Shape

Description automatically generatedFaculty of Engineering and Informatics

School of Electrical Engineering and

Computer Science

Effective machine learning algorithms to detect DDoS attacks

A Cyber Security Project

Supervisor: Assistant Prof. Ibrahim Ghafir

Student: Harjeevan Gill

UB: 19021207

A thesis submitted in part fulfilment of the degree of MSc Cyber Security 2023

**Declaration:**

Name: Jeevan Gill

Date: 22/09/23

Table of Contents

[Chapter 1: Introduction 7](#_Toc146201140)

[1.1 Aims and Objectives 7](#_Toc146201141)

[1.2 Project Outline 8](#_Toc146201142)

[Chapter 2: Literature review 9](#_Toc146201143)

[2.1 DDoS Attack Background 9](#_Toc146201144)

[2.2 How DDoS Attacks Work 10](#_Toc146201145)

[2.3 Ethical Concerns and Motives 11](#_Toc146201146)

[2.4 Impact of DDoS attacks 11](#_Toc146201147)

[2.5 Historical attacks 12](#_Toc146201148)

[2.6 Open Systems Interconnection (OSI) layer 14](#_Toc146201149)

[2.7 Types of DDoS attacks 14](#_Toc146201150)

[2.7.1 Indirect DDoS Attack 14](#_Toc146201151)

[2.7.2 Direct DDoS Attack 15](#_Toc146201152)

[2.7.3 Common DDoS attacks 15](#_Toc146201153)

[2.8 Reflection-Based DDoS Attacks 15](#_Toc146201154)

[2.9 Exploitation-Based DDoS Attacks 19](#_Toc146201155)

[2.10 Triad of Weakness: OS, Network, and Process Vulnerabilities 19](#_Toc146201156)

[2.11 Attack methods 21](#_Toc146201157)

[2.12 Solutions to detect attacks in real time 22](#_Toc146201158)

[2.13 Current solutions for detecting DDoS attacks 23](#_Toc146201159)

[2.14 Related Work 26](#_Toc146201160)

[2.15 Classification of Machine learning Algorithms 28](#_Toc146201161)

[2.16 Tools review 30](#_Toc146201162)

[2.17 Reinforcement Learning Challenges for DDoS Attack Detection 30](#_Toc146201163)

[Chapter 3: Proposed Method and Dataset 32](#_Toc146201164)

[3.1 Random Forest 32](#_Toc146201165)

[3.2 K-Nearest Neighbours 33](#_Toc146201166)

[3.3 Neural Network 33](#_Toc146201167)

[3.4 Naïve Bayes Classifier 34](#_Toc146201168)

[3.5 Autoencoder 34](#_Toc146201169)

[3.6 CIC-DDoS2019 Dataset 35](#_Toc146201170)

[3.7 Activation functions 36](#_Toc146201171)

[Chapter 4 Implementation 38](#_Toc146201172)

[4.1 Process of Machine Learning 38](#_Toc146201173)

[4.2 Data Pre-Processing 38](#_Toc146201174)

[4.3 Random Forest 40](#_Toc146201175)

[4.4 Neural Network 41](#_Toc146201176)

[4.5 K-Nearest Neighbours 43](#_Toc146201177)

[4.6 Auto encoder and Random forest 44](#_Toc146201178)

[4.7 Naïve Bayes classifier 46](#_Toc146201179)

[Chapter 5: Evaluation and results 47](#_Toc146201180)

[This chapter presents the evaluation process and results for the machine learning DDoS detection models proposed and implemented in this study. 47](#_Toc146201181)

[5.1 Evaluation metrics 47](#_Toc146201182)

[5.2 ROC Curve comparison 49](#_Toc146201183)

[5.3 Performance metrics results 54](#_Toc146201184)

[Chapter 6 Conclusion and Future perspectives 56](#_Toc146201185)

[Appendix 57](#_Toc146201186)

[Bibliography 64](#_Toc146201187)

**List of Figures**

[*Figure 1: DDoS attack architecture (Aljuhani, 2021).* 10](#_Toc146201188)

[*Figure 2: DDoS attack efficacy (Aljuhani, 2021).* 12](#_Toc146201189)

[*Figure 3: Increasing magnitudes of DDoS attacks. (Aljuhani, 2021).* 13](#_Toc146201190)

[*Figure 4: OSI Layer Model.* 14](#_Toc146201191)

[Figure 5 Common DDoS attacks (Abidoye & Obagbuwa, 2018) (Aljuhani, 2021). 15](#_Toc146201192)

[*Figure 6: Types of DDoS Attacks (Gebremeskel et al., 2023).* 19](#_Toc146201193)

[Figure 7: Operational framework development model. 32](#_Toc146201194)

[*Figure 8: Datasets used* 36](#_Toc146201195)

[Figure 9: Split in packet attack types in three datasets. 36](#_Toc146201196)

[Figure 10: example of features with standard deviation zero 39](#_Toc146201197)

[Figure 11: Section of code to fill in missing data. 39](#_Toc146201198)

[Figure 12: RFFI and random forest classification. 40](#_Toc146201199)

[Figure 13: Ten features selected by the random forest feature importance. 40](#_Toc146201200)

[Figure 14: StandardScaler and one-hot encoding conversion. 41](#_Toc146201201)

[Figure 15: Layering the neural network. 42](#_Toc146201202)

[Figure 16: starting neural network. 43](#_Toc146201203)

[Figure 17: Creation of KNN classification. 43](#_Toc146201204)

[Figure 18: Scaling data for autoencoder. 44](#_Toc146201205)

[Figure 19: Setting up autoencoder. 45](#_Toc146201206)

[Figure 20: Autoencoder trained on scaled data 45](#_Toc146201207)

[Figure 21: Autoencoder encodes scaled data. 45](#_Toc146201208)

[Figure 22: Encoded data is inputted into the random forest. 46](#_Toc146201209)

[Figure 23: Creation of Naïve Bayes classifier 46](#_Toc146201210)

[Figure 24: Random Forest ROC curve 49](#_Toc146201211)

[Figure 25: ROC curve for KNN 50](#_Toc146201212)

[Figure 26: Neural network ROC curve 51](#_Toc146201213)

[Figure 27: ROC curve for autoencoder + random forest 52](#_Toc146201214)

[Figure 28: ROC curve for Naive Bayes classifier 53](#_Toc146201215)

[Figure 29: Classification performance metric results 54](#_Toc146201216)

[Figure 30: classification report for random forest classifier 57](#_Toc146201217)

[Figure 31: Confusion matrix for random forest classifier 57](#_Toc146201218)

[Figure 32: Classification report for K-nearest neighbours 58](#_Toc146201219)

[Figure 33: Confusion matrix for k-nearest neighbours 58](#_Toc146201220)

[Figure 34: Classification report for neural network classifier 59](#_Toc146201221)

[Figure 35: Confusion matric for neural network classifier 59](#_Toc146201222)

[Figure 36: classification report for autoencoder and random forest 60](#_Toc146201223)

[Figure 37: Confusion matrix for autoencoder and random forest 60](#_Toc146201224)

[Figure 38: Classification report for Naïve Bayes classifier 61](#_Toc146201225)

[Figure 39: Confusion matrix for Naïve Bayes classifier 61](#_Toc146201226)

[Figure 40: Bar chart comparing accuracy scores of all classifiers 62](#_Toc146201227)

[Figure 41: Bar chart comparing F1 scores of all classifiers 62](#_Toc146201228)

[Figure 42: Bar chart comparing ROC AUC scores of all classifiers 63](#_Toc146201229)

[Figure 43: Bar chart comparing precision of all classifiers 63](#_Toc146201230)

**Abstract**

The threat landscape is always changing resulting in the expansion of connected devices and the complexity of network infrastructures, necessitating increasingly effective defences against cyberattacks. Internet connected devices are easily accessible combined with the advancement of computing power and innovation it can lead to users not having to correct knowledge to protect their network and devices. DDoS attacks, have the potential to seriously interrupt network services and jeopardise the security and integrity of crucial systems thus affecting the availability.

In order to understand the potential effects of DDoS attacks on network infrastructures, the paper investigates their characteristics and behaviour’s; the suitability of different machine learning algorithms, including supervised and unsupervised learning methods, for DDoS detection. The Canadian Institute of Cyber Security's CICDDoS2019 dataset is used to test and train machine learning algorithms. Random forests, Nave Bayes classifier, K-nearest neighbours, neural network, and autoencoder with random forest are the machine learning classifiers implemented in TensorFlow and Scikit-Learn

# Chapter 1: Introduction

Distributed denial-of-service (DDoS) attacks pose a severe threat to organisations by overwhelming networks and infrastructure through malicious traffic from multiple sources. Detecting and mitigating these attacks quickly and accurately is critical to limit disruption and damages (Dong & Sarem, 2020). Traditional DDoS detection relies on rule-based systems, but these require extensive manual effort to define signatures and fail to adapt as attacks evolve.

Machine learning (ML) offers a powerful alternative by automatically recognising patterns and anomalies in traffic without explicit programming. Recent research has applied ML techniques across supervised, unsupervised, and semi-supervised paradigms for DDoS detection. Algorithms including neural networks, support vector machines, forest algorithms, and clustering have shown promising results (Aladaileh et al., 2022). However, limitations remain in model generalisation, explain ability, and operationalisation.

This paper presents a comprehensive methodology to evaluate and compare leading machine learning algorithms for DDoS detection using CIC-DDoS2019 dataset. Their performance will be analysed through metrics such as accuracy, precision, recall, F1-score, and ROC AUC. The goals are to benchmark ML classifiers on a common dataset, gain insights into their detection capabilities, and determine the most effective approach for real-time deployment.

## Aims and Objectives

* Research existing methods used for DDoS detection.
* Review similar journals using machine learning to detect DDoS attacks.
* Acquire a trustworthy dataset with legitimate network traffic, filter the data so there is no missing values and classification for legitimate and DDoS traffic.
* Develop multiple machine learning-based systems for the detection of DDoS attacks.
* Produce visual and statistical analysis of the outcomes of the machine learning algorithms and document findings.
* Evaluate and compare the effectiveness of different machine learning algorithms in detecting DDoS attacks.

## 1.2 Project Outline

Chapter two will present an overview of relevant prior research in the DDoS detection methods. It will summarise existing methods like honeypots, machine learning techniques, vulnerabilities, type of DDoS attacks.

Chapter three will explain the set of classification algorithms that will be implemented and compared in this study, including random forests, k-nearest neighbours, neural networks, autoencoders, and Naive Bayes.

Chapter four will discuss the data pre-processing and feature engineering process. It will provide an overview of the software tools and environments used to implement the algorithms. A step-by-step explanation of how the classifiers work will be given.

Chapter five will present the results of implementing the classifiers on the selected dataset. The performance of each algorithm will be analysed quantitatively using the evaluation metrics defined. A comparative analysis will identify the most effective classifier for the given problem.

To conclude, chapter six will summarise the key findings and conclusions of the study. Any limitations of the current work will be acknowledged. Recommendations for future research directions in this field will be provided. The chapter will close with a discussion of potential real-world applications and the broader impact of the research.

# Chapter 2: Literature review

This chapter discusses the research on applying machine learning to detect DDoS attacks as well as current methods for doing so. It gives background information on the issue and an overview of DDoS attack strategies. The research then concentrates on classification of machine learning techniques that possibly could be used DDoS detection. The key instruments and frameworks are outlined.

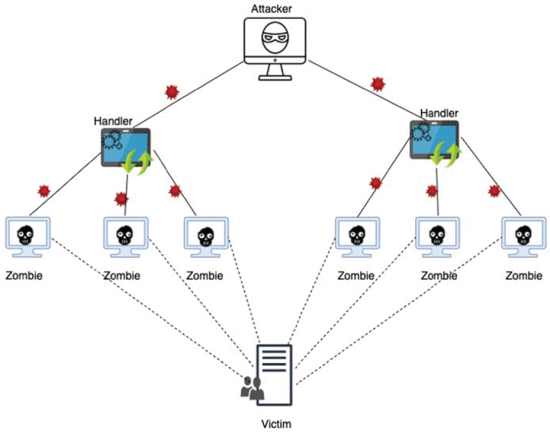
## 2.1 DDoS Attack Background

Unprecedented connectedness and opportunity have been sparked by the constant growth of technology, revolutionising many aspects of modern civilisation. But this development has also spawned sophisticated cyber dangers that prey on holes in network systems. The capacity of distributed denial of service (DDoS) attacks to impair internet services and disrupt crucial activities has made them one of the most concerning of these threats. DDoS attacks can disable network services by continuously flooding the servers' resources with unwanted traffic, which not only denies legitimate users but slows down or potentially prevents the server from working as should (Ujjan et al., 2020). DDoS attacks have detrimental consequences experienced across industries.

DDoS attacks entail an orchestrated attack on targeted systems, rendering them inaccessible to legitimate users by overwhelming network resources, such as bandwidth, processing power, or application layer services (Sharafaldin et al., 2019). DDoS attacks are led by a botmaster who has control of compromised computers or Internet of things (IoT) referred as a botnet. Attacks usually have multiple stages reconnaissance, exploitation, command and control (C&C) setup, and execution of the DDoS attack.

Organisations in a variety of industries, including the government, financial institutions, e-commerce platforms, and internet service providers, are constantly at risk from DDoS attacks (Ujjan et al., 2020). Detecting these kinds of threats in real time with larger network premises has become an absolute necessity. These attacks can have a wide range of motivations, including ideological activism, financial gain, competitive advantage, or malicious enjoyment. The wide spectrum of targets is evidence of the DDoS attacks' extensive influence and the inherent vulnerability of contemporary digital infrastructures.

DDoS attacks are motivated by a number of variables and may be used by hacktivist organisations to propagate their message, condemn perceived injustices, or hinder the activity of their competitors (Gupta et al., 2020). These attacks may be used by criminal actors to extort and blackmail their targets, or as a distraction tool to permit other unlawful activities. DDoS attacks are frequently used as a method to obtain an advantage in situations where rival enterprises must compete for customers' trust and online visibility (Gupta et al., 2020).



*Figure 1: DDoS attack architecture (Aljuhani, 2021).*

## 2.2 How DDoS Attacks Work

DDoS attacks are carried out by leveraging networks of malware-infected internet-connected devices. Botnets are compromised device networks that allow attackers to remotely manipulate the devices. Individually compromised machines are known as bots or zombies (Azeez et al., 2021).

To carry out a DDoS attack, the attacker transmits commands to a network of infected bots under their command. When the botnet is told to attack a certain server or network, each bot delivers a flood of requests to the target's IP address (Lazar et al., 2021). This rush of malicious traffic from thousands of bots has the ability to overload the target, resulting in denial of service for genuine users.

DDoS attackers can marshal significantly more firepower against a target by leveraging networks of hacked devices than a single computer or device (Lazar et al., 2021). DDoS attacks get their name from their distributed nature, which makes them a dangerous cyber threat.

## 2.3 Ethical Concerns and Motives

DDoS attacks raise a number of ethical considerations and motivations that must be thoroughly investigated. To begin, it is critical to recognise their illegality, as they violate the laws of the majority of states. DDoS attack perpetrators face criminal accusations and punishments, firmly demonstrating their unethical and illegal nature. The Computer Misuse Act 1990 (CMA) is a piece of legislation in the United Kingdom that criminalizes unauthorised access to computer systems and data (Home Office, 1990).

One common motivation for DDoS attacks is to cause harm to businesses or organisations. Financial gain, personal vendettas, or competitive advantage are all possible motivations, which has the potential to cause major disruption to essential infrastructure (Ling et al., 2023). The ethical condemnation of bringing harm to innocent beings for personal or financial gain, on the other hand, is unmistakable.

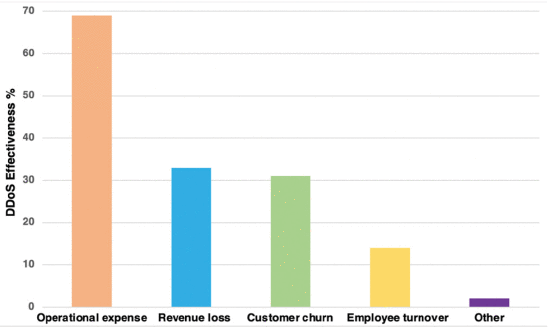
Furthermore, some DDoS attacks are politically or ideologically motivated, aimed at stifling specific ideas by targeting websites or online services. This practise raises ethical questions about free speech and censorship because it restricts access to knowledge and public dialogue.

While certain individuals argue that DDoS attacks are permissible for security testing, it is critical to emphasise that such endeavours must carefully conform to the principles of explicit consent and informed consent (Mamolar et Al,. 2018).

## 2.4 Impact of DDoS attacks

* DDoS attacks can wreak havoc across the digital landscape, negatively affecting organisations, individuals, and the broader internet infrastructure. Some of the major ramifications of DDoS attacks include:
* Disruption of business operations and revenue loss for enterprises as a rush of attack traffic knocks their websites and online services offline; organisations of all sizes are vulnerable (Srinivasan et al., 2019).
* Reputational harm if a company's website or online services are inaccessible to clients for an extended period of time. Customer trust might be tough to re-establish (Srinivasan et al., 2019).
* Reduced productivity and workflow for employees whose occupations rely on internet-connected devices, networks, and websites. Work grinds to a standstill when staff are unable to access necessary web tools (Srinivasan et al., 2019).
* Leaked sensitive data is a risk if a DDoS overwhelms a website's servers to the point of failure, exposing private information (Srinivasan et al., 2019).
* Negative consequences on unrelated websites and internet infrastructure that share network resources with the directly targeted site (Aljuhani, 2021).
* The financial costs of mitigating and recovering from attacks in terms of technological solutions, security team time, and business loss; expenses can quickly add up (Srinivasan et al., 2019).

Figure 2 illustrates the effectiveness of DDoS attacks, which are known to raise operational costs by 69%, revenue by 33%, customer attrition by 31%, and employee turnover by 14% (Balarezo et al., 2022).



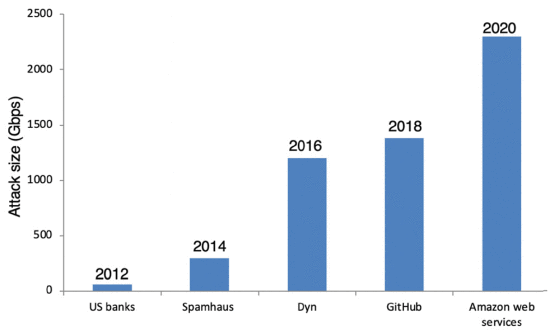
*Figure 2: DDoS attack efficacy (Aljuhani, 2021).*

## 2.5 Historical attacks

GitHub was subjected to a DDoS attack in 2018, which was one of the most significant reported cases. During this incident, the platform's servers experienced a flood of data at an astounding rate of 1.4 terabits per second (Tbps), or 126.9 million packets per second. The attackers used a technique known as "memcaching," which entailed sending fake requests to susceptible servers by forging the victim's Internet Protocol (IP) address. As a result, the impacted servers were overwhelmed with amplified responses. Only one week after this incident, Arbour Networks announced the effective mitigation of a comparable attack by a US service provider, which had reached an onerous threshold of approximately 1.7 Tbps (Adefemi Alimi et al., 2022).

The memcached protocol, which is based on the User Datagram Protocol (UDP), has an estimated amplification factor of around 51,000 (Adefemi Alimi et al., 2022). This amplification factor indicates that the length of the generated answer is 51,000 times longer than the length of the original request.

In a different historical occurrence that happened in 2016, a sizable DDoS attack was organised with the help of the cutting-edge Mirai botnet. The target of this attack was Dyn, a well-known American company in charge of a large section of the Internet's domain name system (DNS) infrastructure. In this case, the attackers used Internet of Things (IoT) equipment, such as digital cameras, as tools to disrupt many websites across Europe and the United States, including Twitter, Netflix, CNN, and many others. This devastating attack had an estimated size of 1.2 Tbps and was spread across a network of over 100,000 infected nodes (Adefemi Alimi et al., 2022).



*Figure 3: Increasing magnitudes of DDoS attacks. (Aljuhani, 2021).*

## 2.6 Open Systems Interconnection (OSI) layer

|  |  |
| --- | --- |
| 7 | Physical |
| 6 | Data Link |
| 5 | Network |
| 4 | Transport |
| 3 | Session |
| 2 | Presentation |
| 1 | Application |

*Figure 4: OSI Layer Model.*

* Layer 7 - Application Layer: Provides network services to applications. Includes protocols like Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File transfer protocol (FTP).
* Layer 6 - Presentation Layer: Translates data into a standard format for the application layer. Handles data compression, encryption, etc.
* Layer 5 - Session Layer: Manages sessions and connections between applications.
* Layer 4 - Transport Layer: Transfers data using transmission protocols like transmission control protocol (TCP) and user datagram protocol (UDP).
* Layer 3 - Network Layer: Routes packets across networks. The IP protocol operates here and handles logical addressing.
* Layer 2 - Data Link Layer: Transfers data frames between network nodes; detects and corrects errors.
* Layer 1 - Physical Layer: Transmits raw bit streams over physical connections like cable and fibre. Deals with voltages, frequencies, physical connections

## 2.7 Types of DDoS attacks

DDoS attack's main strategy is to use flaws in certain network protocols or services to reroute and magnify the attack flow. In order to carry out the attack, the attacker identifies exploits systems that have greater bandwidth or processing power than their own (Sharafaldin et al., 2019).

## 2.7.1 Indirect DDoS Attack

Indirect DDoS attacks involves the attack being launched from several sources rather than directly from the attacker's own network or devices, overwhelming the target server or network with a torrent of malicious traffic. Amplification or reflection attacks are other names for indirect DDoS attacks (Li et al., 2021).

## 2.7.2 Direct DDoS Attack

Direct DDoS attack entail the attacker gaining direct management and control of the target server. By changing the server resource configuration (e.g., CPU), the target server is unable to deliver appropriate computing resources for the services within a specified time frame (Li et al., 2021).

## 2.7.3 Common DDoS attacks

| HTTP flood | Application (Layer 7) |
| --- | --- |
| DNS amplification | Application (Layer 7) |
| SSL DDoS | Application (Layer 7) |
| Slowloris | Application (Layer 7) |
| Session Initiation protocol | Application (Layer 7) |
| Memcached reflection | Application (Layer 7) |
| Botnet attack | Varies, often Application (Layer 7) |
| SYN flood | Transport (Layer 4) |
| UDP flood | Transport (Layer 4) |
| ACK flood | Transport (Layer 4) |
| ICMP flood | Network (Layer 3) |
| Smurf attack | Network (Layer 3) |
| Ping of Death | Network (Layer 3) |

Figure 5 Common DDoS attacks (Abidoye & Obagbuwa, 2018) (Aljuhani, 2021).

Application layer (Layer 7) attacks target web servers, applications, and services directly.

Transport layer (Layer 4) attacks overwhelm network sessions and connectivity.

Network layer (Layer 3) attacks exploit vulnerabilities in network routing and packet handling.

## 2.8 Reflection-Based DDoS Attacks

***2.8.1* Portmap (PM):**

Several security weaknesses in older versions make the portmap service vulnerable. Most Unix-like operating systems support Portmap, which transforms RPC programme numbers into IP address and port pairs (Marvi & Arfeen, 2020).

Portmap usually uses the UDP and TCP protocols on port 111. Prior implementations, however, had flaws that might have let remote attackers run arbitrary code or resulted in a denial of service (Marvi & Arfeen, 2020). It's important to note that victims are not liable for preventing such attacks, as server owners must correctly configure their systems in order to close vulnerabilities according to National Institute of Standards and Technology (NIST) (Pranggono., (2015).

A common exploit makes use of an out dated portmap daemon's buffer overflow vulnerability. An attacker might cause a buffer overflow and execute malicious code with the authority of the portmap process by creating a malicious RPC request and sending it to port 111 (Marvi & Arfeen, 2020).

***2.9.2 Lightweight Directory Access Protocol (LDAP):***

Lightweight Directory Access Protocol (LDAP) is a commonly used application protocol for querying and changing directory service data stored in repositories such as Microsoft Active Directory (AD). Users authenticate to LDAP servers to perform lookup and update activities on the directory database. LDAP systems are commonly targeted by distributed denial-of-service (DDoS) attacks due to their inherent client-server design (Naiem et al., 2023).

An LDAP amplification attack uses botnets of compromised hosts to send LDAP lookup requests to a large number of LDAP servers while spoofing the target's IP address. The servers respond to the target system using LDAP payloads that are far larger than the initial requests as a result of the forged IP, creating an amplification effect. The User Datagram Protocol (UDP) has been abused by attackers to enable attacks above 1 Tbps with larger amplification factors, sometimes surpassing 50x (Marvi & Arfeen, 2020). TCP-based LDAP amplification has also been noticed as a DDoS method, while being less effective than UDP.

Attackers have demonstrated the capacity to overwhelm LDAP servers even in the absence of amplification effects through volumetric request flooding attacks that consumes memory, CPU, and network resources (Marvi & Arfeen, 2020). . These attacks demonstrate how vulnerable the LDAP infrastructure is to denial-of-service strategies based on resource exhaustion (Aljuhani, 2021).

***2.8.3 Microsoft SQL (MSSQL):***

Microsoft SQL Server (MSSQL) is a proprietary relational database management system (DBMS) that operates on client-server architecture. The MSSQL protocol enables communication between MSSQL client programmes and database servers for tasks including querying, updating, and maintaining database contents. However, MSSQL packets and queries have the potential for amplification, attackers frequently target exposed MSSQL servers for DDoS attacks.

In MSSQL DDoS attacks, attackers send high volumes of specific SQL requests to victims to amplify traffic. For instance, attackers are able to create and send REVERSE\_LOGIN packets with payloads far larger than the initial requests (Marvi & Arfeen, 2020). Additionally, layered SQL searches enable a single request to repeatedly start processes that amplify replies. Attackers also overwhelm servers with expensive queries that use an excessive amount of CPU resources in addition to amplifying their attacks, leading to significant infrastructure risk. The vulnerability of DBMS servers to resource exhaustion and traffic amplification techniques are highlighted by MSSQL DDoS attacks (Marvi & Arfeen, 2020).

***2.8.4 Network type protocol (NTP):***

Network Time technology (NTP) allows devices to synchronise clocks across networks and the Internet by contacting time servers. However, the User Datagram Protocol (UDP) based NTP protocol has flaws that can be used to launch DDoS attacks that leverage amplification (Jeitner et al., 2020).

NTP amplification attacks occurs when adversaries transmit tiny fake UDP packets using the victim's IP address as the source to publicly accessible NTP servers (Jeitner et al., 2020). The NTP servers’ response is to send a considerably larger UDP packets to the spoofed IP address in accordance with the protocol requirements, leading to an amplified the signal. Attackers can overwhelm victim networks by achieving amplification factors greater than 500x with enough traffic sent from numerous NTP servers (Marvi & Arfeen, 2020). NTP servers running on out-of-date, unpatched software are particularly vulnerable to exploitation, allowing DDoS amplification using system resources.

***2.8.5 Domain Name System (DNS)***

The mapping of domain names to the underlying IP addresses necessary to transport traffic on the Internet and within organisations is made easier by the DNS. For the purpose of answering inquiries and offering resolutions, it uses a hierarchically distributed database of DNS servers (Hesselman et al., 2020). However, to carry out amplification-based DDoS attacks, thieves take use of holes in DNS protocols.

DNS amplification, adversaries fake victims' IP addresses and send recursive DNS requests to publicly available DNS resolvers. The DNS servers then route queries to additional DNS servers as a result of request recursion, resulting in bloated answers pointed at the spoof IP address. Attackers can overwhelm target networks by amplifying their responses by more than 50 times with enough traffic volume mirrored from DNS infrastructure (Marvi & Arfeen, 2020).

***2.8.6 Trivial file transfer protocol (TFTP):***

Trivial File Transfer Protocol (TFTP) is used for transferring files over a network. TFTP uses port 69 and the User Datagram Protocol (UDP) protocol thus it does not provide error checking or retransmission capabilities, making the protocol prone to data corruption issues (Marvi & Arfeen, 2020). Since only the basic upload and download file transfer operations may be performed, it does not require user authentication or directory browsing skills, resulting in it being quicker and cheaper than other, more robust protocols like FTP.

A lockstep method is used by TFTP to transport data allowing a request to upload or download a file to be made by the client. The server replies with the first data packet of the file, before sending the next packet, the client must acknowledge receipt of the first one (Marvi & Arfeen, 2020). This request-response cycle keeps going until the whole file has successfully transferred. This ACK-based procedure is also vulnerable to DDoS amplification strategies.

Due to its ease of use, TFTP is primarily utilised within networking devices and appliances for automatic transfers of configuration files or firmware images. But the absence of security mechanisms also leaves exposed TFTP servers open to DDoS amplification attacks (Akanji et al 2021).

TFTP amplification attacks occur when attackers use botnets to fake the victim's IP address and send TFTP Read Request packets to TFTP servers that are at risk (Akanji et al 2021). Depending on the TFTP specifications, the servers react to the spoof IP address with data packets that are substantially larger than the initial requests. Attackers have exploited flaws in TFTP based on UDP to reach amplification factors of over 70x (Marvi & Arfeen, 2020). This attack shows the bandwidth exhaustion flaws that unsecured TFTP servers have unrestricted access.

***2.8.7 Simple service discovery protocol (SSDP):***

The Simple Service Discovery Protocol (SSDP) gives network devices and applications the ability to find services over IPv4 and IPv6 networks. Multicast messaging is used by SSDP, which runs over UDP, to provide smooth networked device and capability identification (Marvi & Arfeen, 2020). SSDP is susceptible to DDoS amplification vulnerabilities due to its reflecting nature.

Attackers use spoofed multicast SSDP M-SEARCH query packets to send SSDP amplification requests to adjacent network gateways and devices. These multicast packets are reflected to all SSDP nodes connected as per the protocol's design, which causes bloated answers to overwhelm the spoof victim IP address. Attackers have used UDP reflection capabilities to achieve SSDP amplification of more than 30x (Marvi & Arfeen, 2020).

***2.8.8 Network-based input/output system (NetBIOS):***

The Network Basic Input/Output System (NetBIOS), which enables network naming, session management and data transfer capabilities, offers services for communication between programmes and network hardware. Applications like Windows file sharing, which rely on NetBIOS name services (NBTNS) for host naming and discovery, incorporate NetBIOS, which operates over TCP and UDP (Marvi & Arfeen, 2020). However, DDoS attacks can target insecure NetBIOS server implementations.

NetBIOS amplification attacks, adversaries impersonate victims' IP addresses to deliver tiny UDP packets to exposed NetBIOS name servers. In response, the weak servers amplify bloated UDP payloads sent to the spoof addresses by a factor of more than 10x (Marvi & Arfeen, 2020). Such requests can be sent by botnets to numerous servers, enlarging their victims' networks.

***2.8.9 Simple network management protocol (SNMP):***

TCP/IP networks, the Simple Network Management Protocol (SNMP) offers functionality for managing and monitoring networked devices. SNMP provides data gathering from network nodes using get/set instructions sent from SNMP management systems over UDP. Attackers use the SNMPv1/v2 UDP-based protocols, though, to launch DDoS amplification attacks (Marvi & Arfeen, 2020).

SNMP amplification, adversaries send several get/get-bulk requests to vulnerable SNMP devices by spoofing the victim's IP address. With amplification factors surpassing 14x, the devices respond with bloated SNMP trap and inform messages that are aimed at the fake IP (Marvi & Arfeen, 2020).

## 2.9 Exploitation-Based DDoS Attacks

DDoS attacks based on exploitation overwhelm targets by taking advantage of flaws in software or protocols. Typical exploitation methods include:

* **2.9.1 SYN flood** - The attacker bombards a server with SYN requests, which it answers and leaves open while it waits for a response from the client. The server resources are used up by the attacker, who never answers, until the connections time out (Al-Arnaout et al., 2022).
* **2.9.2 UDP flood** – The attacker bombards the victim server with arbitrary UDP packets on open ports. The server responds to each packet with an ICMP destination unreachable packet. Both server resources and network bandwidth are used in this (Bijalwan et al. 2015).

A diagram of a system

Description automatically generated

*Figure 6: Types of DDoS Attacks (Gebremeskel et al., 2023).*

## 2.10 Triad of Weakness: OS, Network, and Process Vulnerabilities

This section describes vulnerabilities in operating systems, networks, and procedures that potentially allow DDoS assaults to occur. It addresses OS issues such as resource management and TCP/IP faults that could cause instability and service disruption. The details of network vulnerabilities such as BGP routing, VLAN, and ICMP flooding flaws that potentially sabotage connectivity are provided. Finally, there are weaknesses in the process that could allow for excessive resource utilisation via attacks.

***2.10.1 OS Vulnerabilities:***

* DDoS attacks can be launched by taking advantage of flaws in an operating system's implementation of the TCP/IP stack. Through methods like SYN floods or TCP fragmentation attacks, these vulnerabilities could give attackers the ability to control network connections, send erroneous packets, or deplete system resources (Softić & Vejzović., (2022).
* There are vulnerabilities in the operating system's core component, which manages system resources and operations. Exploiting these weaknesses can cause resource depletion or instability, which would disrupt services or cause system failures (Softić & Vejzović., (2022).
* • Memory Management flaws can be used to rapidly deplete system memory. Attackers can cause a denial of service by incorrectly allocating and releasing memory, as well as draining available memory resources (Softić & Vejzović., (2022).
* DDoS attacks can take advantage of flaws in the way an operating system manages processes. For instance, it is possible to overtax system resources and interrupt services by taking advantage of weaknesses in the procedures for creating, scheduling, or terminating processes (Softić & Vejzović., (2022).
* Implementation of an operating system implementation may have flaws that can be used to disrupt services. For instance, attackers may take advantage of flaws to saturate the file system with multiple files, using up all available disc space or taxing the file system's processing power (Softić & Vejzović., (2022).
* DDoS attacks can potentially take advantage of vulnerable operating system functions or protocols. Resource depletion crashes, or other types of interruption may result from taking advantage of flaws in these services or protocols (Softić & Vejzović., (2022).

***2.10.2 Network Vulnerabilities:***

* BGP routing's vulnerabilities allow attackers the ability to modify routing data and reroute or interfere with network traffic (Tran et al., 2020).
* Attackers may be able to switch between VLANs and launch DDoS attacks against the network by taking advantage of weaknesses in VLAN setups (Hai et al., 2021).
* Attackers may be able to sabotage network connectivity, reroute traffic, or overburden network devices by taking advantage of flaws in network switches or routers (Hai et al., 2021).
* ICMP floods use ICMP Echo Request (ping) packets, attackers flood the network with mass amounts of network requests to use up network resources and possibly triggering network congestion (Bouyeddou et al., 2020).

***2.10.3 Process vulnerabilities:***

* Vulnerabilities that can result in the exhaustion of resources like CPU, RAM, or disc space can be exploited to cause service interruptions or slowdowns (Suleman, 2021).
* Attackers can take advantage of flaws in the creation or management of processes, attackers might launch a "fork bomb" attack that causes a sudden influx of excessively many processes, taxing the system's resources (Piergiorgio et al., 2022).
* Attackers may drain system resources and impair system performance by taking advantage of flaws in the process handling mechanisms, such as a lack of process limits or faulty input validation (Suleman, 2021).

## 2.11 Attack methods

***2.11.1 Portmap:***

Using a fake source IP, an attacker creates a port mapper query. The target then receives answers whose amplification factor depends on the query. This attack uses the transport protocol UDP/TCP, port number 111 and has an amplification factor 7 to 28 in bandwidth (Marvi & Arfeen, 2020). The ratio between the response and request is known as the amplification factor.

***2.11.2 LDAP:***

Using a fake source IP, an attacker creates an LDAP server request, and the victim receives boosted answers. This attack uses the transport protocol UDP/TCP, port number 389 and has an amplification factor 55 to 70 in bandwidth (Marvi & Arfeen, 2020).. The ratio between the response and request is known as the amplification factor.

***2.11.3 MSSQL:***

Attackers utilise SQL servers by faking the originating IP and sending automated requests. Due to the nature of the SQL servers the victim will receive responses that are magnified depending on the number of instances present in the servers. This attack uses the transport protocol UDP/TCP, port number 1434 and has an amplification factor 25 in bandwidth (Marvi & Arfeen, 2020).

***2.11.4 NTP:***

An attacker sends a command to an NTP server using the victim's fake source IP, and the server responds with data on the 600 most recent connections. This attack uses the transport protocol UDP/TCP, port number 123 and has an amplification factor 560 in bandwidth (Marvi & Arfeen, 2020).

***2.11.5 DNS:***

An attacker accesses public DNS servers for a DNS name lookup by using the victim's fake source IP. Depending on the choice made in the attacker's created query, the victim receives an amplified response in return. This attack uses the transport protocol UDP/TCP, port number 53 and has an amplification factor 28 to 54 in bandwidth (Marvi & Arfeen, 2020).

***2.11.6 TFTP:***

The attacker uses a forges IP address to generate a read request (RRQ) for a file on TFTO server. This generates an amplified response including retransmission and error codes that the victim receives. This attack uses the transport protocol UDP/TCP, port number 69 and has an amplification factor 60 in bandwidth. This attack uses the transport protocol UDP/TCP, port number 69 and has an amplification factor 60 in bandwidth (Marvi & Arfeen, 2020).

***2.11.6 SSDP:***

A SOAP request is sent by the attacker to open a UPnP device on the Internet after faking the source IP of the victim. The apparatus then communicates with the victim in an enhanced manner as a result. This attack uses the transport protocol UDP/TCP, port number 1900 and has an amplification factor 30 in bandwidth (Marvi & Arfeen, 2020).

***2.11.7 NetBIOS:***

An attacker makes a broadcast query to find out the names of other computers in the network by using a spoofed IP of a victim. This causes machines or other networked devices to respond to the victim with amplified reactions. This attack uses the transport protocol UDP/TCP, port number 137 and has an amplification factor 2 to 4 in bandwidth (Marvi & Arfeen, 2020).

***2.11.8 SNMP:***

An attacker sends a request to a local area network device that is vulnerable after faking the Source IP. The victim then receives an amplified reaction from the device. This attack uses the transport protocol UDP/TCP, port number 161 and has an amplification factor 3 to 7 in bandwidth (Marvi & Arfeen, 2020).

***2.11.9 SYN flood:***

The client deliberately never replies to the server's SYN-ACK signals while sending an excessive amount of SYN queries. Thus, the attacker maintains the connection active at the target server until the time-out. This attack uses the transport protocol TCP, using any port number and the amplification factor depends on the number of open connections (Marvi & Arfeen, 2020).

***2.11.10 UDP flood:***

Random ports on the host are targeted by the attacker, who then floods them with IP packets containing User Datagram Protocol (UDP) packets. The attacker searches for applications connected to these datagrams to which allows the attack to take place. This attack uses the transport protocol UDP, using any port number over 1024 and the amplification factor depends on the number of requests (Marvi & Arfeen, 2020).

## 2.12 Solutions to detect attacks in real time

As DDoS attacks grow more frequent and complex, organisations require robust detection capabilities to activate timely and effective mitigation. DDoS attacks now leverage IoT botnets and multi-vector techniques to overwhelm targets in minutes with terabits per second of malicious traffic. To minimise business disruption and prevent costly outages, DDoS detection systems must provide the speed, accuracy and precision needed to initiate surgical mitigation responses (Dong & Sarem, 2020). This overview examines the key success criteria for DDoS detection in enabling prompt, targeted attack mitigation with limited impact to legitimate user traffic. Achieving rapid threat visibility and surgical mitigation requires a combination of automated analytics, signature-based systems, machine learning and integrated response processes (Ulemale., 2022).

***2.12.1 Speed of Detection***

Rapid detection is critical to activate mitigation before an attack can inflict significant damage. DDoS attacks can ramp up to peak traffic volumes within minutes, so detection latencies need to be in the sub-minute range. Capabilities like automatic threshold-based alerting offer detection speeds much faster than manual monitoring. Tracking rate-of-change on traffic metrics can identify sudden spikes indicative of DDoS activity. High-speed statistical analysis techniques like adaptive cumulative sum (CUSUM) algorithms can detect anomalies in near real-time (Nguyen et al., 2023). Machine learning models trained to differentiate DDoS patterns from benign traffic spikes can accelerate detections.

Tight integration is required between DDoS detection systems and mitigation platforms to activate policy-based traffic filtering within seconds of an alert. Large network operators may need to distribute detection capabilities to critical network aggregation points to minimise any transfer delay in alert data (Katuk, 2023). The goal is to never allow large-scale DDoS attacks to ramp up unimpeded at their target destination.

***2.12.2 Accuracy of Detection***

Imprecise detection with high false positive rates can be just as detrimental as slow detection by blocking legitimate traffic incorrectly (Gebremeskel et al., 2023). Positive identification of real DDoS attacks is vital to avoid unnecessary business disruption. Correlating DDoS indicators across multiple analytic systems such as behavioural analysis, signature detection, protocol analysis, reputation databases, and edge router monitoring enables high-confidence alerting (Halladay et al., 2022). Machine learning-based traffic profiling adds precision by learning characteristics of normal vs. anomalous flows for the environment.

***2.12.3 Precision of Detection***

Precision detection capabilities allow targeted mitigation measures like rate limiting specific attack traffic while minimising collateral damage. Detailed alert data on the attributes of suspected DDoS traffic enables the crafting of very selective filtering policies. Multi-stage scrubbing techniques can isolate and progressively block only traffic confirmed as malicious (Gebremeskel et al., 2023). The goal is to maintain normal operations and user experience as much as possible during attacks.

## 2.13 Current solutions for detecting DDoS attacks

***2.13.1 Network Behaviour Analysis:***

Network behaviour analysis (NBA) establishes statistical baseline profiles for normal traffic volumes and patterns based on historical data. By continuously analysing flow data from critical aggregation points, NBA can detect significant anomalies or deviations in metrics like bandwidth utilisation, flows per second, connections, packets per second, etc. (Bougueroua et al., 2021). Adaptive machine learning algorithms enable automated fine-tuning of detection sensitivity and baseline thresholds to limit false positives. DDoS detection is enhanced by correlating multiple anomaly events across metrics and inspection points. However, NBA provides no insight into the specific attributes or root causes of anomalies (Bougueroua et al., 2021).

***2.13.2 Signature-based Detection:***

Signature detection relies on deep packet inspection (DPI) to match packet and flow attributes against constantly updated threat intelligence detailing known DDoS malware variants, botnet addresses, and attack tool traffic profiles (Praseed & Thilagam, 2022). Signatures contain IP addresses, header values, payload snippets, or traffic behaviour patterns allowing accurate identification of common DDoS types (Praseed & Thilagam, 2022). Large signature sets require highly optimised pattern matching capabilities to avoid throughput and latency impacts. DPI is less effective against zero-day DDoS attacks with no known signature.

***2.13.3 Edge Traffic Monitoring:***

Edge router and firewall monitoring analyses traffic volumes and flows at network ingress and egress points to identify volume-based DDoS attacks. Tracking metrics like packets per second, connections, and bandwidth consumption per destination IP or subnet provides visibility into network flooding attempts (Efe & Abaci., 2022). Smart rate limiting can automatically contain suspect traffic entering the network. However, edge monitoring alone lacks context into the specific sources and nature of anomalies.

***2.13.4 Intrusion Detection/Prevention Systems:***

Network-based intrusion detection/prevention systems (IDS/IPS) analyse traffic patterns to detect anomalies, policy violations, and known attack signatures. Detected threats can be blocked in real-time via inline prevention capabilities. IDS/IPS often integrates denial-of-service (DOS) and DDOS attack detection based on protocol anomalies, traffic thresholds, and rate-based signatures (Efe & Abaci., 2022). For example, a TCP SYN flood would be identified by an extremely high rate of half-open TCP sessions (Efe & Abaci., 2022). IPS systems can terminate half-open connections from offending source IPs to mitigate SYN floods and other protocol DDoS attacks.

***2.13.5 Application Firewalls:***

Web application firewalls (WAFs) establish baseline profiles for normal application traffic across metrics like requests per second, connections, errors, payload sizes, IPs per user, etc (Dawadi et al., B., 2023). Any anomalous deviations that may indicate DDoS or other web application attacks can be blocked in real-time. WAF rules specifically target layer 7 DDoS attacks that aim to overwhelm web servers and infrastructure through application requests vs network floods (Dawadi et al., B., 2023). However, WAFs are limited to the context of a single application vs network-wide visibility.

***2.13.6 Cloud Mitigation Services:***

Cloud-based scrubbing services absorb attack traffic on behalf of the target organisation, filter out the malicious traffic, and forward only clean traffic to the end destination (El-Sofany., 2020). Routing to cloud scrubbers helps avoid denial-of-service on the target's local internet connections. Large cloud providers can absorb and scrub massive attack volumes utilising DDoS detection and mitigation technologies at scale (El-Sofany., 2020). However, rerouting traffic introduces latency and relies on third-party systems. Hybrid models maintain local scrubbing with overflow to the cloud during large attacks.

***2.13.7 Honeypots:***

Honeypots are decoy systems and assets deployed to attract and detect malicious traffic. Since honeypots have no authorised activity, all interactions are considered suspicious. DDoS honeypots can emulate services and vulnerabilities to attract botnet traffic, providing visibility on botnet IP addresses, malware signatures, and attack patterns (Kopp et al., 2021). This data feeds detection and blocking of botnet DDoS attacks. However, honeypots only capture a sample of botnet activity and do not confirm if an attack is underway (Kopp et al., 2021).

***2.13.8 Baseline Profiling:***

Baseline profiling establishes statistical models for normal traffic volumes and patterns over time (Bii et al., 2021). Metrics like bandwidth utilisation, concurrent connections, requests per second, packets per flow are measured across systems and network segments. Significant deviations from baselines can indicate anomalies warranting investigation. However, general baselining lacks the precision of behaviour-based models that account for fluctuations in usage patterns and seasonal trends. Baselining is most effective when integrated with other detection methods (Bii et al., 2021).

***2.13.9 Machine Learning Models:***

Machine learning models can be trained on datasets of normal and DDoS attack traffic to automatically detect new anomalies and attack patterns (Aamir & Ali Zaidi., 2021). Models utilise algorithms like supervised learning, clustering, neural networks and more. Benefits include adaptive detection as models self-improve over time and analysis is expanded beyond static rules or thresholds. Challenges include model complexity, false positives, and large sets of training data required (Aamir & Ali Zaidi., 2021).

Due to its capacity to identify intricate nonlinear correlations and patterns within network traffic data, neural networks have become a highly successful machine learning approach for DDoS detection (Najafimehr et al., 2022). Various neural network architectures are utilised, including convolutional neural networks that specialise in analysing input features maps, and recurrent neural networks like LSTMs that effectively process sequential data like network flows over time. Key benefits of neural networks include learning highly complex models to accurately classify normal vs DDoS attack traffic, and the ability to retrain the model on new data to continuously improve detection capabilities. However, significant volumes of training data are required.

Support vector machines (SVMs) are supervised learning models commonly leveraged for DDoS detection (Najafimehr et al., 2022). SVMs classify traffic as either normal or attack-related based on mapping input feature vectors to optimal decision boundaries or hyperplanes. SVM kernels like radial basis functions are utilised to handle nonlinearity. One-class SVMs can detect anomalies, while multi-class SVMs categorise type of attack (Najafimehr et al., 2023). SVMs deliver high accuracy but have limitations in handling new unknown attacks beyond training data.

Decision tree models classify traffic by modelling attack and benign flow properties and features through a tree of hierarchical rules. Trees are tuned via algorithms like ID3, Random Forest and C4.5 to determine optimal root-to-leaf branching logic based on information gain (Li et al., 2021). Random forest models further boost accuracy by aggregating results from an ensemble of decision trees analysing different traffic features. A key benefit of decision trees is the ability to explain the classification logic and extracted traffic attributes (El-Sofany., 2020).

## 2.14 Related Work

Several research publications described below used the same dataset as the current study, while some also used others.

In (Najafimehr et al., 2023) the authors categorised the DDoS attacks into two types: high-rate "flooding" attacks that waste bandwidth and low-rate attacks that mimic normal traffic patterns. TCP SYN floods, UDP floods, DNS amplification, and HTTP floods are examples of common flooding attacks.

The authors also suggested that there are two main approaches for DDoS attack detection: signature-based methods that recognise established attack patterns and anomaly-based methods that find anomalies from ordinary behaviour profiles (Najafimehr et al., 2023). The study suggests that machine learning techniques are frequently used to detect anomalies. Supervised algorithms such as decision trees, naive Bayes, and support vector machines require labelled data for training. Unsupervised techniques, such as k-means clustering and DBSCAN, can detect fresh attacks by clustering data. Machine learning is combined with other approaches such as entropy analysis in hybrid methods. Size, diversity, authenticity, labelling quality, and real-world network traffic are important training dataset criteria (Najafimehr et al., 2023).

The study used KDDCup99, NSL-KDD, UNSW-NB15, CICIDS2017, and CICDDoS2019 datasets in the study (Najafimehr et al., 2023). However, current datasets do not include new and complex real-world attacks. Differences between lab circumstances and actual attacks can diminish the efficacy of machine learning methods. The study using simulated data demonstrate good accuracy but low recall on live networks. Recommendations include creating larger and more realistic datasets, assessing algorithms on real-world testbeds, focusing on metrics such as recall rather than just accuracy, and merging machine learning with complementing techniques.

In (Alduailij et al., 2022) the author proposes a machine learning strategy for detecting DDoS attacks in cloud computing settings. The datasets CICIDS 2017 and CICDDoS2019, which contain network traffic flow records labelled with benign and DDoS attacks, are used. The CICIDS 2017 dataset has over 200,000 flows with 79 attributes, whereas the CICDDoS2019 dataset includes over 1 million flows. Two feature selection strategies are utilised to identify the most relevant features: Mutual Information (MI) and Random Forest Feature Importance (RFFI). MI measures the interdependence of characteristics and labels. The Gini impurity criterion is used by RFFI to select features that optimally segregate the data. These strategies are used to extract 16, 19, and 23 feature subsets from the original feature sets. Five supervised classifiers have been looked at: Logistic Regression, K-Nearest Neighbours, Gradient Boosting, Random Forest, and Weighted Voting Ensemble. To optimise each model, hyperparameter tuning via grid search is used. The models are trained on the reduced feature sets and assessed in terms of accuracy, precision, recall, and F1-score. Random Forest outperforms other classifiers on the CICIDS 2017 dataset, with the best accuracy of 99.99% with 19 features chosen by RFFI. The confusion matrices reveal that Random Forest and Weighted Voting Ensemble have lower misclassification rates than the alternatives. Experiments on the CICDDoS2019 dataset show that utilising MI and RFFI for feature selection reduces mistakes.

In comparison to existing approaches, the study suggested strategy using RFFI and Random Forest decreases misclassifications by utilising the optimal feature subset. It also improves on previous work's accuracy results. The findings emphasise the importance of machine learning model parameter adjustment and feature selection for improving DDoS detection performance in cloud environments.

In (Batchu & Seetha, 2021) the authors employed only the CICDDoS2019 dataset in the pre-processing step, which contains 12 distinct DDoS attacks. To solve class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) and Tomek Links are used to oversample the minority class and under sample the majority class. Based on domain expertise, missing values are imputed and details such as IP addresses are deleted. To normalise and alter the data, scaling and encoding techniques are used. To extract 8 ideal features, the model tuning phase adopts a hybrid feature selection strategy that combines filter methods such as deleting duplicate features with embedding methods such as Random Forest. Grid search is used on five classifiers to determine the best hyper parameters: Logistic Regression, Decision Tree, Gradient Boosting, K-Nearest Neighbours, and Support Vector Machine. The tuned models are trained on the balanced dataset with selected features to discriminate between normal and attack traffic during the classification phase. The models are trained and tested on different data splits from days 1 and 2 to evaluate four scenarios.

The suggested Gradient Boosting model beats prior methods such as Convolutional Neural Networks and Long Short-Term Memory networks, reaching 99.97% accuracy, 98.9% precision, 99.99% recall, 99.44% F1-score, and 99.97% AUC. The advanced data pre-processing and model tuning procedures are important to the proposed model's increased performance.

In (Marvi et al., 2020) the authors used the CICDDoS2019 dataset. The framework includes advanced data pre-processing, model tweaking, and classification stages. Exploratory analysis is performed throughout the data pre-processing phase to understand feature distributions and handle class imbalance using SMOTE and TomekLinks under sampling. Missing values are substituted, and category variables are eliminated. To convert the data into a usable format, critical feature scaling and encoding techniques are used. During the model tuning phase, a new integrated feature selection strategy is used, which combines filter methods such as F-test with embedded techniques such as Random Forest and LightGBM. This technique shrinks the high-dimensional feature space by 77% while keeping the most informative characteristics. Furthermore, 5 classification algorithms are subjected to rigorous hyperparameter tuning using grid search to discover optimal configurations. The tailored gradient boosting model is trained and verified on data from a single attack type. This training method improves generalisation capability for detecting various DDoS attacks .Extensive empirical testing demonstrates that the proposed technique performs extraordinarily well, with accuracy, precision, recall, and AUC of 99.97%, 99.99%, and 98.9%, respectively. Their results exceed previous machine learning and deep learning models on the same dataset.

The article contributes significantly to the development of a machine learning-based DDoS detection system. Data balance, thorough feature analysis, and model tweaking strategies are important to outperforming state-of-the-art methods in terms of accuracy. Testing on previously encountered attacks to assess generalisation capability is also commendable. One limitation is that just one dataset is evaluated. Validating the algorithms on other public DDoS datasets could demonstrate their robustness even further. Furthermore, testing in real-world networks could provide insights into practical issues. Overall, the research proposes a thorough machine learning approach for accurate and generalised DDoS detection.

## 2.15 Classification of Machine learning Algorithms

**2.15.1 Supervised Learning:**

To guide the learning process, algorithms are trained using input data that has been marked with the desired output. Among the most common supervised learning methodologies are:

* Linear regression, which predicts continuous target variables based on linear correlations with the input features (Aljuhani, 2021).
* Logistic regression, which handles binary classification tasks by estimating discrete class probabilities (Géron, 2022).
* • Decision trees, which use recursive data partitioning to build tree-structured models that improve classification or regression performance (Géron, 2022).
* • Random forests, an ensemble strategy that enhances overall accuracy by combining forecasts from numerous individual decision trees (Géron, 2022).
* Support vector machines, which separate classes using optimal decision boundaries maximising the margin between data points (Aljuhani, 2021).
* Naive Bayes classifiers, which apply Bayes' theorem with strong feature independence assumptions to perform probabilistic classification (Aljuhani, 2021).
* K-nearest Neighbours algorithms, which categorise new examples by finding the most similar labelled examples in the training data (Aljuhani, 2021).

***2.15.2 Unsupervised Learning:***

Unsupervised learning algorithms can discover hidden patterns and insights in data that lack pre-defined labels or categories. They investigate the fundamental structure of datasets in order to cluster data points with similar characteristics, reduce dimensions to core components, and extract relevant features. Unsupervised learning techniques that are widely used include:

* Clustering methods such as k-means, hierarchical clustering, and DBSCAN use proximity and density to group similar data points together (Aljuhani, 2021).
* Principal component analysis (PCA) and other dimensionality reduction approaches locate and compress high-dimensional data into its most impactful lower-dimensional representations (Géron, 2022). PCA can be supervised if labels are used.
* Independent component analysis (ICA) is a statistically independent method for separating multivariate signals into different components (Géron, 2022).
* Autoencoders are artificial neural networks that learn how to encode inputs in a lower-dimensional space and subsequently reconstruct the original inputs, facilitating dimensionality reduction and feature extraction (Géron, 2022).

***2.15.3 Reinforcement Learning:***

Reinforcement learning (RL) is a set of machine learning techniques used by autonomous agents to optimise control rules for sequential decision-making processes. Agents learn to maximise cumulative rewards through repeated episodes of interaction in dynamic environments. Reinforced learning techniques that are commonly used include:

* Q-learning, an off-policy algorithm that estimates long-term value functions for state-action pairs to derive greedy policies (Géron, 2022).
* Deep Q-networks, which integrate Q-learning with deep neural networks to approximate complex value functions and policies (Géron, 2022).
* Monte Carlo tree search, which constructs search trees online to evaluate future action sequences using randomness and simulations (Géron, 2022).
* Policy gradient methods that directly adjust policy parameters via gradient ascent on expected rewards.
* Proximal policy optimisation, an on-policy algorithm enhancing sample efficiency and stability for policy gradient approaches.

***2.15.4 Hybrid Learning:***

Hybrid learning refers to the combination of several types of learning algorithms and methodologies to generate more powerful and adaptable models. Here are some important facts concerning hybrid learning:

* It leverages the strengths of different approaches to offset weaknesses. For example, combining supervised and unsupervised learning allows models to benefit from labelled and unlabelled data (Maier et al., 2022).
* Popular hybrids include ensemble methods like random forests that combine multiple decision tree models to reduce variance and improve generalizability (Maier et al., 2022).
* Neural networks can also be hybridised by having multiple types of layers suited for different tasks. For instance, CNNs for feature extraction and RNNs for sequence learning (Maier et al., 2022).

***2.15.5 Deep Learning***

* Models are constructed up of stacked layers of artificial neural networks between the input and output. Each layer transforms the data from the preceding layer using a non-linear transformation (Lye et al., 2020). Many layers enable the modelling of extremely complicated interactions.
* A deep neural network's hidden layers collect significant features from raw input data in a hierarchical manner. Lower layers recognise fundamental patterns, whereas higher levels recognise abstract concepts. This eliminates the requirement for feature engineering by hand.
* To learn strong feature representations, models must be trained on thousands to millions of tagged samples (Maier et al., 2022). The massive datasets enable for the identification of detailed patterns for enhanced prediction. Deep learning models overfit when there is insufficient training data.
* Computationally intensive training entails iteratively propagating data through numerous layers to update internal weight parameters. Training across multiple epochs using graphics processing units (GPUs) or specialised hardware like tensor processing units (TPUs) to expedite training yields cutting-edge findings (Lye et al., 2020).
* Neural networks, interconnected layers of simple computing nodes inspired by biological neurons, capable of learning complex nonlinear mappings between inputs and outputs (Aljuhani, 2021).

## 2.16 Tools review

Jupiter Notebook is the chosen learning environment. This choice stems from its interactivity for quick experimentation, seamless data analysis and visualisation capabilities, documentation and collaboration assistance, and accessibility for education and sharing of replicable outcomes. The ability to combine code, explanatory text, mathematical expressions, and graphical features in a single document makes it particularly well-suited for the complexities of machine learning operations.

Python is the preferred programing language as it open source and all necessary libraries are easily available. With python being one of the most used programming languages there is a vast amount of built in tools and very versatile and easy to read.

## 2.17 Reinforcement Learning Challenges for DDoS Attack Detection

Reinforcement learning algorithms could theoretically be applied to detect DDoS attacks. However, in practice RL is rarely employed for this specific task due to several key challenges:

Delayed Reward Function a core component of RL involves an agent learning based on rewards, which may only be observed in the future (Fachantidis et al., 2017). False positives may only be evident after business impact. True positive rewards rely on successful attack mitigation, thus the delay makes reward engineering tricky.

Excessive Complexity RL introduces substantial complexity versus other machine learning techniques (Fachantidis et al., 2017). Simpler supervised learning using models like decision trees or support vector machines may be equally effective for DDoS detection. Unsupervised learning through clustering or auto encoders is another viable approach. The complexity of RL may be unwarranted for this task.

Data Scarcity Effective RL requires large, labelled datasets. Acquiring and annotating significant volumes of data specific to DDoS attacks presents a challenge. Lack of ample relevant training data limits the applicability of RL in this domain.

Computational Cost RL algorithms are computationally expensive, requiring extensive training time. This makes RL poorly suited for real-time DDoS detection, where analysis and detection must be performed rapidly using live network traffic data. The computational burden renders RL impractical.

However despite all the downside it is still possible to use reinforced learning like Bhutto et al., 2022 where they used Reinforced Transformer Learning for VSI-DDoS Detection in Edge Clouds.

The review finds encouraging developments, with methods like ensemble learning, neural networks, and support vector machines successfully used for DDoS detection using network traffic data. In particular, deep learning architectures have demonstrated strong potential for automated feature learning. However, there are still issues with model flexibility and explainability. The literature also highlights unresolved issues including investigating online and unsupervised learning algorithms.

In conclusion, as compared to conventional DDoS defence solutions, intelligent machine learning algorithms have a significant potential for real-time detection, speed, and accuracy. However, additional study is required to increase the model's robustness, interpretability, and capability to identify zero-day assaults. Adaptive ensemble approaches and utilising cutting-edge deep learning techniques promise to be particularly interesting paths for future research as DDoS tactics continue to change. Networked systems will become more safe and resilient when machine learning is applied to DDoS defence.

# Chapter 3: Proposed Method and Dataset

While the body of existing literature serves as a starting point, more investigation is required to create machine learning methods that can precisely identify new DDoS attacks in actual networks. This section outlines the suggested approach and sample size for an experimental study meant to overcome existing constraints.

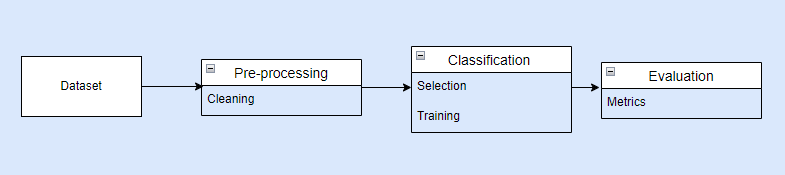


Figure 7: Operational framework development model.

## 3.1 Random Forest

The Random Forest technique is a supervised learning method that may be applied to classification and regression tasks. It is an ensemble technique, which implies that it makes predictions by combining numerous decision trees rather than depending on a single decision tree (Maslan et al., 2020).

Some features of the random forest include:

Based on the bootstrap aggregation principle, often known as bagging; sampling with replacement from the original set, numerous versions of the training set are created (Biau & Scornet., 2016). On each of the bootstrapped training sets, a distinct decision tree is trained. Random Forest is a fundamental strategy for constructing decision trees, instead of evaluating all features when selecting how to split a node, it selects only a random selection of features. This brings randomness into the trees and de-correlates them, resulting in a more robust ensemble (Biau & Scornet., 2016).

The algorithm employs a technique known as out-of-bag evaluation. Given each tree is trained on a separate bootstrapped subset, approximately one-third of the samples are excluded from the training set for that tree (Biau & Scornet., 2016). These out-of-bag samples can be used to calculate out-of-bag error, which is an unbiased measure of the model's performance. This enables for the detection of overfitting without the need for a separate validation set.

Each individual decision tree in the forest analyses the new input data point and creates a prediction to make a prediction with a random forest. This is equivalent to every tree casting a "vote" for the class it believes the data point belongs to in classification tasks (Biau & Scornet., 2016). The random forest then accumulates all of the votes from the various trees and utilises majority voting to predict the overall class; the class with the most "votes" from the decision trees is expected to be the final class. Individual predictions from all decision trees are averaged together to create the final predicted value for the new data point in regression tasks. Resulting in the predictions of the various trees is integrated in some way to produce the overall forecast, whether through voting for classification or averaging for regression (Biau & Scornet., 2016).

Gini impurity is one of the factors used to make decisions when constructing individual decision trees (Gwetu et al., 2020). Gini impurity assesses how likely it is that a randomly selected element would be incorrectly labelled if it was randomly classified according to the class distribution (Gwetu et al., 2020). Higher Gini impurity means more incorrectly classified samples. Decision trees, like entropy, seek to minimise Gini impurity while evaluating probable splitting points in order to divide data into purer subsets.

A third of the samples are excluded and not used to train any given tree because each tree is trained on a subset of data. This out-of-bag (OOB) data can be used to calculate the OOB error, which is an impartial measure of the model's performance (Gwetu et al., 2020).

## 3.2 K-Nearest Neighbours

The K-Nearest Neighbours (KNN) algorithm is a supervised machine learning method that can be used for classification and regression applications (Sayed et al., 2021). Its central principle is to discover the K nearest data points or "neighbours" to a new input and use those neighbours to forecast the label or value for that new point.

Choosing the appropriate value for K is an important component of KNN, a low K risk being influenced too much by outliers, whereas a large K risk missing local patterns in the data. Tuning K allows trade off bias versus variance (Sayed et al., 2021). KNN is also computationally costly, especially with large datasets.

KNN can be beneficial for detecting DDoS attacks since network traffic patterns during the attack differ from normal traffic (Alduailij et al., 2022). KNN classifies new traffic data points based on their resemblance to the training data's normal vs attack samples. Because of its local, instance-based approach, it excels at pattern detection.

## 3.3 Neural Network

A neural network is made up of an interconnected network of nodes called neurons that are organised in layers and work together to process input data (Kindermans et al., 2017). The raw data to be analysed is delivered to the input layer. This is followed by hidden layers, which are made up of neurons that compute the data and pass it on to the next layer. The final layer of output returns a forecast or categorisation (Kindermans et al, 2017).

Neural networks operate in such a way that they are excellent at detecting anomalies. Through pattern recognition over multiple hidden layers, neural networks can learn to identify anomalies in traffic and recognise attack signatures among regular variations (Kindermans et al., 2017). Their adaptive learning approach recognises new attack varieties by updating model weights. The algorithm may automatically extract meaningful information from raw traffic data as part of the multilayer processing. This makes the neural network algorithm in theory a viable technique for constructing robust, reliable DDoS detection systems due to their representational power and adaptive learning capabilities.

The main computations take place within the neurons that span the hidden layers. Each neuron weights the various input features and computes the weighted sum. The sum is then run through a nonlinear activation function, such as a sigmoid or ReLU function, with an additional bias inserted (Kindermans et al, 2017). This allows for the simulation of intricate interactions between inputs and outputs. Through a process known as backpropagation, the network learns the best weights to minimise the loss between predictions and true targets, iteratively altering weights to improve accuracy.

## 3.4 Naïve Bayes Classifier

The Naive Bayes classifier is a simple supervised machine learning technique for classification tasks. It is based on the Bayesian theorem and assumes that features are conditionally independent in the presence of the class variable (Mishra & Singh., 2022).

The algorithm calculates the likelihood that a given data point belongs to a specific class based on its feature values (Mishra & Singh., 2022). It estimates this probability using Bayes' theorem.  The prediction is given to the class with the highest posterior probability.

Naive Bayes could be useful for detecting DDoS attacks since it predicts the conditional likelihood of attack and regular traffic depending on traffic characteristics such as packet size, time, etc (Mishra & Singh., 2022). This probabilistic technique helps deal with traffic pattern fluctuations and performs effectively with high-dimensional data. DDoS attacks are more likely to produce distinct feature distributions than regular traffic; Naive Bayes should detect these odd patterns for accurate attack detection.

## 3.5 Autoencoder

An autoencoder is a form of artificial neural network that is used for unsupervised learning of efficient input data coding’s. It works by encoding the input into a lower-dimensional representation, then decoding that representation back into the original input space (Tennakoon & Fernando., 2022).

Autoencoders are made up of two parts: an encoder that compresses the input into a latent space code and a decoder that reconstructs the code's output. The network is trained to minimise the disparity between input and output, forcing it to capture the most important data aspects in the code (Tennakoon & Fernando., 2022).

Autoencoders can be applied to network traffic data to learn a representation that characterises regular traffic patterns for DDoS attack detection. During an attack, traffic distribution would deviate from the norm, resulting in more reconstruction error. The autoencoder model can identify anomalies and detect potential attacks by establishing an error threshold.

A hybrid approach can be used with the autoencoder and random forest. The method combines the advantages of unsupervised and supervised learning paradigms by combining an autoencoder's ability to learn typical traffic patterns in a self-directed manner with a random forest's ability to exploit labelled data for identifying known harmful signatures (Sabrina et al., 2021). This gives the ability to detect anomalies that deviate from a baseline as well as classify specific attacks based on discriminative traits. Furthermore, the unsupervised nature of the autoencoder allows it to possibly discover novel attack variations that a supervised model could miss, yet the random forest remains robust in identifying proven malicious patterns (Sabrina et al., 2021).

This hybrid strategy could also contribute to addressing the on-going arms race between attackers and defenders, in which attackers constantly innovate strategies to avoid detection. The autoencoder's flexible learning adjusts to changes in normal traffic continually, while the random forest retrains on fresh labelled data of attacks, both of which contribute to countering the arms race dynamics through constant monitoring (Sabrina et al., 2021). Furthermore, rather than relying on a single technique, the capabilities of efficient data representation by the autoencoder and robust ensemble learning by the random forest complement each other when employed in tandem.

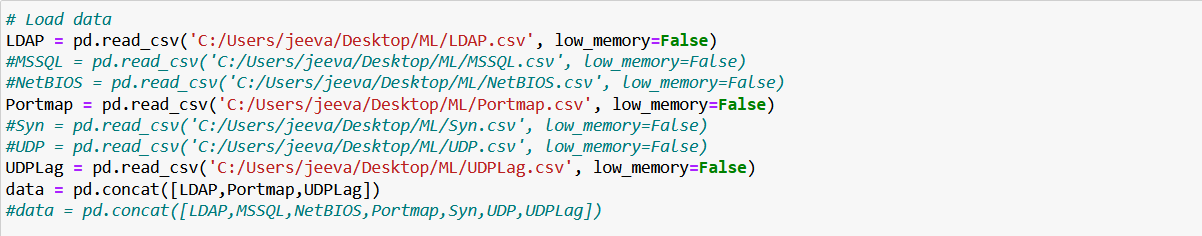
## 3.6 CIC-DDoS2019 Dataset

The CIC-DDoS2019 dataset offers a valuable resource for training and evaluating machine learning-based approaches for detecting Distributed Denial of Service (DDoS) attacks. Introduced in 2019, it provides a large, contemporary selection of network traffic data along with labels identifying the types of packets in the file. The dataset was generated within a rigorously configured emulation environment designed to resemble real-world networks and traffic behaviours. There are 7 CVS files in the dataset named: LDAP, MSSQL, NetBIOS, Portmap, Syn, UDP and UDPLag which totals roughly 8.7 GB and 88 features.

A key advantage of using CIC-DDoS2019 is that it enables supervised learning on recent real-world attack samples. The diversity of malicious and benign behaviours can train robust models capable of detecting known and zero-day DDoS attacks. Public availability and widespread adoption further facilitates standardised benchmarking and reproducible comparisons.

However, the simulated nature of the dataset has limitations. It may not fully capture the complexity and diversity of modern high-throughput networks. Models derived from this data could fail to transfer effectively to operational environments. Regular updating with new data points can help address evolving attack types and benign traffic patterns. Additional real-world network data should be incorporated to improve generalisability. Ideally a hybrid training approach combining simulated and real-world data may produce the most resilient and practical DDoS detection models.

Due to a lack of computational resources, the three smallest datasets are chosen, therefore the implementation will be based on the Portmap, UDPLag, and LDAP CSV files. In turn using a smaller dataset can lead to different results as larger datasets usually are more accurate (Najafimehr et al., 2023). Figure 8 shows datasets used. Figure 9 shows the split in packet label in the three datasets used.



*Figure 8: Datasets used*

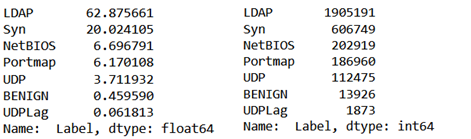


Figure 9: Split in packet attack types in three datasets.

## 3.7 Activation functions

Machine learning algorithms like neural network rely heavily on activation functions to help optimise them. They are mathematical equations that, given an input, determine the output of a node or model. Non-linearities introduced by activation functions enable machine learning algorithms to discover complex relationships and patterns in data. A few of the more popular activation functions are:

Sigmoid - Squashes inputs to probabilities between 0-1 and is used for binary classification output layers (Yuen et al., 2021).

ReLU - Rectified Linear Unit returns maximum of input or 0 and is popular for hidden layers (Yuen et al., 2021).

Leaky ReLU - Variant of ReLU that returns a small slope for negative values. Fixes "dying ReLU" issue (Yuen et al., 2021).

Adam (Adaptive Moment Estimation optimiser) - Computes adaptive learning rates during training (Wassermann et al., 2019).

To summarise, this paper proposes detecting DDoS attacks using the models random forest, k-nearest neighbours, neural network, autoencoder with random forest and Naïve Bayes classifier. The CIC-DDoS2019 dataset will be leveraged, which provides labelled network traffic captures reflecting modern DDoS tactics. This contemporary, publicly available dataset will enable more robust evaluation. These algorithms will classify network traffic as either benign or a DDoS attack based on learned patterns.

Ensemble approaches will be investigated by merging autoencoder output with a random forest classifier. This aims to increase detection accuracy across the wide range of attack types that the CIC-DDoS2019 dataset represents. The effectiveness of each model will be assessed using parameters like precision, recall, F1-score, and computing overhead.

# Chapter 4 Implementation

This chapter will demonstrate how python was used to develop the machine learning DDoS detection models that were suggested in this study since it is frequently used for data science applications. To facilitate effective model building and evaluation, several specialised libraries were used.

## 4.1 Process of Machine Learning

The machine learning process can be broken down into several stages:

Developing and deploying machine learning models successfully demands the precise execution of several essential processes. First, clearly identify the prediction or decision task at hand, then specify relevant performance measures and model requirements based on how the model will be used (Marvi et al., 2020). Next, obtain or build a substantial, representative dataset with appropriate attributes to train the model strategically. The raw data may need to be improved by cleaning missing values, dealing with outliers, designing relevant features, and reducing dimensions. Select a relevant class of machine learning methods, optimise model hyper parameters, and fit the parameters to the training data using the pre-processed data. Before deploying the model, statistically assess its performance using conventional metrics such as accuracy, AUC, precision, recall and F1 score on the test data (Marvi et al., 2020). Finally, integrate the validated model into production systems to generate predictions or decisions on new real-world data.

## 4.2 Data Pre-Processing

As previously indicated, the data quality is already extremely good; nevertheless, there are certain difficulties such as infinity and NaN values not being processed by the algorithms, as well as problems with the "SimillarHTTP" column. The data pre-processing code is shown in Figure 11.

There were a total of 19 dropped features with the majority of them being dropped due to the standard deviation of their column being zero. This would result in unnecessary computational power being used as the result in theory will always be the same. Other columns were dropped The columns were dropped owing to conversion concerns with IP addresses and other columns not being required as they are also non-numeric or identifiers. The following columns have been removed, bringing the total number of features to 69:

['Unnamed: 0', 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp', 'Label', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'FIN Flag Count', 'PSH Flag Count', 'ECE Flag Count', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate'].

A screenshot of a computer

Description automatically generated

Figure 10: example of features with standard deviation zero

Given that the value of the simillarHTTP column is a string, it can be translated into numeric labels using the label encoder so that the algorithms can interpret it.

To address infinity values, they are first converted to NaN (Not a Number) values, resulting in all missing and outliers in the data having the same value.

The missing values section replaces any missing values with the column's mean value and refreshes the dataset.

A close-up of a computer code

Description automatically generated

Figure 11: Section of code to fill in missing data.

4.3 Random Forest

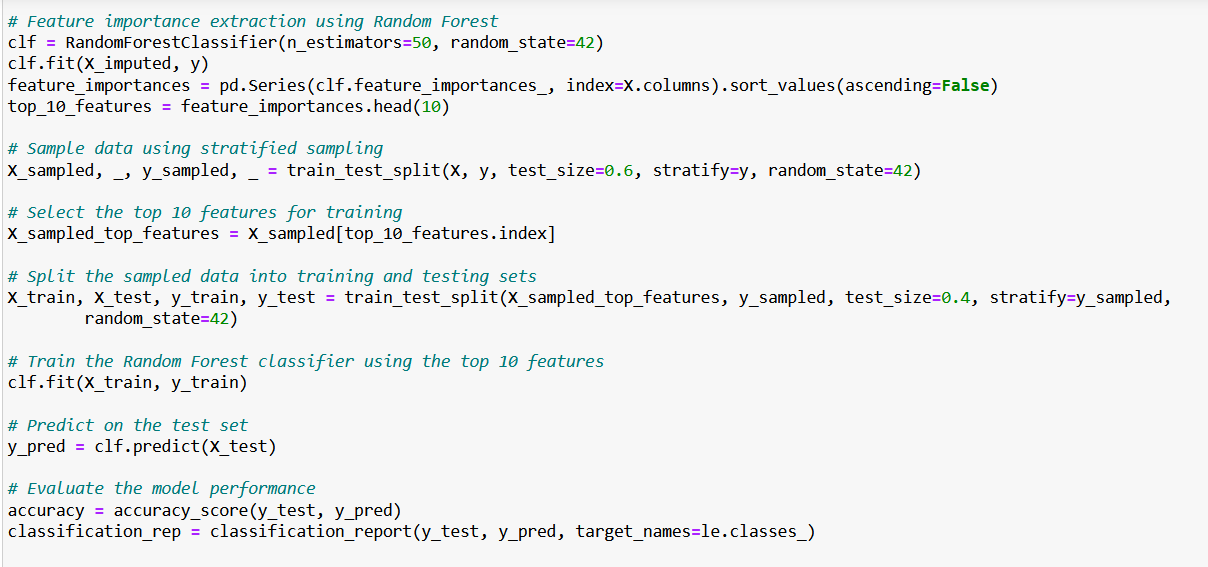


Figure 12: RFFI and random forest classification.

When looking at the feature importance the algorithm uses n\_estimators 50 and a random state of 42. The estimators are set to 50 to speed up the amount of time it takes for the algorithm to process the data, scikit-learn usually uses 100 trees by default. The random state is set to 42 to ensure that the results are to ensure reproducibility. Using the RFFI allows the top 10 most suitable feature for the random forest algorithm to be extracted.

Next clf.fit(x\_imputed, y) is used to imput the data. Here x\_imputed is taken from the preprocessing section where the NaN values were replaced with the mean and y is used to show the correct outcomes of the algorithm. After this is computed the output can be sorted to show the features of most importance in descending order and can be saved as top\_10\_features. Figure 13 shows the results.

A screenshot of a computer

Description automatically generated

Figure 13: Ten features selected by the random forest feature importance.

The data is then split using stratified sampling to maintain the distribution of the target variable y. Having a test size of 0.6 shows that only 60% of the data is set aside as a test set (which isn't used in subsequent steps), and the remaining 40% is kept for further processing. The top 10 features are then sampled.

Data is then split into training and testing with a split of 40% for testing and 60% for training. The Random Forest classifier is then re-trained using the training data with only the top 10 features which gives predictions which can be evaluated show in in figure 12.

## 4.4 Neural Network

In order to run the neural network, the necessary modules need to be imported as seen in fig 14, keras may need to be imported as it is not installed by default on Juypter Notebook . The function is to\_categorical which is used to convert class vector (integers) to binary class matrix (one-hot encoding). StandardScaler is used to standardise the features by removing the mean and scaling to unit variance.

The feature is then scaled, and the scaler instance is established The method "fit\_transform" computes the mean and standard deviation of "X\_train" for scaling (standardising) and then applies this scaling to the training data. Resulting in a standardised version of "X\_train" called "X\_train\_scaled". Finally the previously computed mean and standard deviation is applied to the test data. This ensures that the test data is scaled proportionally with the training data; the outcome is "X\_test\_scaled," a standardisation of "X\_test".

One hot encoding is used as it will allow the neural network to output multi-class classification otherwise the output would only show two outputs classifying the packet as part of a DDoS attack or normal traffic. The variable num\_classes will represent each label in the dataset. The following two lines convert the class labels in “y\_train” and “y\_test” to one-hot encoded matrices.

A screen shot of a computer code

Description automatically generated

Figure 14: StandardScaler and one-hot encoding conversion.

The modules “sequential” is used to construct a model layer by layer and “dense” is used fully connected layer in a neural network, in which each neuron in the layer is connected to every neuron in the preceding and following layers as show in in figure 15.

The model is initialised and will be the starting point for adding layers to the neural network. The model.add() method is used to add a layer to the neural network model. Dense module is used to add on the new layers and for the first layer the number of neurons used is 64 and has been activated using relu. Relu was chosen since it is a simple and widely used with neural networks and it merely tests if the result is positive or negative. If the value is negative, relu returns 0; if it is positive, the value remains constant. The “input\_shape” parameter specifies the shape of the input data, which is the number of features in “X\_trained\_scaled”.

The next layer is exactly the same however it uses 32 neurons rather than 64 to save on computation power. The shape is not defined due to the first layer having this defined so it is presumed that all subsequent layers are the same.

The final dense layer has a number of neurons equal to “num\_classes”, which corresponds to the number of unique classes in the target variable. The activation function is “softmax” implies that this is the output layer, and converts the network's output into probability distributions over the classes.

The model is then compiled, meaning it is being set up for the learning process. The loss function “categorical\_crossentropy” is used as the target variables are one-hot encoded. It calculates the difference between the predicted and actual target class probability. The optimiser “adam” which stands for Adaptive Moment Estimation and adapts the learning rates of each parameter during training making it effective for a detecting DDoS attacks. Finally the metrics functions will monitor the accuracy.

A computer code with text

Description automatically generated

Figure 15: Layering the neural network.

Next the model is trained using model.fit as show in in figure 16 and uses the training data and labels created in the previous steps. The function trains the model for a fixed number of times (50) which are defined using the epoch’s variable, meaning the data is looked at 50 times. The batch size is set to 256 this the model will update its weights after processing every 256 samples during each training iteration. This mini-batch gradient descent addresses the issue of computationally expensively when performed on the entire dataset. Validation\_data allows the model to evaluate the performance of each epoch compared to the validation data. The history variable in turn will store the training history which includes metrics such as the accuracy and loss for each epoch on both the training and validation datasets.

A screenshot of a computer code

Description automatically generated

Figure 16: starting neural network.

## 4.5 K-Nearest Neighbours

The module KNeighborsClassifier needs to be imported so that the algorithm can be worked with. The KNN variable is an instance of the k-NN classifier and is set to have 5 neighbours, which indicates that when making predictions for a new instance, the algorithm will evaluate the 5 closest data points. The classifier is trained using the top 10 features which were found using the random forest algorithm; the predictions are made on the test data, “X\_test” and predictions are stored in “y\_pred\_knn” shown in figure 17 for source code.

A screenshot of a computer

Description automatically generated

Figure 17: Creation of KNN classification.

## 4.6 Auto encoder and Random forest

As the autoencoder is a type of neural network it first needs to be standardised to ensure that all features have comparable scales and improves convergence during training.

The standard sclaer module is imported to standardise the features by removing the mean and scaling to unit variance. An instance of “StandardScaler” can be created to hold the scaled features from the test and training dataset. Under that the scaled features are set to the variables “X\_train\_scaled” and “X\_test\_scaled” and are accomplished with the method “transform” as seen in fig 18.

A screen shot of a computer code

Description automatically generated

Figure 18: Scaling data for autoencoder.

In fig 19 the number of features needs to be set under the variable input\_dim and the encoding \_dim is set to five. Encoding\_dim variable represents the number of neurons in the encoding layer of the autoencoder neural network. The input data is encoded into a 5-dimensional feature space, which compresses the data while keeping the most critical information.

Following that, the decoding layers attempt to recreate the original input data from this low-dimensional compressed representation. After that, the input layer is generated with the input shape having the same dimensions as the number of features. There are two encding layers: The first dense layer has 7 neurons and the ReLU activation function transforms the input data. The second “Dense” layer with encoding\_dim neurons further reduces the dimensionality of the data. (Encoded) is key as it connects the previous layer's output to the new encoding layer's inputs.

Once the layers have been encoded they are decoded again using the same amount of layers and same inputs except instead of “ReLU” “sigmoid” is used. This is due to it haing better activation functions when looking at reconstruxction ranges, probability and avoiding saturation when comparing to “ReLU”. The aim is to produce an output in the range of [0, 1] for each feature.

The autoencoder is built and uses the input layer and the output of the “decoded” layer.

The next line: encoder = Model(input\_layer, encoded) is used to extract the encoded representations of the input data.

Finally the autoencoder is compiled and uses the optimisation algorithm “adam” and loss function “mean\_squared\_error”. The “adam” optimisation algorithm adaptively estimates the first and second moments of gradients to modify the learning rate for each weight parameter. This aids in the acceleration of convergence and the handling of sparse gradients.

The loss function "mean\_squared\_error" computes the mean of the squared differences between the target and forecasted values. It quantifies the difference between the autoencoder's original inputs and reconstructed outputs. Minimising the loss the autoencoder weights will be adjusted to create reconstructions that closely resemble the original data.

A computer screen shot of text

Description automatically generated

Figure 19: Setting up autoencoder.

The autoencoder is trained using the scaled training data both input and target. It trains for 50 epochs with a batch size of 256, rearranging the data between them. Validation data is supplied in order to monitor the model's performance during training as shown in fig 20.

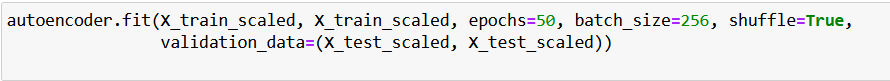


Figure 20: Autoencoder trained on scaled data

Finally, the trained encoder can be used to encode the scaled training and test data, resulting in the encoded representations show in in fig 21.

A black text on a white background

Description automatically generated

Figure 21: Autoencoder encodes scaled data.

The encoded data is utilised in conjunction with the random forest technique shown in figure 22.

A screenshot of a computer program

Description automatically generated

Figure 22: Encoded data is inputted into the random forest.

## 4.7 Naïve Bayes classifier

First the Gaussian Naive Bayes algorithm is imported using GaussianNB. The data is divided between training and test sets using the "train\_test\_split" function. The random state function ensures that the splits are repeatable and that the data is split 60/40 for training and testing.

The algorithm is initialised and the training data is input. The predictions are made on the test data, “X\_test” and stored in “y\_pred\_gnb”. Along with this the accuracy score and classification report is generated.

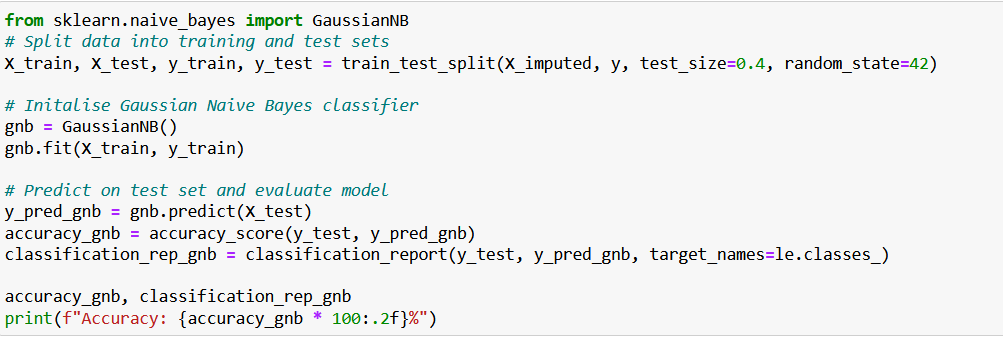


Figure : Creation of Naïve Bayes classifier

To summarise, the proposed DDoS detection models in this work were implemented using Python and all necessary libraries and packages are detailed so it can be recreated. Implementations of classification methods like random forest, KNN, Naive Bayes, and neural networks were made available through the SciKit-Learn library. The construction and training of the autoencoder were made possible by Keras and TensorFlow.

# Chapter 5: Evaluation and results

## This chapter presents the evaluation process and results for the machine learning DDoS detection models proposed and implemented in this study.

## 5.1 Evaluation metrics

Some metrics and notations are utilised for further evaluations, and they are as follows:

True Positives (TP): The number of entries in the positive class that the model correctly classifies as "Positive."

True Negatives (TN): The number of records in the negative class that the model correctly classifies as "Negative."

False Positives (FP): The number of records in the negative class that the model incorrectly recognises as "Positive."

False Negatives (FN): The number of entries in the positive class that the model incorrectly classifies as "Negative."

**5.1.1 Precision**

Precision is a key evaluation parameter for DDoS detection classifiers, calculating the fraction of valid positive predictions out of all positive predictions generated. It is defined as the ratio of true positives (TP) to all projected DDoS attack detections TP / (TP + FP), where true positives are malicious traffic correctly categorised as a DDoS attack and false positives (FP) are normal traffic instances wrongly classed as an attack (Najafimehr et al., 2023). Precision denotes a DDoS attack detector's accuracy, specifically its ability to avoid false alerts.

**5.1.2 True Positive Rate**

The True Positive Rate (TPR) is an important performance indicator that gauges the proportion of true positive cases that are accurately identified as such (Najafimehr et al., 2023). Formally, True Positive Rate is defined as the ratio of true positives to the sum of true positives and false negatives, as calculated by the formula:

TPR = TP / (TP + FN)

**5.1.3 False Positive Rate**

The False Positive Rate (FPR) is a crucial performance indicator that quantifies the proportion of true negative cases that the model incorrectly identified as positive (Najafimehr et al., 2023). The False Positive Rate, often known as the False Alarm Rate, is formally defined as:

FPR = (TN + FP) / FP

**5.1.4 Accuracy**

Keeping track of the overall accuracy during training provides useful information about the classifications performance (Najafimehr et al., 2023). The ratio of correct classifications to total predictions across all traffic cases is defined as accuracy:

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

Accuracy is simply the overall rate of correct predictions across all classes in most classification situations. However, problems develop when one class is far more common than others, or when the impact/costs of multiple types of misclassification are relatively uneven. In addition, failing to detect malicious traffic (false negatives) might be significantly more damaging than false alarms (false positives) (Najafimehr et al., 2023).

**5.1.5 F1 Score**

The F1 Score is calculated by taking the harmonic mean of precision and recall. It gives a balanced assessment of a classifier's performance, especially when the positive and negative classes are imbalanced. This accuracy metric calculates how many times a model predicted correctly over the full dataset (Alduailij et al., 2022).

F1 Score = 2 \* (Precision \* Recall) / (Precision + Recall)

**5.1.6 Receiver Operating Characteristic and Area under the ROC Curve**

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a classifier's performance at various thresholds. It shows the True Positive Rate (TPR or Sensitivity) against the False Positive Rate (FPR) while the categorisation threshold changes. The Area Under the ROC Curve (AUC) evaluates the classifier's overall performance, with a greater AUC indicating better performance.

There is no straightforward mathematical procedure that can be used to calculate ROC and AUC by hand. They are commonly generated using software libraries or functions that take as input the classifier's prediction probabilities and true labels.

The area under the ROC curve is measured by the AUC metric. AUC is a metric that measures classifier performance across all feasible thresholds (Cummings, 2023).

AUC is defined formally as the chance that a randomly selected positive example is ranked higher than a randomly selected negative example. AUC values range between 0 and 1, with higher values suggesting better model discrimination between positive and negative classes. AUC of 0.5 indicates the model is no better than a coin flip, between 0.7 and 0.8 are considered acceptable, 0.8 to 0.9 are considered excellent, while >0.9 are considered outstanding classifiers (Cummings, 2023); the greater the variation in AUC ratings, the greater the performance disparities between models. An AUC of 1 denotes flawless classification, with no false positives and 100% true positives.

Scale-invariance, classification-threshold invariance, and classifier assessment independent of the distribution of positive/negative cases are key advantages of AUC. Rather than relying on absolute numbers, AUC as a ranking indicator measures how well model assesses order situations. AUC measures the model's intrinsic discrimination ability rather than the dataset's.

AUC provides a reliable single-number benchmark of model performance for binary classification problems. In multiclass issues, the Macro-Averaged AUC combines the AUC for each class vs. the remainder (Cummings, 2023).

## 5.2 ROC Curve comparison

**5.2.1 Random Forest**

The ROC curve of the RF classifier is superb and hugs the top left corner. This shows that it has a very high true positive rate with a low false positive rate, indicating that it accurately identifies positive cases while minimising false alarms. The curve begins with a sharp inclination, indicating good sensitivity even at very low false positive rates. In general, the RF curve exhibits excellent classifier behaviour.

A green and blue line graph

Description automatically generated

Figure 24: Random Forest ROC curve

**5.2.2 K-Nearest Neighbours**

The KNN model has a curve that is extremely similar in shape and position to the Random Forest model and it similarly performs quite well. With a sharp beginning rise, it stays near the upper left corner. The comparison confirms that KNN is comparable to RF for identifying DDoS attacks. The curve may be slightly less convex than RF, implying that RF has a modest advantage in maximising genuine positives while minimising false positive rates. However, KNN appears to be excellent overall.

A green and blue line graph

Description automatically generated

Figure 25: ROC curve for KNN

**5.2.3 Neural Network**

The neural network ROC curve is much lower than the RF and KNN ROC curves, indicating poorer discrimination. The curve climbs more gradually, indicating that the classifier does not consistently detect positive cases until false positive rates are greater. The position of the curve indicates significantly lower sensitivity. This supports the numerical measurements that reveal neural networks underperform for this task in the absence of further tweaking and training.

A graph with a line

Description automatically generated

Figure 26: Neural network ROC curve

**5.2.4 Autoencoder and random forest:**

The autoencoder curve is similar in form and location to the independent RF model, indicating that adding autoencoder features to an already good classifier delivers negligible benefit.

The ROC curves support the quantitative metrics-derived findings. They confirm the overshadowing performance of Random Forest and K-Nearest Neighbours, highlight the room for improvement in neural networks, and suggest that Naive Bayes could potentially contribute value in auxiliary scoring roles, despite its weaker classification capabilities. These graphical representations provide a more sophisticated understanding of the differences between classifiers.

A green and blue line graph

Description automatically generated

Figure 27: ROC curve for autoencoder + random forest

**5.2.5 Naive Bayes**

While the Naive Bayes classifier exhibited poor classification metrics, its ROC curve is higher in the upper left corner, albeit lower than RF and KNN. The initial strong increase suggests that, with correct threshold selection, the NB scores could be useful for ranking cases even if the raw decision boundary is unsuccessful. This is consistent with the high AUC, indicating scoring viability.

A graph of a curve

Description automatically generated

Figure 28: ROC curve for Naive Bayes classifier

## 5.3 Performance metrics results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Accuracy | TRP | FPR | Precision | F1 Score | ROC AUC |
| RF | 0.934587 | 0.93459 | 0.01090 | 0.934539 | 0.934539 | 0.808128 |
| KNN | 0.934587 | 0.93371 | 0.01175 | 0.934539 | 0.934561 | 0.811438 |
| Neural Network | 0.932151 | 0.93215 | 0.01131 | 0.932106 | 0.932151 | 0.457405 |
| Autoencoder + RF | 0.934136 | 0.93414 | 0.0198 | 0.933888 | 0.933993 | 0.806014 |
| Naive Bayes Classifier | 0.657716 | 0.65772 | 0.05705 | 0.673576 | 0.574567 | 0.923801 |

Figure 29: Classification performance metric results

Random Forest has the highest overall accuracy of 93.46% and the highest true positive rate of 93.46%, suggesting that it properly identifies the majority of DDoS attacks. The false positive rate was modest, at 1.09%, indicating that just a small percentage of typical traffic was misclassified as an attack. Precision and recall were both 93.45%, and the F1 score was also 93.45%, indicating a high positive predictive value. The RF model has an excellent discriminative ability, with a ROC AUC of 0.81.

The K-Nearest Neighbours model performed similarly to RF, with a 93.46% accuracy. The genuine positive rate was 93.37%, which was still very high. The false positive rate was 1.17%, which was equivalent to RF. Precision, recall, and F1 score all matched RF at 93.45%, 93.37%, and 93.46%, respectively. The ROC AUC of 0.81 was equal to the RF, demonstrating that the classification performance is strong.

The accuracy of the Neural Network classifier was 93.22%, which was matched by the true positive rate. The false positive rate was 1.13%, which was comparable to RF and KNN. However, precision and recall were 93.21% and 93.22%, respectively, and the F1 score was 93.22%, all of which were lower than the other top performers. The ROC AUC of 0.46, in particular, illustrates the neural network's poor classification performance.

The addition of Autoencoder features to RF resulted in modest improvement, with accuracy, true positive rate, precision, recall, and F1 score staying comparable to those of the standalone RF. The false positive rate, on the other hand, was greater at 1.98%, and the ROC AUC was somewhat lower at 0.80.

With a poor accuracy of 65.77%, a true positive rate of 65.72%, and a false positive rate of 5.71%, Nave Bayes performed much worse than other classifiers. Precision, recall, and F1 score were all low as well. It did, however, have a good ROC AUC of 0.92, indicating that its score outputs could be useful if correctly calibrated.

#### 5.4 Discussion

The Random Forest and KNN models performed best, with roughly similar classification scores. The main difference was that KNN had a slightly lower false positive rate, which made it less prone to false alarms, and RF had a higher true positive rate, which better captured actual attacks. For this DDoS detection challenge, both ensemble tree-based and nearest neighbour methods appear to be effective.

The neural network underperformed likely due to insufficient tuning and training. Its performance may improve with additional hyperparameter optimisation and training data. The modest contribution of autoencoder features indicates that the original inputs already provided useful information that the current RF model could maximise.

The Naive Bayes classifier scored poorly, which can be attributed to the significant relationships between features, which violated its key assumption of feature independence. In order to estimate joint probabilities as the product of individual feature probabilities, Naive Bayes relies on the simple assumption that features are standalone and uncorrelated. When this criterion is not met, which is prevalent in real-world data, Naive Bayes cannot effectively model the genuine joint probability, resulting in inefficient categorisation results.

Despite the limitations of raw Naive Bayes probability outputs for identifying class boundaries, the patterns within its scores provided useful information for distinguishing between normal and abnormal data items. This was demonstrated by the reasonably strong ROC AUC result, which assesses a model's scores' ability to rank positives over negatives regardless of precise probability calibration. Even while the particular classification criteria determined from the probability were untrustworthy, the intrinsic information inside the scores allowed Naive Bayes to assign higher values to more threatening samples. Its ability to rank examples could thus be used for threat prioritisation, despite the fact that inappropriate assumptions make direct categorisation impossible.

Based on the evaluation, the classifiers are ranked as follows for DDoS attack detection effectiveness:

1. Random Forest

2. K-Nearest Neighbours

3. Neural Network recheck the roc score

4. Autoencoder + Random Forest

5. Naïve Bayes

The Random Forest and KNN models provided the optimum balance of accuracy, true positive rate, precision, recall, and low false positive rate. Additional tweaking and training data could potentially increase the neural network's performance. Although Nave Bayes was poor as a classifier, its scoring merits more investigation for ranking and calibration.

# Chapter 6 Conclusion and Future perspectives

In conclusion, the overall objective of this study was to create and test machine learning algorithms for detecting DDoS attacks using the CIC-DDoS2019 dataset. Several categorisation algorithms were constructed and tested, including random forests, KNN, neural networks, autoencoders, and Naive Bayes. On this dataset, the random forest classifier has the highest accuracy (93.46%) and F1-score (0.93). The KNN model likewise performed well, with a 93.46% accuracy and 0.93 F1-score. However, the neural network did not perform as well, demonstrating that complicated deep learning algorithms may not provide advantages over simpler ensemble and instance-based methods for this application. Furthermore, the Naive Bayes classifier showed a substantially lower accuracy of 65.77%, indicating that probabilistic models are not optimal for DDoS detection. The Naive Bayes model, on the other hand, achieved a high ROC AUC score of 0.92. This suggests that, despite its low accuracy, the model was capable of distinguishing between the DDoS and benign groups. The Naive Bayes classifier could be enhanced for this task with extra adjustment of the classification threshold and hyperparameters. The high ROC score indicates that it still has potential if properly optimised.

Based on the CIC-DDoS2019 dataset, the random forest and KNN models show the most promise for identifying DDoS attacks. However the Naïve Bayes classifier with the correct tuning may perhaps be a more viable choice due to the ROC score being high despite low accuracy. The findings imply that ensemble and instance-based techniques can adequately model the network traffic patterns associated with DDoS attacks. Further research with larger and more diversified network traffic datasets, on the other hand, could help corroborate these findings and continue to refine the machine learning algorithms. A critical next step in designing a viable DDoS defence system would be to test the detection system in a live network environment. While this study was limited to pre-recorded datasets, the encouraging results for random forests and KNN show that machine learning can enable automatic detection of DDoS activity in real-time network traffic; these would be the recommended classifications going forward.

### Appendix

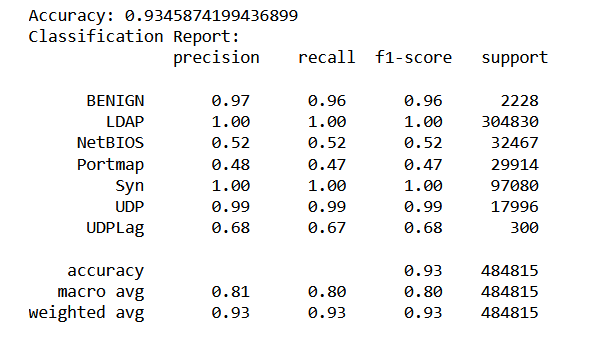


Figure 30: classification report for random forest classifier

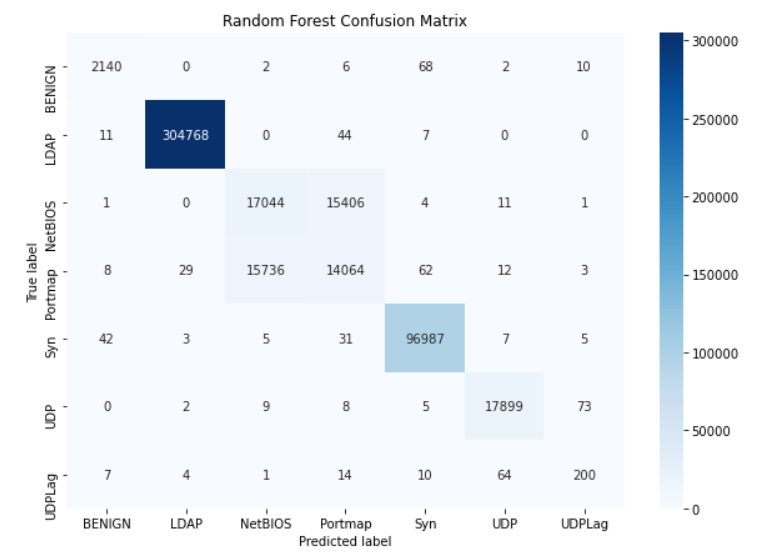


Figure 31: Confusion matrix for random forest classifier

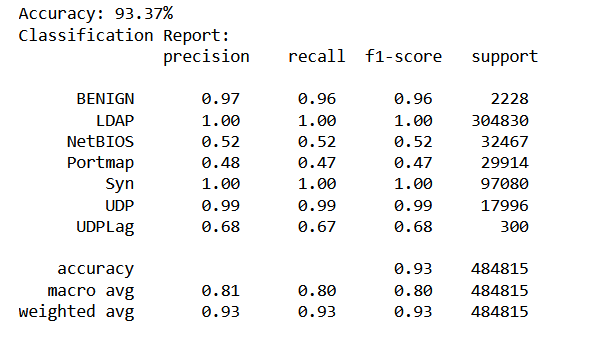


Figure 32: Classification report for K-nearest neighbours

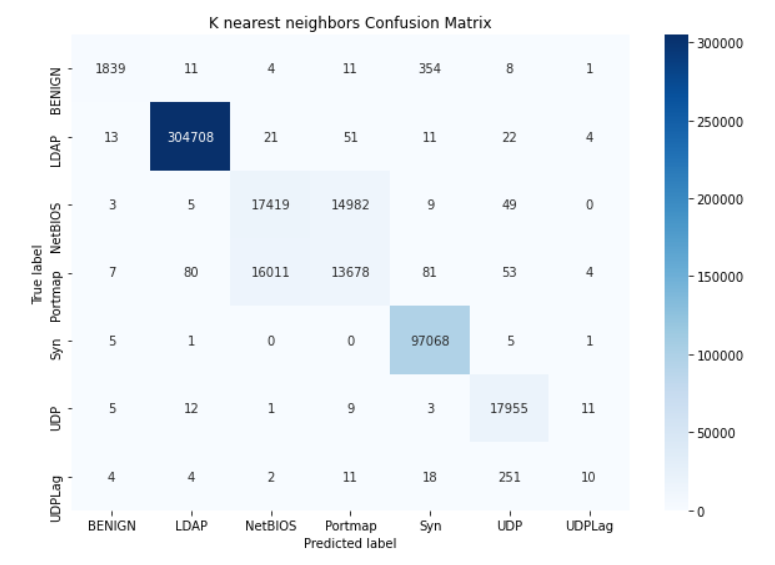


Figure 33: Confusion matrix for k-nearest neighbours

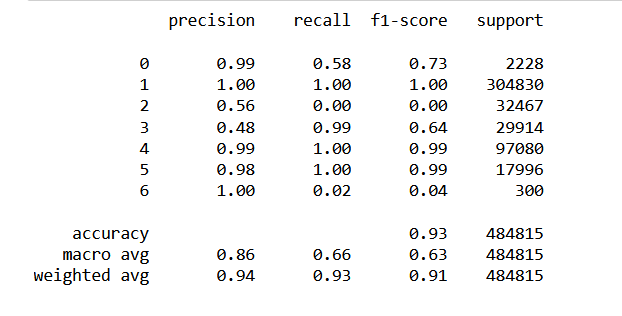


Figure 34: Classification report for neural network classifier

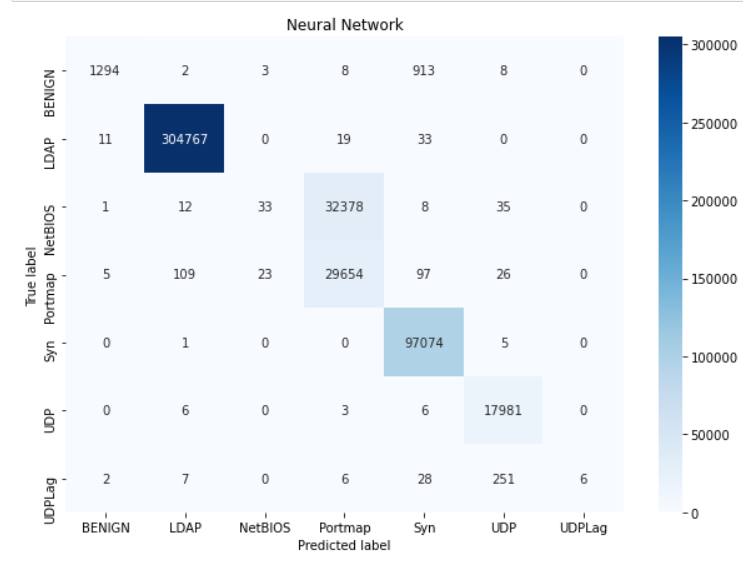


Figure 35: Confusion matric for neural network classifier

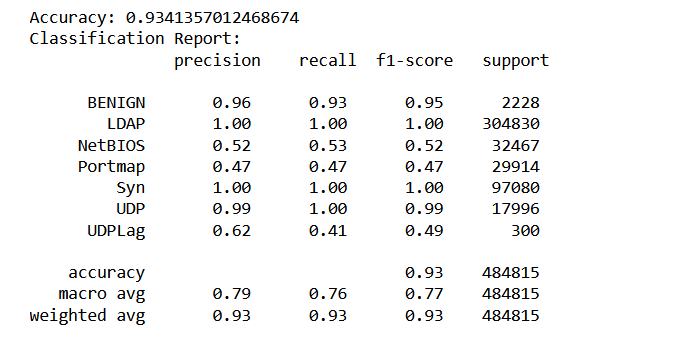


Figure 36: classification report for autoencoder and random forest

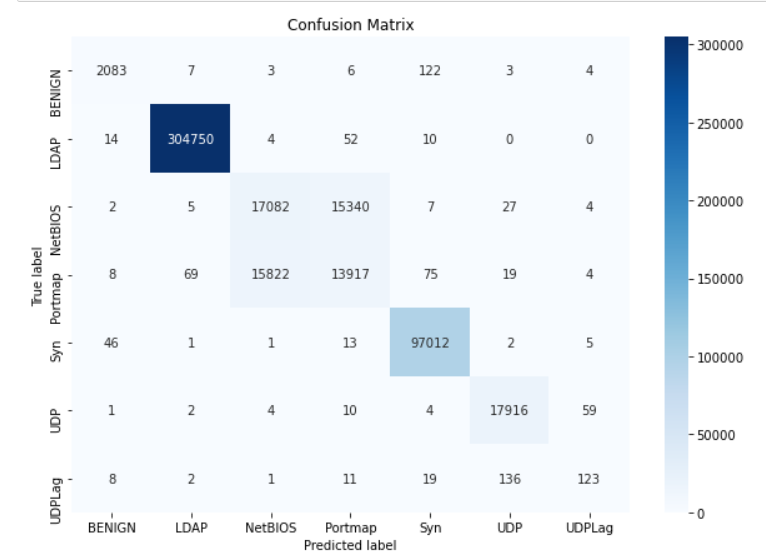


Figure 37: Confusion matrix for autoencoder and random forest

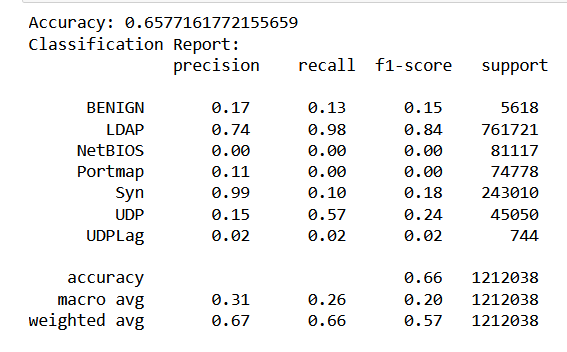


Figure 38: Classification report for Naïve Bayes classifier

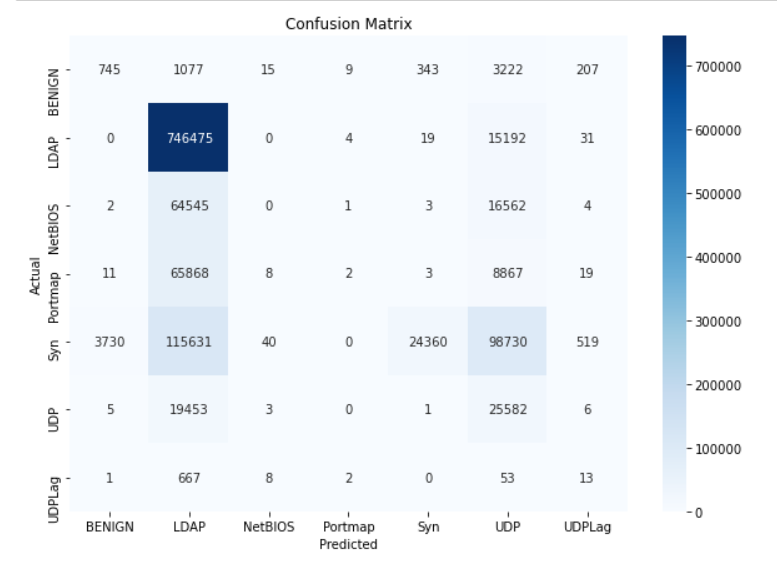


Figure 39: Confusion matrix for Naïve Bayes classifier

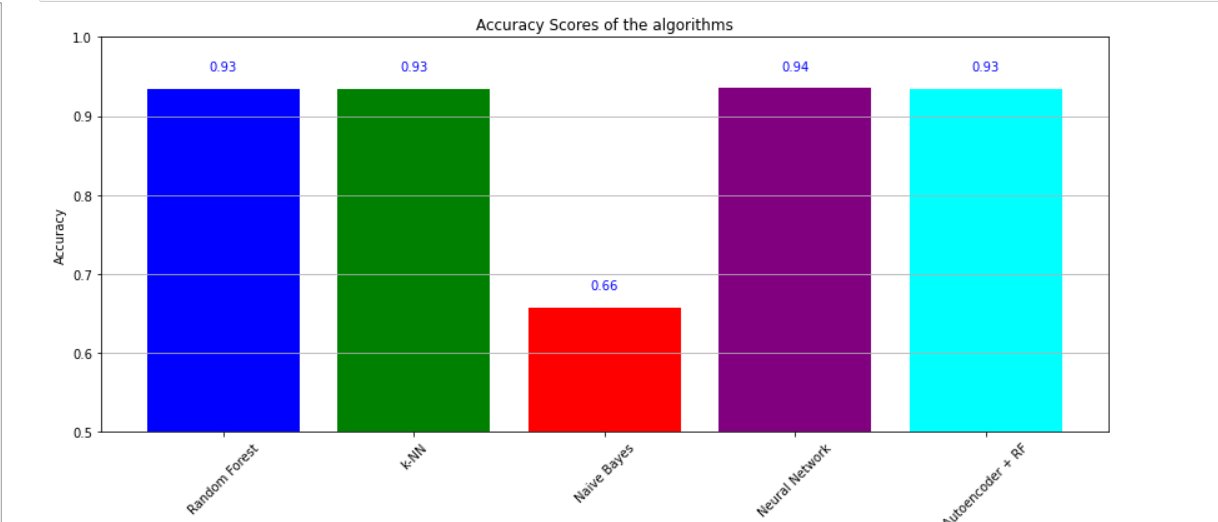


Figure : Bar chart comparing accuracy scores of all classifiers

A graph with different colored squares

Description automatically generated

Figure : Bar chart comparing F1 scores of all classifiers

A graph with red green and purple squares

Description automatically generated

Figure : Bar chart comparing ROC AUC scores of all classifiers

A graph of a bar chart

Description automatically generated with medium confidence

Figure : Bar chart comparing precision of all classifiers

# Bibliography

Aamir, M. and Ali Zaidi, S.M. (2021) ‘Clustering based semi-supervised machine learning for ddos attack classification’, *Journal of King Saud University - Computer and Information Sciences*, 33(4), pp. 436–446. doi:10.1016/j.jksuci.2019.02.003.

Abidoye, A. and Obagbuwa, I. (2018). ddos attacks in wsns: detection and countermeasures. Iet Wireless Sensor Systems, 8(2), 52-59. <https://doi.org/10.1049/iet-wss.2017.0029>

Akanji, O., Abisoye, O., & Iliyasu, M. (2021). Mitigating slow hypertext transfer protocol distributed denial of service attacks in software defined networks. Journal of Information and Communication Technology, 20. https://doi.org/10.32890/jict2021.20.3.1

Aladaileh, M., Anbar, M., Hintaw, A., Hasbullah, I., Bahashwan, A., & Al-Sarawi, S. (2022). Renyi joint entropy-based dynamic threshold approach to detect ddos attacks against sdn controller with various traffic rates. Applied Sciences, 12(12), 6127. <https://doi.org/10.3390/app12126127>

Al-Arnaout, Z., Mostafa, N., Alabed, S., Aly, W., & Shdefat, A. (2022). Rapt: a robust attack path tracing algorithm to mitigate syn-flood ddos cyberattacks. Sensors, 23(1), 102. <https://doi.org/10.3390/s23010102>

Alduailij, M., Khan, Q., Tahir, M., Sardaraz, M., Alduailij, M., & Malik, F. (2022). machine-learning-based ddos attack detection using mutual information and random forest feature importance method. Symmetry, 14(6), 1095. https://doi.org/10.3390/sym14061095

Alimi, K., Ouahada, K., Abu-Mahfouz, A., Rimer, S., & Alimi, O. (2022). Refined lstm based intrusion detection for denial-of-service attack in internet of things. Journal of Sensor and Actuator Networks, 11(3), 32. <https://doi.org/10.3390/jsan11030032>

Aljuhani, A. (2021). Machine learning approaches for combating distributed denial of service attacks in modern networking environments. Ieee Access, 9, 42236-42264. <https://doi.org/10.1109/access.2021.3062909>

Azeez, N., Odufuwa, O., Misra, S., Oluranti, J., & Damaševičius, R. (2021). Windows pe malware detection using ensemble learning. Informatics, 8(1), 10. <https://doi.org/10.3390/informatics8010010>

Balarezo, J.F. *et al.* (2022) ‘A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks’, *Engineering Science and Technology, an International Journal*, 31, p. 101065. doi:10.1016/j.jestch.2021.09.011.

Banitalebi Dehkordi, A., Soltanaghaei, M. and Boroujeni, F.Z. (2020) ‘The ddos attacks detection through machine learning and statistical methods in SDN’, *The Journal of Supercomputing*, 77(3), pp. 2383–2415. doi:10.1007/s11227-020-03323-w.

Batchu, R.K. and Seetha, H. (2021) ‘A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning’, *Computer Networks*, 200, p. 108498. doi:10.1016/j.comnet.2021.108498.

Bhutto, A., Vu, X., Elmroth, E., Tay, W., & Bhuyan, M. (2022). reinforced transformer learning for vsi-ddos detection in edge clouds. Ieee Access, 10, 94677-94690. <https://doi.org/10.1109/access.2022.3204812>

Biau, G. and Scornet, E. (2016). A random forest guided tour. Test, 25(2), 197-227. https://doi.org/10.1007/s11749-016-0481-7

Bii, J.K., Rimiru, R. and Mwangi, R.W. (2021) ‘OAAE: Optimized adaptive anomaly detection ensemble—Base model boosting by parameter optimization’, *Engineering Reports*, 4(2). doi:10.1002/eng2.12449.

Bijalwan, A., Wazid, M., Pilli, E., & Joshi, R. (2015). Forensics of random-udp flooding attacks. Journal of Networks, 10(5). <https://doi.org/10.4304/jnw.10.5.287-293>

Bougueroua, N., Guessoum, Z., Belaoued, M., Seddari, N., Derhab, A., & Bouras, A. (2021). A survey on multi-agent based collaborative intrusion detection systems. Journal of Artificial Intelligence and Soft Computing Research, 11(2), 111-142. https://doi.org/10.2478/jaiscr-2021-0008

Bouyeddou, B., Harrou, F., Kadri, B., & Ma, J. (2020). Detecting network cyber-attacks using an integrated statistical approach. Cluster Computing, 24(2), 1435-1453. <https://doi.org/10.1007/s10586-020-03203-1>

Cummings, C. (2023). Letter regarding “developing a predictive model for spinal shock in dogs with spinal cord injury”. Journal of Veterinary Internal Medicine, 37(2), 400-401. https://doi.org/10.1111/jvim.16631

Dasari, K.B. and Devarakonda, N. (2022) ‘Detection of ddos attacks using machine learning classification algorithms’, *International Journal of Computer Network and Information Security*, 14(6), pp. 89–97. doi:10.5815/ijcnis.2022.06.07.

Dawadi, B., Adhikari, B., & Srivastava, D. (2023). Deep learning technique-enabled web application firewall for the detection of web attacks. Sensors, 23(4), 2073. https://doi.org/10.3390/s23042073

Dayal, N. and Srivastava, S. (2023) ‘Analyzing effective mitigation of ddos attack with software defined networking’, *Computers &amp; Security*, 130, p. 103269. doi:10.1016/j.cose.2023.103269.

Dong, S. and Sarem, M. (2020). Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks. Ieee Access, 8, 5039-5048. <https://doi.org/10.1109/access.2019.2963077>

Efe, A. and Abaci, İ. (2022). Comparison of the host based intrusion detection systems and network based intrusion detection systems. Celal Bayar Üniversitesi Fen Bilimleri Dergisi, 18(1), 23-32. <https://doi.org/10.18466/cbayarfbe.832533>

El-Sofany, H. (2020). A new cybersecurity approach for protecting cloud services against ddos attacks. International Journal of Intelligent Engineering and Systems, 13(2), 205-215. https://doi.org/10.22266/ijies2020.0430.20

Gebremeskel, T.G. *et al.* (2023) ‘DDoS Attack Detection and Classification Using Hybrid Model for Multicontroller SDN’, *Wireless Communications and Mobile Computing*, 2023, pp. 1–18. doi:10.1155/2023/9965945.

Fachantidis, A., Taylor, M., & Vlahavas, I. (2017). learning to teach reinforcement learning agents. Machine learning and Knowledge Extraction, 1(1), 21-42. https://doi.org/10.3390/make1010002

Géron, A. (2022) *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. Sebastapol, CA: O’Reilly.

Gupta, S. and Vanjale, S. (2020). Cyber security measures for internet of things devices. International Journal of Engineering Research and Technology, 13(8), 1830. <https://doi.org/10.37624/ijert/13.8.2020.1830-1839>

Gwetu, M., Tapamo, J., & Viriri, S. (2020). random forests with a steepend gini-index split function and feature coherence injection., 255-272. https://doi.org/10.1007/978-3-030-45778-5\_17

Hai, T., Khiem, N., & Phuc, N. (2021). Toward an online dos/ddos classification: an empirical study for network intrusion detection systems. Journal of Computer Science, 17(3), 304-318. <https://doi.org/10.3844/jcssp.2021.304.318>

Halladay, J., Cullen, D., Briner, N., Warren, J., Fye, K., Basnet, R., … & Doleck, T. (2022). detection and characterization of ddos attacks using time-based features. Ieee Access, 10, 49794-49807. https://doi.org/10.1109/access.2022.3173319

Hesselman, C., Grosso, P., Holz, R., Kuipers, F., Xue, J., Jonker, M., … & Laat, C. (2020). A responsible internet to increase trust in the digital world. Journal of Network and Systems Management, 28(4), 882-922. https://doi.org/10.1007/s10922-020-09564-7

Home Office. (1990). Computer Misuse Act 1990. Retrieved from <https://www.legislation.gov.uk/ukpga/1990/18>

Jeitner, P., Shulman, H., & Waidner, M. (2020). The impact of dns insecurity on time.. <https://doi.org/10.1109/dsn48063.2020.00043>

Katuk, N. (2023). An observational mechanism for detection of distributed denial-of-service attacks. International Journal of Advances in Applied Sciences, 12(2), 121. <https://doi.org/10.11591/ijaas.v12.i2.pp121-132>

Kindermans, P Kindermans et al. (2017). Learning how to explain neural networks: patternnet and patternattribution.. https://doi.org/10.48550/arxiv.1705.05598

Kopp, D., Dietzel, C., & Hohlfeld, O. (2021). ddos never dies? an ixp perspective on ddos amplification attacks., 284-301. <https://doi.org/10.1007/978-3-030-72582-2_17>

Lazar, D., Cohen, K., Freund, A., Bartik, A., & Ron, A. (2021). Imdoc: identification of malicious domain campaigns via dns and communicating files. Ieee Access, 9, 45242-45258. <https://doi.org/10.1109/access.2021.3066957>

Lye, K., Mishra, S., & Ray, D. (2020). deep learning observables in computational fluid dynamics. Journal of Computational Physics, 410, 109339. https://doi.org/10.1016/j.jcp.2020.109339

Li, Y., Zhao, Y., Li, J., Yu, X., Zhao, Y., & Zhang, J. (2021). ddos attack mitigation based on traffic scheduling in edge computing- enabled twdm-pon. Ieee Access, 9, 166566-166578. <https://doi.org/10.1109/access.2021.3134671>

Ling, M. (2023). Machine-learning-based network sparsification modeling for iots security analysis.. <https://doi.org/10.1117/12.2690061>

Maier, A. *et al.* (2022) ‘Known operator learning and hybrid machine learning in medical imaging—a review of the past, the present, and the future’, *Progress in Biomedical Engineering*, 4(2), p. 022002. doi:10.1088/2516-1091/ac5b13.

Marvi, M., Arfeen, A. and Uddin, R. (2020) ‘A generalized machine learning‐based model for the detection of ddos attacks’, *International Journal of Network Management*, 31(6). doi:10.1002/nem.2152.

Mamolar, A. S., Pervez, Z., Calero, J. M. A., & Khattak, A. M. (2018). Towards the transversal detection of ddos network attacks in 5g multi-tenant overlay networks. Computers &Amp; Security, 79, 132-147. <https://doi.org/10.1016/j.cose.2018.07.017>

Maslan, A., Mohamad, K., & Foozy, F. (2020). Feature selection for ddos detection using classification machine learning techniques. Iaes International Journal of Artificial Intelligence (Ij-Ai), 9(1), 137. <https://doi.org/10.11591/ijai.v9.i1.pp137-145>

Mishra, N. and Singh, R. (2022). detection of ddos vulnerability in cloud computing using the perplexed bayes classifier. Computational Intelligence and Neuroscience, 2022, 1-13. https://doi.org/10.1155/2022/9151847

Naiem, S., Khedr, A., Idrees, A., & Marie, M. (2023). Iterative feature selection-based ddos attack prevention approach in cloud. International Journal of Electrical and Computer Engineering Systems, 14(2), 197-205. https://doi.org/10.32985/ijeces.14.2.9

Najafimehr, M., Zarifzadeh, S. and Mostafavi, S. (2022) ‘A hybrid machine learning approach for detecting unprecedented ddos attacks’, *The Journal of Supercomputing*, 78(6), pp. 8106–8136. doi:10.1007/s11227-021-04253-x.

Najafimehr, M., Zarifzadeh, S. and Mostafavi, S. (2023) ‘DDoS attacks and machine‐learning‐based detection methods: A survey and taxonomy’, *Engineering Reports* [Preprint]. doi:10.1002/eng2.12697.

Naveen Bindra and Manu Sood (2019) ‘Detecting ddos attacks using machine learning techniques and contemporary Intrusion detection dataset’, *Automatic Control and Computer Sciences*, 53(5), pp. 419–428. doi:10.3103/s0146411619050043.

Nguyen, D., Le, M., & Cung, T. (2023). The ability to detect the linear attack of wl-cusum and fma algorithms. Bulletin of Electrical Engineering and Informatics, 12(1), 131-140. https://doi.org/10.11591/eei.v12i1.4400

Piergiorgio, L., Martinez, M., & Barais, O. (2022). Risk explorer for software supply chains.. https://doi.org/10.1145/3560835.3564546

Pranggono, B. (2015). Impact of distributed denial-of-service attack on advanced metering infrastructure. Wireless Personal Communications, 83(3), 2211-2223. <https://doi.org/10.1007/s11277-015-2510-3>

Sabrina, F., Jang-Jaccard, J., Xu, W., & Wei, Y. (2021). Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems. Sensors, 22(1), 32. <https://doi.org/10.3390/s22010032>

Sayed, M.S. *et al.* (2022) ‘A flow-based anomaly detection approach with feature selection method against ddos attacks in Sdns’, *IEEE Transactions on Cognitive Communications and Networking*, 8(4), pp. 1862–1880. doi:10.1109/tccn.2022.3186331.

Sayed, S., El-Korany, A., & Mohammad, S. (2021). Applying different machine learning techniques for prediction of covid-19 severity. Ieee Access, 9, 135697-135707. https://doi.org/10.1109/access.2021.3116067

Sharafaldin, I., Habibi Lashkari, A. and Ghorbani, A.A. (2019) ‘Toward generating a new intrusion detection dataset and intrusion traffic characterization’, *Proceedings of the 4th International Conference on Information Systems Security and Privacy* [Preprint]. doi:10.5220/0006639801080116.

Softić, J. and Vejzović, Z. (2022). operating systems vulnerability – an examination of windows 10, macos, and ubuntu from 2015 to 2021. Advances in Science Technology and Engineering Systems Journal, 7(6), 230-235. https://doi.org/10.25046/aj070625

Srinivasan, K., Mubarakali, A., Alqahtani, A., & Kumar, A. (2019). A survey on the impact of ddos attacks in cloud computing: prevention, detection and mitigation techniques., 252-270. <https://doi.org/10.1007/978-3-030-28364-3_24>

Suleman, M. (2021) ‘A Machine Learning-Based Intrusion Detection of DDoS Attack on IoT Devices’, *International Journal of Advanced Trends in Computer Science and Engineering*, 10(4), pp. 2792–2797. doi:10.30534/ijatcse/2021/221042021.

Tennakoon, C. and Fernando, S. (2022). Deep learning model for distributed denial of service (ddos) detection. International Journal of Advanced and Applied Sciences, 9(2), 109-118. https://doi.org/10.21833/ijaas.2022.02.012

Tran, M., Choi, I., Moon, G., Vu, A., & Kang, M. (2020). A stealthier partitioning attack against bitcoin peer-to-peer network.. https://doi.org/10.1109/sp40000.2020.00027

Ujjan, R.M. *et al.* (2020) ‘Towards sflow and adaptive polling sampling for deep learning based ddos detection in SDN’, *Future Generation Computer Systems*, 111, pp. 763–779. doi:10.1016/j.future.2019.10.015.

Ulemale, T. (2022). Review on detection of ddos attack using machine learning. International Journal for Research in Applied Science and Engineering Technology, 10(3), 764-768. <https://doi.org/10.22214/ijraset.2022.40742>

Wassermann, S., Cuvelier, T., Mulinka, P., & Casas, P. (2019). adam &amp; ral: adaptive memory learning and reinforcement active learning for network monitoring.. https://doi.org/10.23919/cnsm46954.2019.9012675

Yuen, B., Hoang, M., Dong, X., & Lu, T. (2021). Universal activation function for machine learning. Scientific Reports, 11(1). https://doi.org/10.1038/s41598-021-96723-8