering and selection of relevant features from the network traffic data.

**Model Training and Hyperparameter Tuning**

* Test Case 2: Verify that the training process completes successfully without errors.
* Test Case 3: Evaluate the impact of different hyperparameter settings on model performance using grid search or other optimization techniques.
* Test Case 4: Monitor training and validation loss/accuracy curves to identify potential overfitting or underfitting issues.

**Model Prediction Accuracy**

* Test Case 5: Calculate performance metrics like precision, recall, F1-score, and confusion matrix for a comprehensive evaluation.

## 5.6 Testing

Table 8 Test Case Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Intended Results | Actual results |
| 1 | Test if the system can effectively extract relevant features from the network traffic data. | * The system should use a well-defined set of features that effectively capture the characteristics of network traffic relevant to DDoS attacks. | Successful |
| 2 | * Verify that the model training process runs successfully without errors or unexpected termination. | The training process should complete successfully, with logs indicating convergence or reaching a pre-defined stopping criterion | Successful |
| 3 | * Assess the impact of different hyperparameter settings on the model's performance. | * There should be a noticeable difference in performance between different hyperparameter settings. This indicates the model is sensitive to these parameters and optimization is necessary. | The system was reactive to changes in number of convolutional layers |
| 4 | Identify potential issues with overfitting or underfitting during training | The validation loss should be low | The system detected attacks on test data with a loss of 0.02% which is very low |
| 5 | * Evaluate the model's ability to accurately detect DDoS attacks | The system should identify the traffic as a DDoS attack with a high accuracy score. | The system detected attacks on test data with an accuracy of 0.99% which is high |

## 5.7 Conclusion

This chapter comprehensively explored the design and implementation aspects of the development of a DDoS System that detects attacks using convolutional neural networks. The researcher discussed in detail the implementation tools and how they have been used. Additionally, we delved into the implementation details, providing insights into the codes, libraries, and frameworks utilized to bring the design to life.

# **CHAPTER 6 RESULTS EVALUATION AND CONCLUSION**

## 6.1 Introduction

The implement of a neural network-based DDoS detection system capable of identifying DDoS attacks in real-time was the main goal of the researcher’s project. This chapter covers the evaluation metrics such as accuracy and f1 score of the trained convolutional neural network. The conclusion whether the aim and objectives were met is discussed and future work recommended.

## 6.2 Evaluation Metrics

The researcher employed a variety of measures to analyse the models’ performance. As explained by (Brownlee, 2020) the concept of Confusion Matrix classification can produce 4 possible outcomes which are:

• True Positive (TP): A classification where predicted class matches targeted.

• True Negative (TN): A classification where predicted class matches inverse target class.

• False Positive (FN): A classification where predicted class is the targeted class but was supposed to be the inverse.

• False Negative (TN): A classification where predicted class was supposed to the target class but model predicts the inverse target class

These counts of the outcomes are used to come up with the metrics used in this research including recall, accuracy, precision and F1 score.

**Accuracy**

Accuracy is a metric that expresses the overall percentage of false alerts and false alarms produced by an IDS model; it indicates the overall success rate of all IDS and is computed as

Accuracy (A) = (TP + TN) / (TP + TN + FP + FN)

TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative

**Precision**

The false negative rate (FNR), also known as accuracy, is the ratio of incorrectly classified attacks to all attack incidences. The accuracy obtained from the following formula reveals how many precise positive forecasts are made.

Precision = 𝑇𝑃 ⁄ (𝐹𝑃 + 𝑇𝑃)

**Recall**

The detection rate (DR), sometimes referred to simply as the true positive rate (TPR) or recall, seems to be the proportion of malicious incidents that have been correctly recognized relative to all harmful vectors. The recall calculation equation shows how many true positives are accurately predicted:

Recall = ⁄ (𝑇𝑃 + 𝐹𝑁)

**F1 score**

F1 score is a harmonic mean of Precision and Recall. It is important because it offers more details regarding how well the IDS performed. False positives and negatives are considered. Particularly once the dispersion of classifiers is irregular or imbalanced, the F1 score is beneficial. The F-score indicates the reliability of memory and responsiveness and may be determined using the equation below.

F1 = (2 ∗ 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛) ⁄ (𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + 𝑅𝑒𝑐𝑎ll)

**Average Training Time**

This refers to the total time required to train the machine learning model.

Figure 13 shows the confusion matrix of the model.

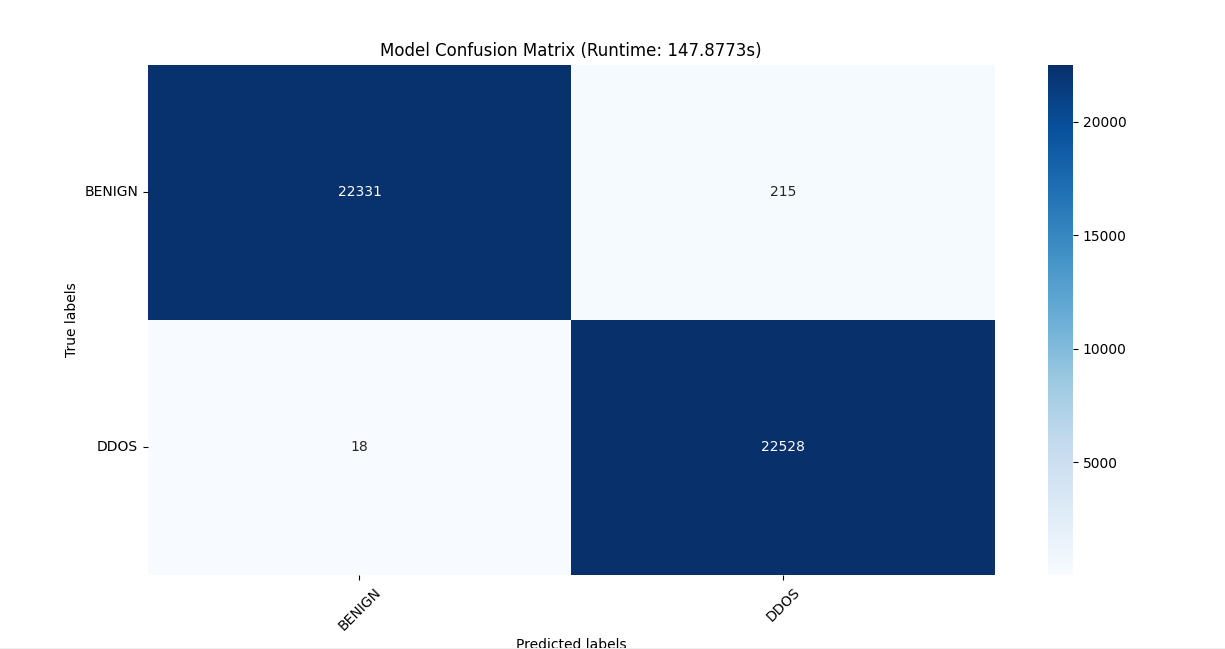


Figure 24 Confusion Matrix

* **22331:** The number of **true positives (TP)**. These are instances where the model correctly predicted the positive class.
* **215:** The number of **false positives (FP)**. These are instances where the model incorrectly predicted the positive class, when they actually belonged to the negative class.
* **18:** The number of **false negatives (FN)**. These are instances where the model incorrectly predicted the negative class, when they actually belonged to the positive class.
* **22528:** The number of **true negatives (TN)**. These are instances where the model correctly predicted the negative class.

Based on the values, the model performed well in predicting the negative class, with a high number of true negatives (22528) and relatively low false positives (215).

The average training time was 147s.

Figure 14 shows the classification report.

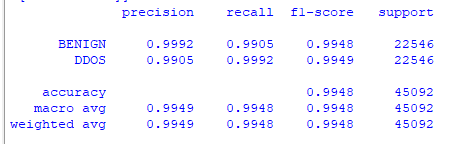


Figure 25 Classification Report

**BENIGN**

This line shows the evaluation metrics for the **benign class** (normal network traffic).

* **precision**: 0.9992 (very high), indicating a low number of false positives (incorrectly classified attacks).
* **recall**: 0.9905 (high), indicating a relatively low number of false negatives (missed attacks).
* **f1-score**: 0.9948 (high), showing a balanced performance between precision and recall.
* **support**: 22546 (number of benign instances in the dataset).

**DDOS**

This line shows the evaluation metrics for the **DDOS class**(attack traffic).

* **precision**: 0.9905 (very high), indicating a low number of false positives (incorrectly classified benign traffic as attacks).
* **recall**: 0.9992 (very high), indicating a very low number of false negatives (missed attacks).
* **f1-score**: 0.9949 (very high), showing a balanced performance between precision and recall.
* **support**: 22546 (number of DDOS instances in the dataset).

## **6.3 Summary of findings**

* **accuracy**: 0.9948 (very high), indicating the model correctly classified a high proportion of all instances.
* **macro avg**: Averages the precision, recall, and F1-score across both classes, showing similar performance for both benign and DDOS detection.
* **weighted avg**: Similar to macro avg, but weights the average based on the class support (number of instances). In this case, both averages are very close, as the class sizes are balanced.

These results show that the model performs very well in both detecting DDOS attacks and classifying normal network traffic. It has a low rate of both false positives and false negatives, indicating high accuracy and reliable performance.

## 6.4 Review of Project Aim and Objectives

The aim of the project was to design and implement a neural network-based DDoS detection model capable of identifying DDoS attacks in real-time.

The objectives were:

* To distinguish distributed denial of service (DDoS) attacks from normal network traffic in real time
* To alert the network administrator regarding any attack
* To produce reports on detected attacks in a network

At the time of documenting the system was able to distinguish between ddos attacks and normal attacks, generate reports and alerts. All objectives were successfully met.

The system offers several potential **advantages such as**:

**Automated Detection:** The system can continuously analyze network traffic, reducing the burden on human administrators and enabling faster response times.

**Improved Accuracy:** Convolutional Neural networks can learn complex patterns in data, potentially leading to more accurate attack detection compared to traditional methods.

**Scalability:** The system can be potentially adapted to handle diverse network environments and varying traffic volumes.

## 6.5 Problems Encountered

**Limited availability of labelled DDoS attack data:** Acquiring a free, sufficient and well-labelled data for training neural networks was difficult.

**Computational resources:** Training and deploying the convolutional neural network required significant computational resources, especially for complex so the researcher had to reduce the complexity of the model.

**API integration:** The researcher faced challenges in integrating the model with Flask API for smooth communication.

## 6.5 Future Work

* **Imbalanced class handling:** Implement more advanced techniques for handling imbalanced data sets. This could involve cost-sensitive learning algorithms, or class weighting during training.
* **Deep neural network architectures:** Explore deeper CNN architectures (e.g., ResNet, Inception) or investigate alternative neural network architectures like recurrent neural networks (RNNs) or transformers, especially if the temporal dependencies in network traffic data are complex.

## 6.7 Conclusion

The system fulfilled all the objectives that were set at the beginning of the project. The system was able to distinguish between normal and ddos attacks and generate alerts and reports. The system was developed as a web application for easy management. The challenges encountered were solved with the help of the supervisor. The project had added to the body of knowledge of combating cybersecurity attacks.

# References

*cloudflare*. (2023). Retrieved September 28, 2023, from what-is-a-ddos-attack: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/#:~:text=A%20distributed%20denial%2Dof%2Dservice%20(DDoS)%20attack%20is,a%20flood%20of%20Internet%20traffic.

Akilandeswari, V., & Shalinie, S. (2012). Probabilistic Neural Network based attack traffic classification. *Fourth International Conference on Advanced Computing (ICoAC)*, 1-8.

Ali Shiravi, H. S. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. In H. S. Ali Shiravi, *Computers & Security, Volume 31* (pp. 357-374).

Fernandes, M. E. (2023, September 1). *ddos-protection-market-size-trends-analysis-scope-growth*. Retrieved October 3, 2023, from https://www.verifiedmarketresearch.com/: https://www.openpr.com/news/2734080/ddos-protection-market-size-trends-analysis-scope-growth

Gupta, B., Joshi, C., & Misra, M. (2011). ANN Based Scheme to Predict Number of Zombies in a DDos Attack. *International Journal of Network Security, VOL. 13*, 216–225.

kentik. (2023). *kentipedia*. Retrieved September 28, 2023, from ddos-detection/: https://www.kentik.com/kentipedia/ddos-detection/

Li, J., Liu, Y., & Gu, L. (2009). DDoS attack detection based on neural network. *2nd Inter-national Symposium on Aware Computing (ISAC)*, 196 – 199.

Siaterlis, C., & Maglaris, V. (2005). Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics. *Proceedings of the 10th IEEE Symposium. on Computers and Communications, (ISCC)*, 469 – 475.

*cloudflare*. (2023). Retrieved September 28, 2023, from what-is-a-ddos-attack: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/#:~:text=A%20distributed%20denial%2Dof%2Dservice%20(DDoS)%20attack%20is,a%20flood%20of%20Internet%20traffic.

*ddos-attacks*. (2023). Retrieved October 5, 2023, from imperva: https://www.imperva.com/learn/ddos/ddos-attacks/#:~:text=Distributed%20Denial%20of%20Service%20Attack%20(DDoS)%20Definition,-A%20distributed%20denial&text=It%20is%20distinct%20from%20other,a%20target%20with%20malicious%20traffic.

A. Ghaben, M. A. (2021). Mathematical Approach as Qualitative Metrics of Distributed Denial of Service Attack Detection Mechanisms. *IEEE Access, vol. 9*, 123012-123028.

Adefemi Alimi KO, O. K.-M. (2022). Refined LSTM Based Intrusion Detection forDenial-of-Service Attack in Internet of Things. *Journal of Sensor and Actuator Networks*.

Ali Mustapha, R. K. (2023). Detecting DDoS attacks using adversarial neural network. *Computers & Security Volume 127*.

Ayman Ghaben, M. A. (2021). Mathematical Approach as Qualitative Metrics of Distributed Denial of Service Attack Detection Mechanisms. *IEEEAccess*.

Baheti, P. (2021, 07 8). *The Essential Guide to Neural Network Architectures*. Retrieved 10 31, 2023, from V7labs: https://www.v7labs.com/blog/neural-network-architectures-guide#:~:text=are%20Neural%20Networks%3F-,Standard%20Neural%20Networks,Generative%20Adversarial%20Network%20(GAN)

Choi RY, C. A.-C. (2020, 02 27). *Introduction to Machine Learning, Neural Networks, and Deep Learning*. Retrieved 10 31, 2023, from Transl Vis Sci Technol: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7347027/#:~:text=An%20artificial%20neural%20network%20(ANN,inspired%20by%20biological%20neural%20networks.&text=Each%20ANN%20contains%20nodes%20(analogous,analogous%20to%20axons%20and%20dendrites).

Gawande, A. (2018). *DDoS detection and mitigation using machine learning (Doctoral dissertation, Rutgers University-Camden Graduate School).* New Jersey: rucore.libraries.rutgers.edu.

kentik. (2023). *kentipedia*. Retrieved September 28, 2023, from ddos-detection/: https://www.kentik.com/kentipedia/ddos-detection/

Kirtas, M. (2022). Early Detection of DDoS Attacks using Photonic Neural Networks. *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)* (pp. 1-5). Nafplio: IEEE.

Rangapur, A. (2022, 01 24). *DDoSDet: An approach to Detect DDoS attacks using Neural Networks*. Retrieved 10 31, 2023, from arxiv.org: https://arxiv.org/abs/2201.09514

WIKI. (2023). *neural-network-architectures*. Retrieved October 9, 2023, from h2o.ai: https://h2o.ai/wiki/neural-network-architectures/#:~:text=What%20Is%20Neural%20Network%20Architecture,power%20of%20a%20human%20brain.

Brownlee, J. (2020, August 20). *calculate-feature-importance-with-python*. Retrieved January 10, 2024, from machinelearningmastery: https://machinelearningmastery.com/calculate-feature-importance-with-python/

Kniazieva, Y. (2022, September 15). *data-collection-methods-AI*. Retrieved January 11, 2024, from labelyourdata: https://labelyourdata.com/articles/data-collection-methods-AI

MEHTA, S. (2023, March 30). *types-research-methodology*. Retrieved January 10, 2024, from eduvoice: https://eduvoice.in/types-research-methodology/

rikkeisoft. (2022, December 14). *Methodologies-in-software-development*. Retrieved November 16, 2023, from rikkeisoft: https://rikkeisoft.com/blog/methodologies-in-software-development/

Sharafaldin, I. L. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. *International Carnahan Conference on Security Technology (ICCST* (pp. 1-8). IEEE.

Team, I. E. (2023, August 21). *software-development-methodology*. Retrieved January 10, 2024, from Indeed.com: https://au.indeed.com/career-advice/career-development/software-development-methodology

Team, S. E. (2017, March 27). *top-4-software-development-methodologies*. Retrieved January 10, 2024, from Synopsys: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies.html

tiffin.edu. (2022, August 2). *whatareresearchmethodologies*. Retrieved January 10, 2024, from library.tiffin.edu: https://library.tiffin.edu/researchmethodologies/whatareresearchmethodologies

Witwatersrand, U. o. (2023, July 2023). *libguides.wits.ac.za*. Retrieved November 16, 2023, from University of the Witwatersrand: https://libguides.wits.ac.za/c.php?g=693518&p=4914913

contributors, W. (2024, February 6). *Software requirements specification*. Retrieved 8 February, 2024, from Wikipedia: https://en.wikipedia.org/wiki/Software\_requirements\_specification

# Appendices