Faculty of Media Engineering and Technology
German University in Cairo

An Intelligent Cloud Computing Security Model

Bachelor Thesis

by

Omar Mohamed Farouk Mohamed Mansour

Supervised by

Dr. Gamal Abdel Shafy

May 2025

Declaration

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor of Science (B.Sc.) at the German University in Cairo (GUC),

(ii) due acknowledgement has been made in the text to all other material used

---

Name

Omar Mohamed Farouk Mohamed Mansour Date

29/5/2025

Acknowledgments

I would like to express my special thanks of gratitude to my supervisor Dr. Gamal Abdel-Shafy who gave me the golden opportunity to do this wonderful project on the topic (An Intelligent Cloud Computing Security Model), which also helped me in doing a lot of Research and i came to know about so many new things. Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Contents

List of Figures

List of Tables

List Of Acronyms

AI Artificial Intelligence

CNN Convolutional Neural Network

DDoS Distributed Denial of Service

DoS Denial of Service

IDS Intrusion Detection System

LSTM Long Short-Term Memory

ML Machine Learning

RF Random Forest

IaaS Infrastructure as a Service

PaaS Platform as a Service

SaaS Software as a Service

VAE Variational Autoencoders

GAN Generative Adversarial Networks

RBAC Role-Based Access Control

IDS Intrusion Detection System

DoS Denial of Service

DDos Distributed Denial of Service

Abstract

Machine Learning and Deep Learning have become a common practice for classification tasks because they learn patterns from data and delievers high accurate predictions. this makes it a great potential tool for enhancing cloud security, especialy in threat detecton.In this project, I present potential intelligent threat detection models for cloud environments using combinations of Decision trees, Random forests, Convolutional neural networks, Long Short Term Memory and Auto Keras models. My goal is to build models that detects diffrent types of popular cyber threats.

Most studies use only one single type of model for threat detection, which reduces the potential of handling diverse attack patterns effectively,by using many types of models in my project,we increase the potential of increasing the performance and robustness of threat detection compared to prior work.

The results of my models show improvement in some threat types over some other articles. My best models for DoS, Infiltration, and Web Attacks yielded significant increase in accuracy compared to other models.

For DoS attacks, our top-performing model achieved 99.2% accuracy, outperforming the previously reported 98.5%. In the case of Infiltration, we reached 99.8%, compared to an earlier result of 94.4%. For Web Attacks, our model obtained 99.4% accuracy, exceeding the previously reported 94.4%. These results demonstrate the effectiveness of our approach in detecting these specific threat types more accurately than prior methods. For the remaining threat types, the difference in performance was minimal, with accuracy improvements not exceeding 0.5% compared to other reported models.

Keywords: Cloud security, machine learning, threat detection, CIC IDS 2017, Random Forest, LSTM, network threats

Chapter 1

Introduction

This chapter introduces an overview of the thesis by showing the main probelm being addressed, the motivation of choosing this project,the main objectives and goals directing the research and the overall structure of the thesis. Cloud computing services have become increasingly relied upon due to their scalability, flexibility, and cost-efficiency.. However, it also introduced challenging security issues due to the changing nature of cloud environments. Cyber threats targeting cloud infrastructures are becoming more advanced more popular. This project focuses on enhancing machine learning and deep learning techniques to boost cloud security by developing a high performing intrusion detection models capable of accurately classifying various types of cyberattacks. By combining multiple machine learning models including Decision Trees, Random Forest, Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), and automated architecture search through AutoKeras—the research aims to improve detection performance and adaptability to evolving threats.

1.1 Problem Statement

As cloud computing continues to become more known across the globe, the evolution of cyber threats has become increasingly sophisticated and aggressive. Normal and traditional security approaches depends heavily on human monitoring and manual interference, which unfortunately, are no longer efficient enough to protect the enormous and always changing nature of modern cloud environments. Humans face limitations in terms of response time, scalability, and consistency,which makes it difficult to detect

and respond to threats instantaneously in real-time. This creates a growing need for advanced, intelligent tools and non human tools that can assist in safeguarding cloud systems.

1.2 Project Motivation

Machine Learning and Deep Learning have become a common practice for classification tasks because they learn patterns from data and delievers high accurate predictions. this makes it a great potential tool for enhancing cloud security.in addition some of their capabilties are:

- Instantaneous threat detection by analyzing data and identifying anomalies in real-time.
- Scalability across large, enormous infrastructures where manual human supervision isn't efficient enough.
- Continuous learning and adaptation to new and unseen threats, enhancing overall cloud security .
- Integration with advanced encryption techniques such as homomorphic encryption, which enhances data privacy and is highly complex for human implementation.[1]

1.3 Project Objectives

This project aims to identify machine learning models that provide highly reliable accuracy in threat detection, with particular emphasis on minimizing false positives and false negatives.

1.4 Thesis Organization

This thesis is organized into the following chapters:

- Chapter 1: Introduction — Introduces the problem statement, project motivation, objectives, and provides an overview of the thesis structure.
- Chapter 2: Background — Reviews existing research on cloud security and machine learning techniques, highlighting common limitations and key findings in the field.
- Chapter 3: Methodology — Describes the dataset, machine learning models used, and experimental setup.
- Chapter 4: Results — Presents and analyzes the performance results of the models on various threat detection tasks.
- Chapter 5: Conclusion — Discusses the implications of the results, limitations, and potential improvements.

Chapter 2

Background

Cloud computing security is one of the main great aspects found in our modern IT infrastructure. This chapter introduces a comprehensive review of existing research and articles related to our project topic(Intelligent Cloud Computing Security Model), followed by an explanation of triggers which is a crucial aspect in applying our models in practicle world tasks and the steps taken to develop the proposed solution.

2.1 Cloud Computing

This section focuses on the main concepts in cloud computing in general, including its service models, architectural patterns, and key features that makes modern scalable applications.It also highlights common challenges in managing and securing cloud environments.

2.1.1 Infrastructure as a Service

IaaS provides virtualized computing resources over the internet. Users manage everything like their applications, data, and runtime environments while the cloud provider just handles networking, storage, and virtualization. This model offers decent flexibility but needs who understand how to carefully manage resources[2]. An example of some of the services found in IaaS model is illustrated in Figure 2.1.

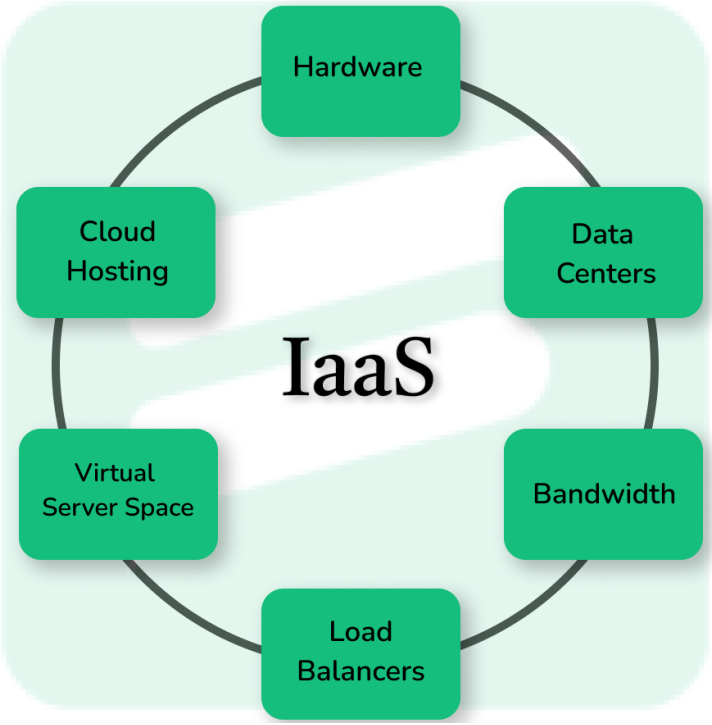

Figure 2.1: Infrastructure as a Service model

2.1.2 Platform as a Service

PaaS simplifies application development by handling all the infrastructure concerns. It offers tools, libraries, and frameworks to make development and deployment, improving productivity while maintaining scalability[3]. A representation of the PaaS model services is shown in Figure 2.2.

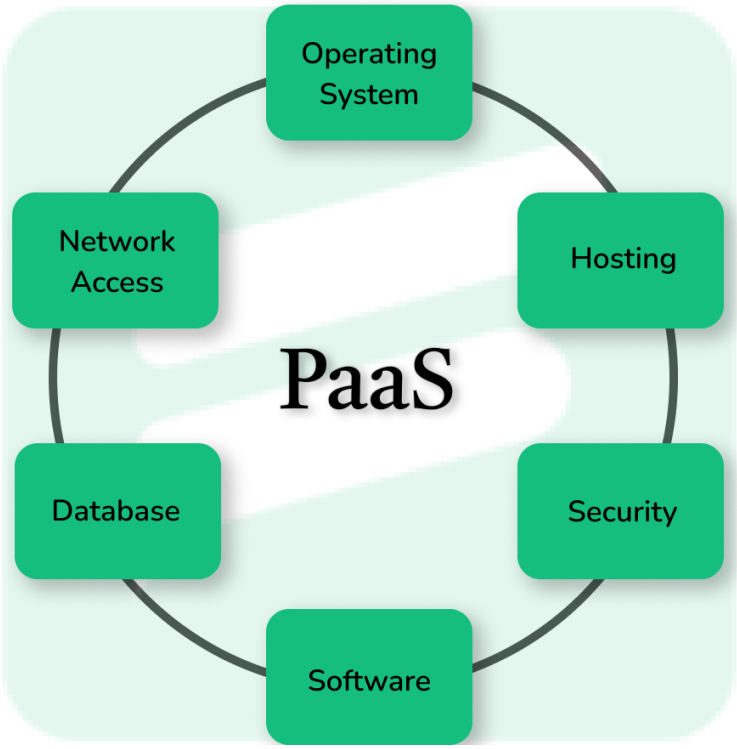

Figure 2.2: Platform as a Service model

2.1.3 Software as a Service

SaaS delivers software applications over the internet, eliminating the need for local installations. Users interact with applications via web browsers while the service provider handles maintenance, updates, and infrastructure[4] .Figure 2.3 illustrates SaaS services

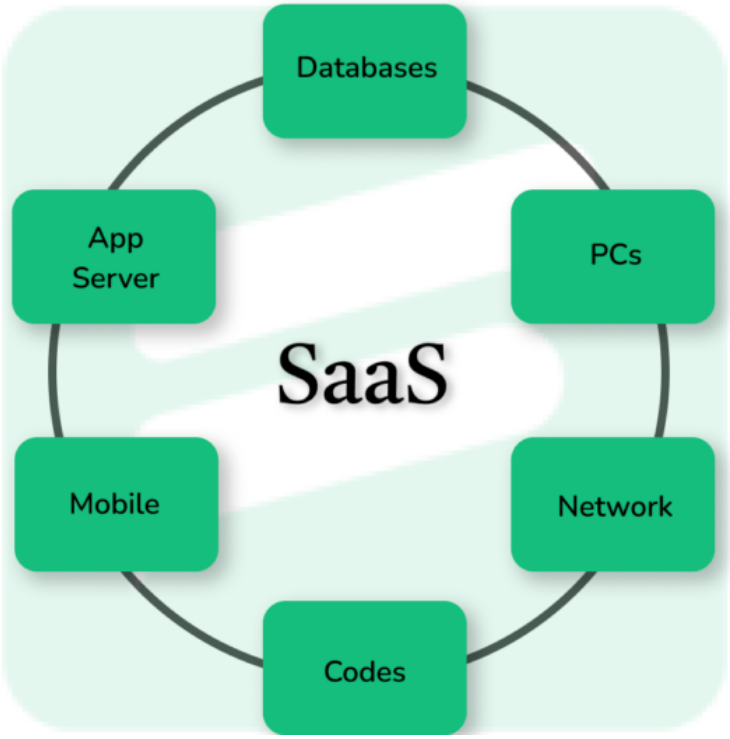

Figure 2.3: Software as a Service model

Additionally, a comparative illustration of IaaS, PaaS, and SaaS models is presented in Figure 2.4, highlighting the distinctions and responsibilities in each service model.

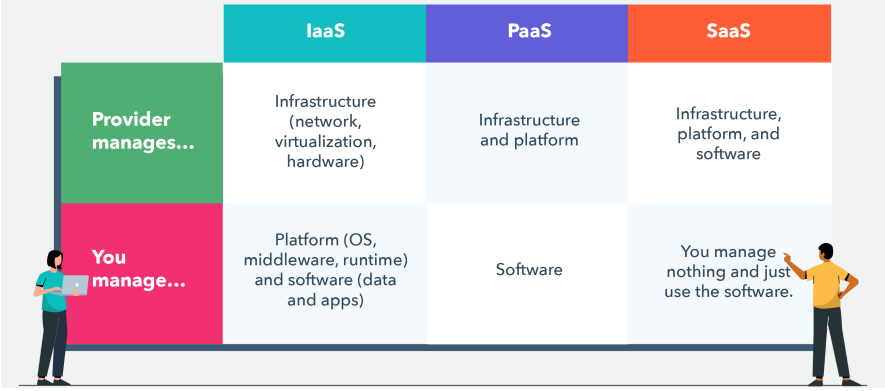

Figure 2.4: Comparison of IaaS, PaaS, and SaaS models

2.1.4 Serverless Computing

Serverless computing abstracts infrastructure management entirely to allow developers to focus on writing and deploying code only. Serverless platforms adapts the scaling

based on incoming requests, ensuring efficient resource utilization[5]. A figure illustrating the advantages and disadvantages of serverless computing will be presented in Figure 2.5.

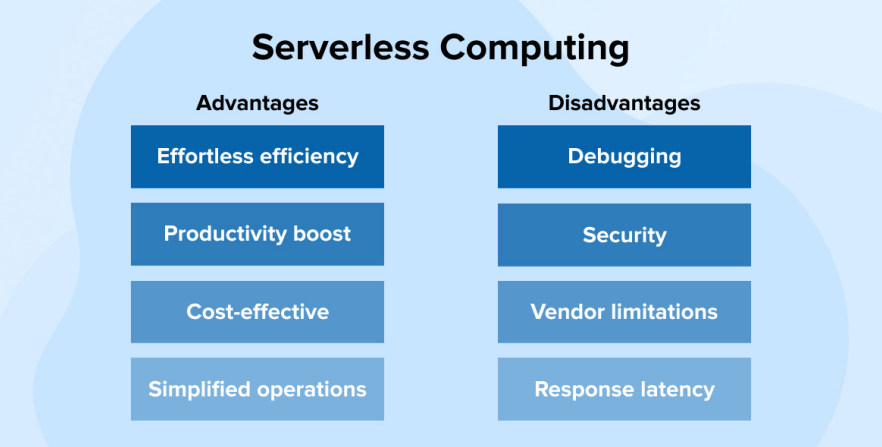

Figure 2.5: Advantages and disadvantages of Serverless Computing

2.1.5 Event-Driven Architecture

EDA enables services to communicate through event triggers using a publisher-subscriber model. This architecture boosts scalability by decoupling components to allow independent scaling and improved fault tolerance.

2.1.6 Auto-Scaling Capabilities

Cloud platforms support auto-scaling and adjusting resource allocation dynamically based on workload demands. This feature enhances performance, reducing costs, and make sure that high availability during traffic surges are found. A figure illustrating the working mechanism of auto-scaling will be shown in Figure 2.6.

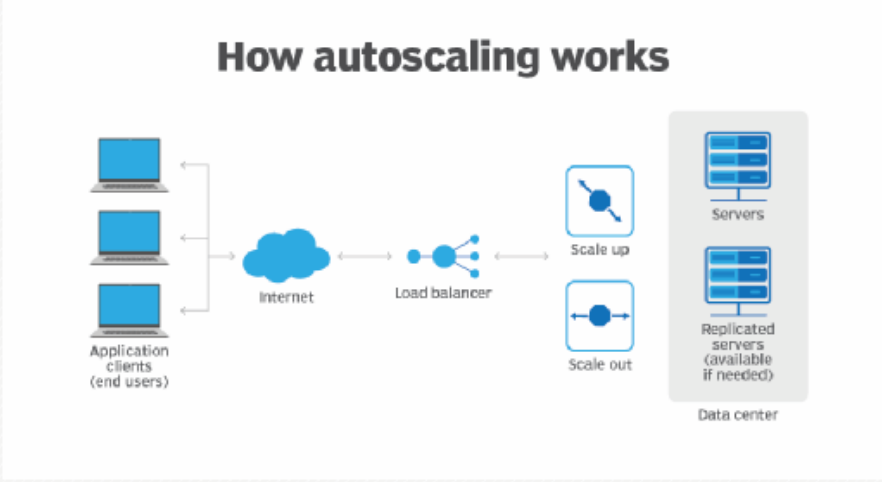

Figure 2.6: Overview of Auto-Scaling Mechanism

By combining these elements, cloud computing enables developers to build resilient, high-performance applications capable of adapting to fluctuating workloads and evolving business needs.

2.2 Challenges in Cloud Security

Cloud security is challenging because it introduces unique complexities that go beyond traditional human on-premise security models. Some reasons why cloud security is considered difficult include:

2.2.1 Shared Responsibility Model

Cloud providers (e.g., AWS, Azure) secure the infrastructure, but you are responsible for securing your data, applications, and access control. Misunderstanding this concept may and will lead to vulnerabilities.

2.2.2 Dynamic and Evolving Environment

Cloud resources are highly changing — virtual machines, containers, and microservices can scale up or down Immediately. This constant change requires automated security strategies to track and manage evolving risks[6].

2.2.3 Data Exposure and Leakage

Data stored in the cloud is commonly spread across many regions and servers. Misconfigured storage services (e.g., AWS S3 buckets left public) can expose sensitive information[7].

2.2.4 Identity and Access Management (IAM) Complexity

Managing permissions across multiple services, environments, and user roles is complicated. Overly permissive roles or misconfigured access controls are common security gaps.

2.2.5 API Vulnerabilities

Cloud environments depend heavily on APIs for communication and networking. Unsecured APIs or excessive permissions can provide attackers with easy entry points.[8]

2.2.6 Compliance and Legal Risks

Different countries have varying data privacy laws (e.g., GDPR, HIPAA). Ensuring data residency, encryption standards, and access control mechanisms meet these regulations can be complex.

2.2.7 Insider Threats

Cloud environments make it easier for employees, contractors, or third parties to access systems remotely. Insider attacks or accidental data leaks are increased risks.

2.2.8 Limited Visibility and Monitoring

Traditional security tools may not provide full visibility into cloud environments. Specialized cloud-native monitoring tools are often required to track network activity, data flows, and security events[9].

2.2.9 Supply Chain Risks

Cloud services frequently integrate third-party tools, libraries, and SaaS platforms. If these dependencies have vulnerabilities, your cloud environment becomes exposed.

2.2.10 DDoS and Account Hijacking

Cloud environments are frequent targets for Distributed Denial of Service (DDoS) attacks. Additionally, stolen credentials can allow attackers to infiltrate accounts and manipulate cloud resources[10]. Figure 2.7 may illustrate how a DDoS attack appears from an external perspective, showing the flood of malicious traffic targeting the cloud infrastructure.

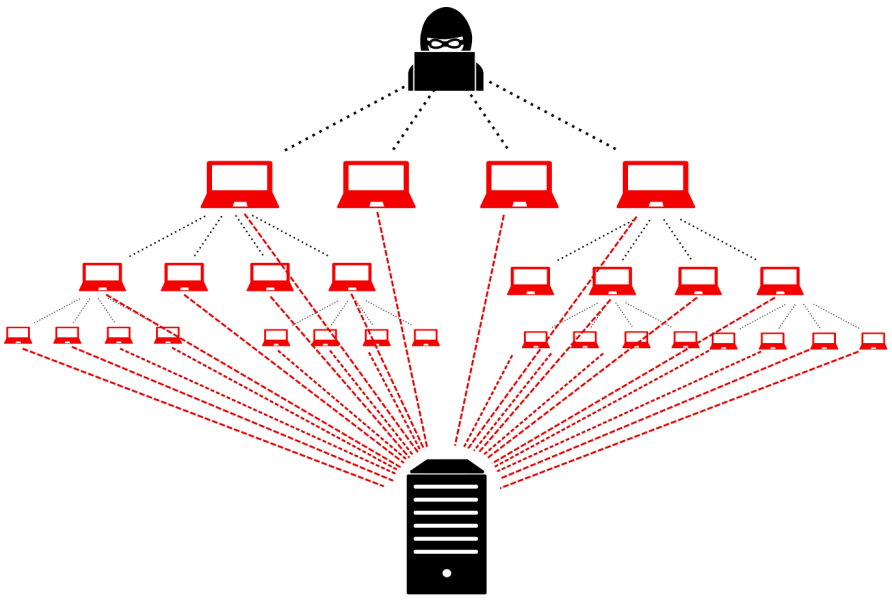

Figure 2.7: Example Simulating a DDoS Attack on Cloud Infrastructure

2.3 Literature Review

This section introduces a review of the literature on research related to my topic focusing on threat detection solutions using machine learning and deep learning techniques. The reviewed articles are divided into two main groups: conventional methods and AI-based models that mainly use machine learning and deep learning techniques. This classification shows clear overview of Traditiona approaches and new machine learning and deep learning methods to evaluate my proposed models.

2.3.1 Conventional Security Models

Conventional security models have always been widely used in cloud computing environments to eliminate security threats. These methods mainly depend on widely used and reliable techniques such as firewalls, role-based access controls (RBAC)[11], and signature-based intrusion detection systems (IDS)[12]. While they are effective in instant threat detection, they always struggle with new and evolving cyber threats and Sophisticated attack patterns.

Previous studies[11][12] indicate that firewalls and signature-based intrusion detection systems remain effective for immediate threat detection; however, they often lack adaptability against new and evolving attack patterns. Role-based access control has proven useful in managing user permissions, though it requires manual configuration. Conventional security systems also tend to be less resource-intensive, offering faster real-time responses when compared to AI-based methods

Limitations

1. Conventional models are less effective against zero-day attacks and advanced persistent threats (APTs).

2. They require frequent manual updates to stay effective.

3. Conventional security lacks predicting of proactive threat prevention.

2.3.2 AI-Based Security Models

2.3.2.1 AI-Driven User Behavior Analysis

The study by Olabanji et al. (2024) investigates the benefits of AI-driven user behavior analysis compared to traditional security measures in cloud computing. The research takesa quantitativee approach by surveying cybersecurity professionals from industries like finance, healthcare, IT, and government. Data analysis using multiple regression techniques proved that AI-driven systems can actually imporve threat detection accuracy and prediction capabilities[13].

AI-driven security models have shown that they enhance the overall accuracy of predictions of threats, which enables more efficient identification of new evolving threats. On the other hand,Normal and Traditional security methods still outperforms AI model when it comes to instant response time having quicker reaction to known attacks,which makes it one of the main reasons that many researchers recommend developing a hybrid framework that integerate the strength of both sides to achieve balanced performance and accuracy.

Limitations

1. AI models depends heavily on the quality of data,which will have the potential of introducing biases.

2. False positives and negatives require human validation.

3. AI costs higher than the traditional security methods which may lead to pose some financial challenges for some organizations.

2.3.2.2 Enhancing Cloud Security with Generative Models for Real-Time Anomaly Detection

This study explores the potential applications of AI models sepcificaly VANs, GANS and Transformers, in boosting the performance of threat detection in cloud computing. It mainly shows how these models performs better against traditional RBAC and IDSby enabling real time anomly detection and dynamic learning of normal and malicious behaviours.This research also mentioned the benefits over traditional methods,implementaion strategies and challenges faced when developing the AI in cloud environments[14].

AI-Based soloutions supports the potential of dynamic threat response which allows identifcation and eliminating security risks.GANS specialy presents a important role by enhancing the recognition of patterns and enabling early detection of potential threats.AI models also presents adaptive learning techniques in recognizing and responding to complex and evolving cyberattacks, which cant be achieved by Traditional security methods.

Limitations

1. Difficulty in model interpretability hinders trust and transparency.
2. Susceptibility to adversarial attacks may reduce robustness.
3. Challenges include high computational overhead and regulatory compliance issues.

2.3.2.3 AI-Driven Threat Detection: A Brief Overview of AI Techniques in Cybersecurity

This study investigates the increasing role of AI in boosting the defense of cybersecurity techniques in the response to the increasing frequent and barely countered cyber threats.It shows how machine learning and deep learning are used in real-time threat detection,predictive analysis, endpoint protection, and intrusion detection. This article

also demonstrates how machine learning and deep learning are used in real-time threat detection,predictive analysis, endpoint protection, and intrusion detection. This paper also invistigates the challenges faced during AI models implementation like data quality,model complexity and ethical considered actions like privacy and bias,while showcasing the potential of technologies like explainable AI and AI-human collaboration to shape the future of cybersecurity[15].

AI has notably improved the ability to detect and respond to cyber threats in real-time, enhancing the overall effectiveness of cybersecurity measures. Its applications span across predictive analytics, intrusion detection, and endpoint security, providing comprehensive protection at multiple levels. Additionally, emerging trends like explainable AI and AI-human collaboration are poised to further advance cybersecurity capabilities by increasing transparency and fostering more effective interactions between automated systems and human experts.

Limitations

1. AI effectiveness depends heavily on data quality and availability.
2. Implementation involves complex integration and high computational cost.
3. Ethical concerns include privacy, bias, and the potential for adversarial misuse.

2.3.2.4 AI-Enhanced Cloud Computing for Resource Management and Security

The study by Bhattarai and Singh (2024) investigates the role of AI-driven techniques in cloud computing, specifically in resource management, fault tolerance, and security . The research explores methods such as machine learning, deep learning, and heuristic algorithms to optimize cloud performance, improve resource allocation, and enhance threat detection[16].

AI models increased both historical and real-time data to accurately predict workloads, optimize resource allocation, and enhance the detection of security threats. Tech-

niques such as machine learning, deep learning, and heuristic algorithms have been shown to effectively improve cloud security while also boosting overall system performance and efficiency.

Limitations

1. AI-driven cloud systems face challenges related to data quality issues and implementation complexity.

2. These systems are vulnerable to adversarial attacks, requiring constant monitoring and updates.

3. Heavy reliance on historical data may introduce biases, potentially affecting decision-making and system reliability.

2.3.2.5 AI-Driven Threat Detection in Cloud Security

The study by Nina and Patel (2019) investigates the transformative role of artificial intelligence (AI) in enhancing threat detection and response mechanisms within cloud security frameworks . The research highlights how AI-driven technologies such as machine learning, anomaly detection, and predictive modeling can significantly improve the identification of cyber threats, outperforming traditional security methods. The study also emphasizes the integration of AI with existing security tools to create a proactive security posture[17].

AI technologies such as machine learning, anomaly detection, and predictive modeling play a crucial role in identifying and mitigating cyber threats within cloud environments. By leveraging real-time analytics and behavioral analysis, cloud security systems are empowered to detect evolving threats promptly and deliver automated responses. Numerous successful case studies further demonstrate the effectiveness of AI-driven systems in enhancing both cloud security and incident response capabilities

Limitations

1. AI-driven threat detection systems may struggle with false positives and data quality issues, impacting system accuracy and response times.

2. These systems are susceptible to adversarial attacks, which may circumvent detection and require continuous updates.

3. Integration of AI with existing security tools can be complex, requiring specialized knowledge and additional resources.

2.4 Contextual Analysis

The collected studies showcase the rising dependency on AI-driven soloutions to enhance cloud security. While most AI-based soloutions imporves preiction,detection,acccuracy and response. Conventional-based models still prove to be better in immediate threat detection and cost efficiency. which recommends integrating both strategies through a hybrid-based system to offer balance.

Performance Matrix

The following observations were drawn from the previous Articles:

- AI-Driven Threat Detection: Studies have proved that AI-based solutions increased the overall accuracy and predictive capabilities, but the dependency on extensive dataset for training introduces potential risks related to data privacy and quality of data.

- False Positives and Negatives:Many studies show case that AI models can yield unaccurate results, which will lead to human supervision to validate critical security decisions.

- Adversarial Attacks: Research emphasizes the vulnerability of AI systems to adversarial manipulation, where malicious actors craft deceptive inputs to bypass security mechanisms.

- Hybrid Approaches: Combining AI-driven security with traditional methods has emerged as an effective solution to mitigate AI-related weaknesses while improving overall system performance.

Relevance To Proposed Model

Building on these key findings,My intelligent cloud security model is developed to make the following challenges:

1. An AI-enhanced threat detection system that improve accuracy and minimize false alarms(false positives).

2. Adaptive learning mechanisms that continuously update AI models to counter adversarial manipulation tactics and improve system robustness.

By developing these limitations, My model can provide a scalable, secure and cost effective cloud security solution that can prevent threats and have response cababilities.

2.5 Common Limitations in AI-Driven Cloud Security

Although the improvement in AI-driven solutions, some limitations were found in the reviewed articles which are listed to follows:

- Data Dependency: AI models rely heavily on large datasets for training. Poor-quality or biased data can compromise model accuracy, leading to ineffective threat detection.

- Adversarial Attacks: AI systems are susceptible to adversarial manipulation, where malicious actors craft deceptive inputs to bypass security mechanisms.
- False Positives and False Negatives: While AI improves detection capabilities, unfortunately, models in some cases yields false alarms, which are coming from the false positives and negatives that always come from any models.
- Implementation Costs: AI-based security systems often demand significant computational resources, skilled personnel, and continuous updates, which may limit their adoption by smaller organizations.
- Scalability Challenges: Ensuring AI models perform efficiently in large-scale cloud environments presents difficulties in terms of performance optimization and resource allocation.
- Privacy Concerns: The extensive data collection required for AI training raises ethical and privacy concerns, particularly when dealing with sensitive user information.

2.6 Triggers

To effectively boost cloud security making a responding trigger is considered a crucial step.The reviewed research articles and studies have also highlighted various triggers that activate security security procedure and mechanism to make sure that the threat detected is prevented from happening. The following keys triggers were identified:

2.6.1 User Behavior Anomalies

Research on AI-driven user behavior analysis[13] emphasizes the role of monitoring unusual activities. Triggers such as:

- Unusual login patterns (e.g., unexpected devices or locations).

- Sudden access to sensitive data beyond the user's typical behavior.

These triggers can launch security alerts or activate multi-factor authentication protocol, preventing the unauthorized user.

2.6.2 Intrusion Triggers in Cloud Environments

AI-driven Intrusion Detection Systems (IDS) [18] identify triggers such as:

- Anomalous network behavior patterns detected using models like Artificial Neural Networks (ANN) and Information Theoretic Models (ITM).
- Unauthorized access signatures identified through misuse detection techniques such as State Transition Diagrams.

These triggers prompt the IDS to generate alerts and activate decision-making mechanisms to mitigate potential security threats in cloud infrastructures.

2.6.3 Triggering Mechanism in Air-Gapped Maritime Systems

Maritime cyber threat research [19] identifies triggers such as:

- Specific signal patterns transmitted via radar or AIS used to remotely activate pre-installed malicious code.
- Template matching techniques that detect attack commands even when systems are air-gapped from other networks.

These triggers enable attackers to bypass traditional connectivity constraints and activate threats while maintaining safeguards against accidental activation.

Chapter 3

Methodology

In this Section,we showcase the used dataset for traing and evaluating models, in addition to the used machine learning and deep learning models used in my project,the data preprocssing steps,model selection reasons and development details that form our experimental setup.

3.1 Data Collection and Description

The dataset used in this project is the CIC IDS 2017 dataset, developed by the Canadian Institute for Cybersecurity.It is an extensive ecaluation framework for testing annd evaluating detection systems and contains close to evry netwrok traffic data having label with normal traffic(Benign) and the malicious activities or potential threat(eg.,DoS,Brute Force, Infiltration).This dataset has been popularly used in many research articles, including studies similar to ours, due to its richness and realism in representing contemporary cyber threats[20] [21].

The dataset is provided as eight separate CSV files, each one represent network traffic data captured at different times during the week targeting a specific threat type.The number of rows in each file varies, ranging from 100,000 to 600,000 rows,Despite the differences in row counts, all CSV files share the same features which are 78 excluding the label.

3.1.1 Feature Overview

Each row in the dataset represents a single network flow, capturing detailed information about that specific interaction. It contains various features such as source and destination IP addresses, timestamps, protocols, and traffic statistics. These features help in analyzing and identifying patterns within the network data. all features are listed in Tabel 3.1

Table 3.1: Overview of All Features

Feature 1	Feature 2	Feature 3
Packet Length Std	Bwd IAT Mean	Bwd IAT Std
Fwd Packet Length Min	Min Packet Length	Active Mean
Fwd IAT Min	Active Min	min_seg_size_forward
Active Max	Bwd IAT Min	Flow IAT Min
Idle Max	Idle Mean	Idle Min
act_data_pkt_fwd	PSH Flag Count	Down/Up Ratio
ACK Flag Count	Idle Std	FIN Flag Count
URG Flag Count	Active Std	SYN Flag Count
Fwd PSH Flags	RST Flag Count	ECE Flag Count
Bwd Avg Bulk Rate	CWE Flag Count	Fwd Avg Packets/Bulk
Fwd Avg Bytes/Bulk	Bwd Avg Packets/Bulk	Fwd URG Flags
Bwd PSH Flags	Bwd URG Flags	Bwd Avg Packets/Bulk
Fwd Avg Packets/Bulk	Bwd Avg Bytes/Bulk	Total Length of Bwd Packets
Subflow Bwd Bytes	Destination Port	Packet Length Variance
Bwd Packet Length Mean	Avg Bwd Segment Size	Bwd Packet Length Max
Init_Win_bytes_backward	Total Length of Fwd Packets	Subflow Fwd Bytes
Init_Win_bytes_forward	Average Packet Size	Packet Length Mean
Max Packet Length	Fwd Packet Length Max	Flow IAT Max
Bwd Header Length	Flow Duration	Fwd IAT Max
Fwd Header Length	Fwd IAT Total	Fwd IAT Mean
Flow IAT Mean	Flow Bytes/s	Bwd Packet Length Std
Subflow Bwd Packets	Total Backward Packets	Fwd Packet Length Mean

Feature 1	Feature 2	Feature 3
Avg Fwd Segment Size	Bwd Packet Length Min	Flow Packets/s
Fwd Packets/s	Bwd IAT Max	Bwd Packets/s
Total Fwd Packets	Subflow Fwd Packets	Bwd IAT Total
Label		

3.2 Dataset Setup

The CIC IDS 2017 dataset was divided into multiple CSV files, as previously mentioned, each corresponding to a specific type of cyberattack. Below is a description of each file used in this project, ordered alphabetically by threat type:

3.2.1 Botnet Traffic Dataset

This dataset contains traffic patterns associated with botnet activity, where infected hosts may be used for malicious communications or as part of coordinated attacks. Here is a sample from the instances in `bot.csv` shown in Tabel 3.2:

Table 3.2: Example Records from Bot Interaction Dataset (bot.csv)

Destination Port	**Bwd Packet Length Max**	**Bwd Packet Length Std**	**Flow Bytes/s**	**Label**
24911	80	0.0	0.000000e+00	1
31415	80	370.0	8.796263e+01	1
49691	80	0.0	0.000000e+00	1
72485	53	59.0	0.000000e+00	0
108808	53	172.0	7.885226e+03	0

3.2.2 Brute Force Attack Dataset

This file includes traffic data related to brute force login attempts, typically targeting SSH or web applications through repeated password guessing. Here is a sample from the dataset shown in Tabel 3.3:

Table 3.3: Example Login Attempts from BruteForce Dataset

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
41637	21	34	14.54975	3.337180e+01	1
431212	22	0	0.00000	0.000000e+00	1
49533	21	0	0.00000	6.140351e+04	1
182173	53	93	0.00000	1.207921e+06	0
100939	53	183	0.00000	3.665317e+03	0

3.2.3 Overview of the DDoS Dataset

This dataset captures Distributed Denial of Service (DDoS) attack traffic, characterized by overwhelming the network with a high volume of malicious requests. A sample from the dataset is shown in Table 3.4.

Table 3.4: Network Traffic Records from DDoS.csv Dataset

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
121983	80	7215	3327.769794	1439.401114	1
91910	80	5840	3173.373883	68404.866490	1
77805	80	7215	2991.555649	122478.431700	1
186100	40474	6	0.000000	5.012003	0
120152	28907	6	0.000000	38.358716	0

3.2.4 Overview of the DDoS Dataset

The Denial of Service (DoS) dataset includes traffic associated with attacks aimed at making a machine or network resource unavailable by flooding it with traffic or triggering crashes.Here is a sample from the dataset shown in Tabel 3.5:

Table 3.5: Sample Dataset Entries from DoS.csv

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
273037	80	4344	1754.831473	120.987802	1
238103	80	4344	1760.597234	119.682114	1
535567	80	10184	4437.056321	1173.846186	1
675356	50419	6	0.000000	88888.888890	0
68284	443	1418	557.143954	60.528727	0

3.2.5 Overview of Infiltration Activity Captured in Infiltration Dataset

This file represents infiltration attacks, where unauthorized access is attempted or achieved, often from within the network.Here is a sample from the dataset found in Tabel 3.6 :

Table 3.6: Host Activity Logs from Infiltration.csv Dataset

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
224706	444	6	0.000000	1.273469e+00	1
85564	444	6	0.000000	1.499021e+03	1
166343	444	6	0.000000	5.476221e+02	1
16	53	49	0.000000	1.064935e+06	0
97711	443	2896	1034.372751	7.568605e+02	0

3.2.6 Overview of Network Probing Events in PortScan Dataset

This dataset contains traffic data related to Port Scanning, where an attacker probes a range of ports to identify potential services to exploit.Here is a sample from the dataset found in Tabel 3.7:

Table 3.7: Sample Dataset Entries from PortScan.csv

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
162826	84	6	0.00000	1.363636e+05	1
193888	4449	6	0.00000	1.111111e+05	1
166717	12345	6	0.00000	2.222222e+05	1
26828	443	1460	744.42413	9.293421e+02	0
2814	53	98	0.00000	1.380952e+06	0

3.2.7 Overview of Web Application Attacks Captured in WebAttack Dataset

This dataset includes traffic related to web-based attacks such as SQL Injection, Cross-Site Scripting (XSS), and Brute Force attacks targeting web applications.Here is a sample from the dataset 3.8:

Table 3.8: Web-Based Attack Patterns Captured in WebAttack Dataset

Index	Destination Port	Bwd Packet Length Max	Bwd Packet Length Std	Flow Bytes/s	Label
162826	84	6	0.00000	1.363636e+05	1
193888	4449	6	0.00000	1.111111e+05	1
166717	12345	6	0.00000	2.222222e+05	1
26828	443	1460	744.42413	9.293421e+02	0
2814	53	98	0.00000	1.380952e+06	0

3.3 Data Preprocessing

Before training the models, I applied several preprocessing steps to clean and prepare the data for training the models:

- Duplicate Removal: I removed all duplicate rows to ensure the dataset only contained unique network traffic records.

- Missing Values: Any rows containing `NaN` (null) values were dropped to avoid introducing bias or errors during training.

- Infinity Handling: Some columns contained infinite values, which were replaced with the maximum floating-point value (`np.finfo(np.float32).max`) to preserve data integrity while avoiding model errors.

- Train-Test Split: I split the dataset into training and testing sets using a 70:30 ratio. This ratio made more time efficent data for model training while saving a considerably great amount of data for evalution.

- Feature Importance Selection: I performed feature selection and dropped all irrelevant columns across all models based on a predefined importance threshold and made sure the features belong to this range 10-15. Using feature importance scores.This step left only the most relevant features that the model can train on, which helps in reducing overfitting and improving generalization across different algorithms. A similar strategy was employed in [22], where the feature set was reduced to enhance model performance.

- Label Encoding and Binarization: The target label contained various specific threat types (e.g., different categories of DDoS attacks). I encoded these values to numerical values and transformed the categories to be all belong to one single class to make the model main task binary classifcation task. Any row representing a threat was labeled as `1`, while benign(Normal traffic) traffic was labeled as `0`. This allowed the models to focus purely on detecting the presence or absence of malicious activity.

- Random Undersampling: Due to theincredibaly large size of the datasets, I applied random undersampling on the training set to reduce computational load. This also helped balance the dataset and improve training speed without affecting model generalization. Random undersampling was also shown to yield better results in building models in research like in [23].

3.4 Data Split After Preprocessing

After preprocessing, the following information is provided for each dataset:

- Bot.csv:
  - Total rows: 191,033
  - Number of attack instances (Bot): 1,966
  - Number of normal instances: 189,067
  - Training split after random under-sampling: 2,804 (1,402 Attack Instances)
  - Testing split: 57,310(546 attack Instances)
- BruteForce.csv:
  - Total rows: 445,909
  - Number of attack instances (BruteForce): 13,835
  - Number of normal instances: 432,074
  - Training split after random under-sampling: 19,346(9,673 Attack Instances)
  - Testing split: 133,773(4,162 Attack Instances)
- DDoS.csv:
  - Total rows: 225,745
  - Number of attack instances (DDoS): 128,027
  - Number of normal instances: 97,718
  - Training split after random under-sampling: 136,912(68,456 Attack Instances)
  - Testing split:67,724(38,462 Attack Instances)
- DoS.csv:

    - Total rows: 692,703
    - Number of attack instances (DoS): 252,672
    - Number of normal instances: 440,031
    - Training split after random under-sampling: 353,886(176,943 Attack Instances)
    - Testing split: 207,811(75,729 Attack Instances)
- Infiltration.csv:
    - Total rows: 288,602
    - Number of attack instances (Infiltration): 36
    - Number of normal instances: 288,566
    - Training split after random under-sampling: 56(28 Attack Instances)
    - Testing split: 86,581(8 Attack Instances)
- PortScan.csv:
    - Total rows: 286,467
    - Number of attack instances (PortScan): 158,930
    - Number of normal instances: 127,537
    - Training split after random under-sampling: 178,480(89,240 Attack Instances)
    - Testing split: 85,941(47,644 Attack Instances)
- WebAttack.csv:
    - Total rows:170,366
    - Number of attack instances (WebAttack): 2,180
    - Number of normal instances: 168,186
    - Training split after random under-sampling: 3,024(15,12 Attack Instances)
    - Testing split: 51,110 (668 Attack Instances)

3.5 Evaluation Metrics

In order to properly evaluate our classification models,I used evaluation metrics which are mainly derived from the confusion matrix components like True Positives,True Negatives, False Positives and False Negatives.

Here is an example of how the confusion matrix looks like in general.

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

3.5.1 Accuracy

Accuracy measures the overall correctness of the model and is defined as the ratio of correctly predicted instances to the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3.5.2 Precision

Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3.5.3 Recall (Sensitivity)

Recall measures the proportion of correctly predicted positive instances out of all actual positive instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

3.5.4 F1-Score

The F1-Score is the harmonic mean of Precision and Recall, providing a balance between the two metrics.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where Precision equation and Recall equation are used to compute the F1-Score.

3.5.5 How To Obtain These Values

To calculate these metrics, we first construct the confusion matrix based on model predictions and true labels:

- True Positive (TP): Number of correctly predicted positive samples.
- True Negative (TN): Number of correctly predicted negative samples.
- False Positive (FP): Number of negative samples incorrectly predicted as positive.
- False Negative (FN): Number of positive samples incorrectly predicted as negative.

By counting the above values from the classification output, we substitute them into the formulas above to compute each evaluation metric.

3.6 Model Architecture

In this section, we will explain how each model works and describe its structure. We will also highlight why each model is suitable for detecting threats in network traffic, focusing on their strengths in identifying patterns and making accurate predictions.

3.6.1 Decision Tree

Overview: The Decision Tree model is a supervised machine learning algorithm used for classification and regression tasks.

How it Works: Its main objective is to recurisvly split the data set based on feature values to creat branches untill it reachches leaves that represent class labels. the algorithm chooses the feauture to split on based on some algorithms like gain or GINI impurity.the splitting continues recursively until a stopping condition is met like maximum depth and minimum sample leaf which are parameters that the machine learning engineer is using to optimize the model results.

Why it Was Chosen: Decision Tree is one of the easiest models to implement in machine learning and considered very effective in classification tasks. Given the dataset's structure and the need for explainable models, Decision Trees were a natural choice for initial modelling. They have also been widely used in similar articles, such as [24], due to these advantages.

Model Parameters:

- Maximum Depth: The depth of the tree is limited to avoid overfitting.
- Minimum Samples Split: Specifies the minimum number of samples required to split a node.

To better illustrate the structure and decision process, Figure 3.1 shows a visual representation of a Decision Tree.

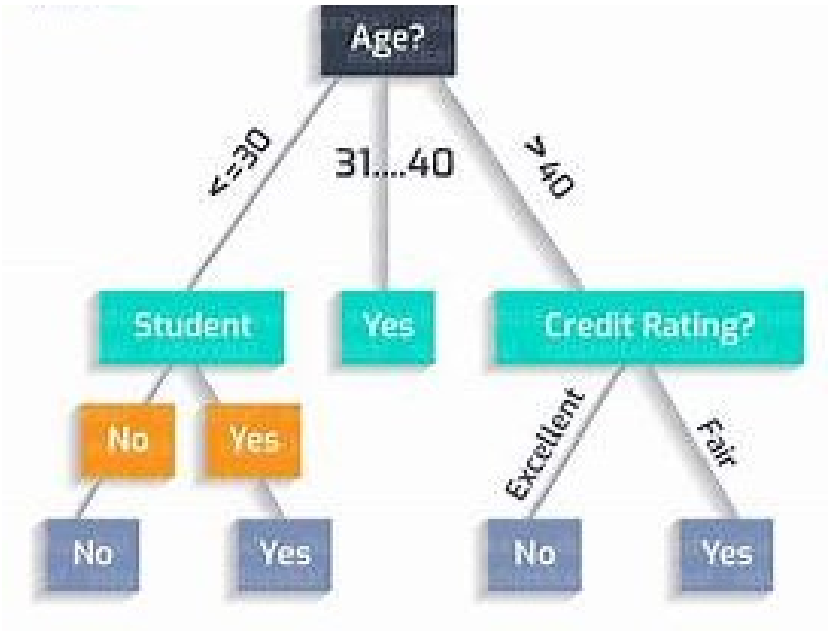

Figure 3.1: Illustration of a Decision Tree model structure and splitting process.

3.6.2 Random Forest

Overview: Random Forest is an ensemble learning technique that builds multiple decision trees and merges their predictions to improve accuracy and robustness through voting. It is widely used for classification tasks because it reduces overfitting and improves generalization by averaging the predictions of several trees.

How it Works: Each tree in a Random Forest is trained on a random subset of the data, with bootstrapping (sampling with replacement) to create diverse trees. At each node, a random subset of features is considered for splitting, which ensures that the trees are less correlated and more independent. The final output is the majority vote (for classification) or the average (for regression) of all trees.

Why it Was Chosen: Random Forest provides a powerful combination of high accuracy and low overfitting by aggregating multiple weak models (decision trees). It is well-suited for large datasets and can handle a high number of features,

making it a suitable choice for threat detection with the CIC IDS 2017 dataset. Random Forest has also been widely used in similar articles, such as [25],[24].

Model Parameters:

- Number of Estimators: The number of decision trees in the forest.
- Maximum Depth: Limits the depth of each individual tree to prevent overfitting.
- Minimum Samples Split: Ensures that nodes contain a minimum number of samples before they are split.

A graphical overview of the Random Forest architecture is presented in Figure 3.2.

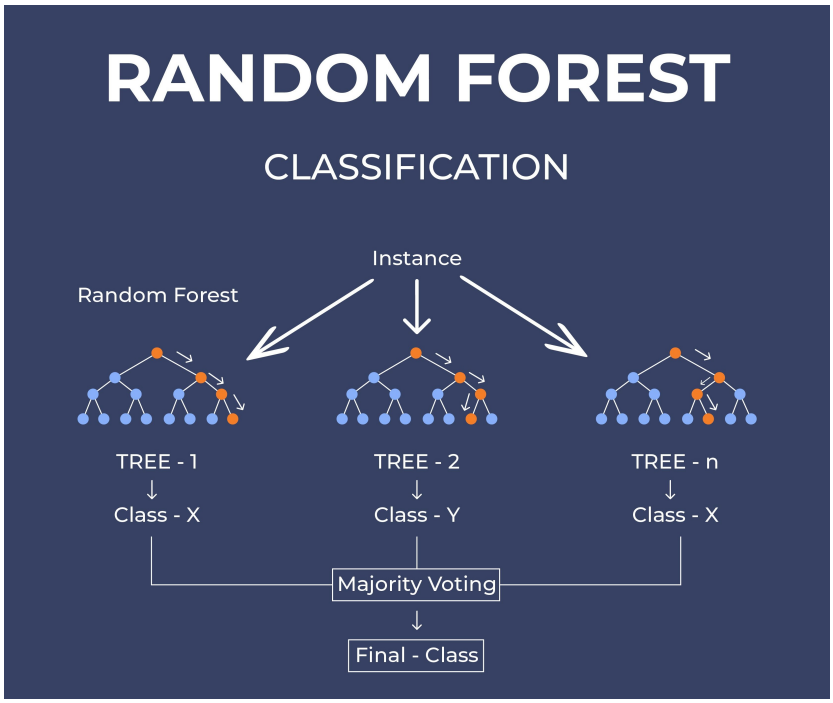

Figure 3.2: Diagram showing how Random Forest combines multiple decision trees for classification.

3.6.3 LSTM (Long Short-Term Memory)

Overview: LSTM is a type of Recurrent Neural Network (RNN) designed to model sequential data. Unlike traditional RNNs, LSTMs are capable of learning long-term dependencies due to their special memory cells that retain information over

time. LSTMs are effective in scenarios where the order of inputs matters, such as time-series prediction or sequential event classification.

How it Works: LSTM networks consist of units called cells, which are designed to store information for long periods of time. These cells have three gates: input, forget, and output gates, which regulate the flow of information into and out of the cell. By controlling these gates, LSTMs can learn which information to remember or forget as new data points are received.

Why it Was Chosen: Since network traffic data can exhibit sequential dependencies (e.g., the order of packets), LSTM was chosen for its ability to capture these temporal relationships, making it ideal for analyzing network behavior over time[26].

Model Parameters:

- Number of Layers: The number of stacked LSTM layers.
- Number of Units: The number of memory units in each LSTM layer.
- Dropout Rate: Regularization to prevent overfitting by randomly setting a fraction of input units to zero during training.

Figure 3.3 depicts the architecture of an LSTM cell, highlighting its key components and gates.

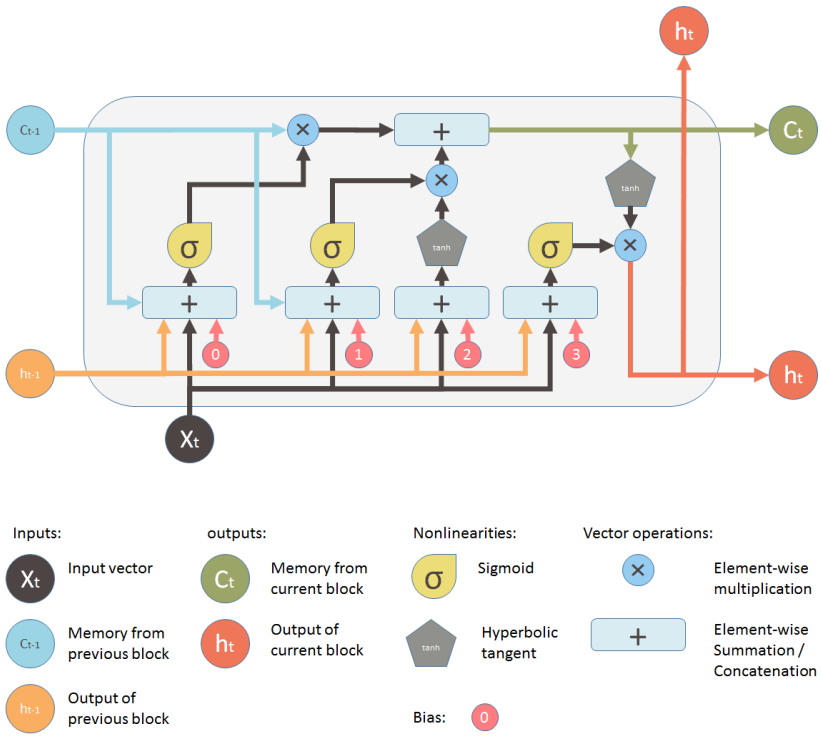

Figure 3.3: Structure of an LSTM cell showing its gates and memory cell components.

3.6.4 CNN (Convolutional Neural Network)

Overview: CNN is a type of deep learning model primarily designed for processing data with a grid-like topology, such as images. However, CNNs have also proven effective in 1D data formats like network traffic flows, where they can automatically learn spatial hierarchies of features through convolutional operations.

How it Works: CNNs consist of layers of convolutional filters that scan across input data to detect local patterns. These are followed by pooling layers, which reduce dimensionality and help retain the most significant features. Fully connected layers at the end of the network perform the final classification. By stacking multiple convolutional layers, CNNs can capture increasingly complex features.

Why it Was Chosen: CNN was chosen for its strong ability to extract local patterns and hierarchical features from structured data. In the context of the CIC IDS 2017 dataset, CNNs can effectively capture spatial correlations within

the features of each traffic flow. CNN has also been widely used in similar intrusion detection research due to its high accuracy and efficiency, as shown in [27],[28].

Model Parameters:

- Number of Convolutional Layers: Controls how many feature maps are extracted at different levels.
- Filter Size: Determines the receptive field of each convolution, affecting the granularity of detected patterns.
- Stride and Padding: Define how the convolution is applied and whether the input size is preserved.
- Pooling Type: Max pooling or average pooling, used to downsample feature maps.
- Dropout Rate: Regularization technique to reduce overfitting.

3.6.5 AutoKeras

Overview: AutoKeras is an open-source software library for automated machine learning (AutoML), which simplifies the process of building and training deep learning models. It automates the search for the best model architecture and hyperparameters using techniques like neural architecture search (NAS).

How it Works: AutoKeras automatically searches for the optimal model architecture and hyperparameters by exploring various combinations of network structures and training configurations. It uses a search algorithm to test different architectures and select the one that performs best on the given task, without requiring the user to manually design the model.

Why it Was Chosen: AutoKeras was chosen for its ability to optimize the deep learning model search process automatically. Given the large dataset and complexity of threat detection, AutoKeras helps in identifying the best-performing

models without the need for manual tuning, saving time and potentially improving performance[29].

Model Parameters:

Search Space: Defines the set of possible model architectures and hyperparameters to explore.

Epochs: The number of training epochs to evaluate each model configuration.

3.6.6 Model Training Process

The training process involved applying multiple machine learning and deep learning models to the preprocessed datasets to create potential threat detection models for every threat type. The models used include Decision Tree, Random Forest, LSTM, CNN and AutoKeras.Now I will discuss more in-depth details that were done while training the models:

- Environment and Tools: All models were implemented using Python. Traditional models were developed using `scikit-learn`, while deep learning models (LSTM and CNN) were built using `TensorFlow/Keras`. `AutoKeras` was used for automated neural architecture search.
- Data Format:
  - For Decision Tree and Random Forest, the data was used as it is without normalization.
  - For LSTM and CNN models, feature values were normalized using `Scaler`.
  - LSTM Input Reshaping: The input data was reshaped into 3D arrays of shape `[samples, timesteps, features]` as required for LSTM models. Each of the 78 original features was treated as a separate timestep with a single feature value, allowing the LSTM model to learn temporal dependencies across the feature vector.

    * CNN Input Reshaping: CNN inputs were reshaped into 3D arrays of shape `[samples, features, 1]`, where each feature vector was treated as a sequence for 1D convolution. This allowed the CNN model to capture local patterns across the feature space using `Conv1D` layers.

- Model-Specific Training:
    * Decision Tree: Hyperparameters such as `max_depth` and `min_samples_leaf` were tuned manually through experimentation to balance model complexity and generalization.
    * Random Forest: Trained using 200 estimators with default bootstrapping. The same manually tuned hyperparameters for `max_depth` and `min_samples_leaf` as the Decision Tree were applied. Additionally, the top 10 features (selected based on feature importance) were used for training to enhance efficiency and reduce overfitting.
    * LSTM: The architecture consisted of two stacked LSTM layers with a tunable number of hidden units, followed by dense layers for final classification. Dropout layers were applied after each LSTM and dense layer to prevent overfitting. The model used binary cross-entropy as the loss function and a tunable optimizer (Adam, RMSprop, or SGD). Hyperparameters such as the number of LSTM units, dense units, and dropout rate were optimized using Keras Tuner with Random Search.
    * CNN: A 1D Convolutional Neural Network was designed using two convolutional layers followed by max pooling and global average pooling. The tabular input data was reshaped into a 1D format with a single channel to make it compatible with convolutional layers. The model included dense and dropout layers for regularization and used binary cross-entropy loss with a tunable optimizer. Key hyperparameters such as the number of filters, kernel sizes, dense units, and dropout rate were tuned using Keras Tuner with Random Search.

    * AutoKeras: Utilized to automatically design a deep learning model. AutoKeras managed preprocessing, model architecture selection, and hyperparameter tuning autonomously.

- Hyperparameter Tuning:
    * Traditional models: Number of trees, max depth, and minimum samples per split were tuned manually.
    * Deep learning models: Epochs, batch size, learning rate, and the number of layers/units were adjusted through experimentation.
- Evaluation During Training:
    * Models were evaluated using accuracy,loss,F1,recall and confusion matrix.
    * For LSTM and CNN, training and validation metrics (loss and accuracy) were plotted over epochs to monitor performance.

3.7 Tools and Environment

The experiments and model training were conducted using the following tools and environment:

- Programming Language: Python was used as the primary programming language due to its extensive libraries for data analysis, machine learning, and deep learning.
- Libraries and Frameworks:
    * TensorFlow and Keras: For building, training, and tuning the neural network models (LSTM, CNN).
    * Scikit-learn: For preprocessing the dataset, such as scaling the features, splitting the data, and applying Random Undersampling (Rus). It was

also used for evaluating model performance (metrics such as accuracy, precision, recall, etc.).

    * Keras Tuner: Used for hyperparameter tuning, allowing for automatic search of the best hyperparameters for the CNN and LSTM models.
    * Pandas and Numpy: For data manipulation, feature extraction, and numerical operations.
    * Imbalanced-learn: Used for handling the class imbalance in the dataset using techniques like Random Undersampling.

- Development Environment:
    * IDE: Visual Studio Code (VSCode) was used as the integrated development environment (IDE) for writing and running the code.

Chapter 4

Results and Discussion

This chapter is organized to showcase and discuss the results of all models deveoped through out the study. I will discuss the performance on each individual attack type dataset in details, a comparative analysis part analyzing the best model developed from each threat type and highlighting strength and weaknesses of the models and finaly comparing my model's results with other articles in each threat type.

4.1 Results on Individual Attack Types

In this section, I present the performance results of the models on each attack type dataset and displaying the accuracy,precision,recall and f1-score of each model on each dataset.

4.1.1 Botnet Attack Results

For the Botnet attack classification, the CNN model demonstrated the best overall performance with the highest accuracy of 99.3%. Additionally, it presented a balanced exchange between precision(0.68) and (0.677), making it more reliable for practical detection. Although the Random Forest model had a considerably close accuracy performance of 97.4% and a slightly better recall of 0.996, its very low precision (0.29) reduces its reliability in real word sitiuatioin, as it would

produce many false positives.More info of performance done by each model is shown in Tabel 4.1.

Table 4.1: Evaluation Metrics of Classification Models on Botnet Dataset

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	97.2%	0.27	0.966	0.436
Random Forest	97.4%	0.29	0.996	0.45
CNN	99.3%	0.68	0.677	0.674
LSTM	81.7%	0.0540	0.972	0.10231
Auto-Keras	93.6%	0.143	0.986	0.2497

4.1.2 Brute Force Attack Results

For the Brute Force attack classification, both the Decision Tree and Random Forest models achieved the highest accuracy of 99.8% with perfect recall (1.0) and very high precision (0.947). but we need to mention that the random forest performs slightly better than decison tree as it yielded better F1 score(0.973)(decision tree was 0.97), making it the best performing and most reliable model for this dataset. The CNN model also performed well with an accuracy of 99.7%, but had slightly lower precision and recall, resulting in a lower F1-Score of 0.95.More info of performance done by each model is shown in Tabel 4.2.

Table 4.2: Comparative Analysis of Classification Models on Brute Force Dataset

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.8%	0.947	1	0.97
Random Forest	99.8%	0.947	1	0.973
CNN	99.7%	0.92	0.987	0.95
LSTM	99.4%	0.801	0.968	0.8766
Auto-Keras	96.7%	0.3409	0.971	0.5046

4.1.3 DDoS Attack Results

For the DoS attack classification, the CNN model achieved the highest accuracy of 99.2%, with a precision of 0.978, recall of 0.997, and F1-Score of 0.988, making it the best overall performer. The Random Forest and Decision Tree models also performed well, with accuracies of 99.1% and 98.7% respectively, but their recall and F1-Scores were slightly lower than CNN. The LSTM model showed the lowest performance with 86.8% accuracy and comparatively lower precision and F1-Score. Auto-Keras performed moderately, but still behind CNN and the tree-based models.More info of performance done by each model is shown in Tabel 4.3.

Table 4.3: Performance Metrics for DDoS Attack Detection Models

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	99.9%	0.999	0.998	0.999
Random Forest	99.8%	0.999	0.998	0.999
CNN	99.3%	0.990	0.998	0.994
LSTM	99.9%	0.999	0.999	0.999
Auto-Keras	98.6%	0.977	0.999	0.987

4.1.4 DoS Attack Results

For the Infiltration attack classification, both the Decision Tree and Random Forest models achieved the highest accuracy of 99.9%. However, Random Forest outperformed Decision Tree in precision (0.8888 vs. 0.72727) and F1-Score (0.842 vs. 0.7619), making it the best performing model overall for this attack type. The CNN, LSTM, and Auto-Keras models showed significantly lower precision and F1-Scores despite relatively high recall values, indicating poor balance between detecting positive cases and minimizing false positives.More info of performance done by each model is shown in Tabel 4.4.

Table 4.4: Model Performance Comparison on DoS Attack Classification

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	98.7%	0.974	0.984	0.979
Random Forest	99.1%	0.988	0.984	0.986
CNN	99.2%	0.978	0.997	0.988
LSTM	86.8%	0.734	0.915	0.814
Auto-Keras	96.7%	0.92039	0.9820	0.95298

4.1.5 Infiltration Attack Results

For the Infiltration attack classification, both the Decision Tree and Random Forest models achieved the highest accuracy of 99.9%. However, Random Forest outperformed Decision Tree in terms of precision (0.8888 vs. 0.72727) and F1-Score (0.842 vs. 0.7619), making it the best-performing model overall for this attack type. The CNN, LSTM, and Auto-Keras models showed significantly lower precision and F1-Scores despite relatively high recall values, indicating a poor balance between detecting positive cases and minimizing false positives.More info of performance done by each model is shown in Tabel 4.5

Table 4.5: Performance Evaluation of Various Models on Infiltration Detection

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.9%	0.72727	0.8	0.7619
Random Forest	99.9%	0.8888	0.8	0.842
CNN	99.8%	0.004	0.6	0.007
LSTM	88.8%	0.0004	0.4	0.0007
Auto-Keras	69.2%	0.0004	0.1	0.0008

4.1.6 PortScan Attack Results

For the PortScan attack classification, the Random Forest model demonstrated the best overall performance with an accuracy of 99.8%, a precision of 0.999, a

recall of 0.998, and an F1-Score of 0.998. While other models such as Decision Tree, CNN, LSTM, and Auto-Keras also showed strong performance with accuracies above 99.4%, Random Forest achieved the best balance across all metrics, making it the most reliable model for detecting PortScan attacks.More info of performance done by each model is shown in Tabel 4.6

Table 4.6: Performance Comparison of Different Models on the PortScan Dataset

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	99.7%	0.996	0.998	0.997
Random Forest	99.8%	0.999	0.998	0.998
CNN	99.6%	0.994	0.997	0.996
LSTM	99.4%	0.994	0.993	0.9934
Auto-Keras	99.4%	0.9889	0.9992	0.99402

4.1.7 Web Attack Results

In the classification of Web Attacks, both the Decision Tree and Random Forest models exhibited the best performance, each achieving an accuracy of 99.3% and an F1-Score of 0.88. While the Random Forest had a slightly higher precision (0.801) compared to the Decision Tree (0.791), both maintained a high recall above 0.98, indicating consistent reliability in identifying web-based threats. The other models, particularly LSTM and Auto-Keras, achieved high recall but suffered from very low precision, resulting in unreliable overall performance.More info of performance done by each model is shown in Tabel 4.7

Table 4.7: Comparative Study of Machine Learning Models on Web Attack Detection

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.3%	0.791	0.991	0.88
Random Forest	99.3%	0.801	0.985	0.88
CNN	97.4%	0.766	0.693	0.728
LSTM	81.7%	0.054	0.972	0.102
Auto-Keras	97.5%	0.123	0.996	0.22

4.2 Comparative Analysis

This section showcases the performance diffrence between the best models from our project with models reported in other research articles, highlighting similarities, differences, and improvements.

4.2.1 Botnet Attack

In this comparison, we evaluate our best-performing model, the Convolutional Neural Network (CNN), against the top model reported in related literature for the Botnet attack classification task[30]. I evaluated their best performing model that have the closest number of features(10). The performance metrics are summarized in Table 4.8.

Table 4.8: Performance of my model against other model on Botnet Attack

Model	Accuracy	Precision	Recall	F1-Score
CNN (My Model)	99.3%	0.68	0.677	0.674
Random Forest	99.9607%	0.992	0.969	0.981

4.2.2 Brute Force Attack

In this subsection, I compare the performance of my Random Forest model on the Brute Force attack dataset with the Classifier Subset Evaluator using Naive Bayes as reported in [31]. Both models are evaluated using common performance metrics such as accuracy and sensitivity (recall). However, their evaluation also includes execution time, which unfortunately i didnt use as one of my perfomance metrics. Their model achieved an accuracy of 99.9888% and a sensitivity of 99.9928% with an execution time of 3.77 seconds. Meanwhile, my Random Forest model achieved an accuracy of 99.8%, precision of 0.947, recall of 1.0, and an F1-score of 0.973, demonstrating strong detection capabilities. Table 4.9 summarizes this comparison.

Table 4.9: Comparison of my model and another Brute Force Attack Dataset

Model	**Time (s)**	**Accuracy**	**Precision**	**Recall / Sensitivity**	**F1-Score**
RF (My Model)	-	99.8%	0.947	1.0	0.973
Classifier Subset Evaluator	3.77	99.9888%	-	99.9928%	-

4.2.3 DDoS Attack

In this section, I compare the performance of my Random Forest model with the results reported in a related article[32]. Both models exhibit very high accuracy and balanced precision, recall, and F1-scores, demonstrating strong classification capability for the considered attack type. The related article's Random Forest model achieves an accuracy of 99.96% with perfect precision, recall, and F1-score (all equal to 1.00), while my Random Forest model attains an accuracy of 99.8% with similarly high precision (0.999), recall (0.998), and F1-score (0.999). Although the related model slightly outperforms mine in accuracy and recall, my model remains highly competitive, showing reliable and robust detection performance. Table 4.14 summarizes this comparison.

Table 4.10: Performance Comparison of Random Forest Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest (My Model)	99.8%	0.999	0.998	0.999
Random Forest (Related Article)	99.96%	1.00	1.00	1.00

4.2.4 DoS Attack

The related article reports performance results on two DoS attack variants, Hulk and GoldenEye[33]. Averaging their results yields an accuracy of 98.5%, precision of 0.99, recall of 99.5%, and F1-score of 99.5%. In comparison, my CNN model achieves a slightly higher accuracy of 99.2%, with precision 0.978, recall 0.997, and F1-score 0.988. This slight improvement in accuracy might be due to the fact that their model performs multiclass classification across different DoS attack types, whereas my model focuses on a single-class classification task for detecting the presence of DoS attacks. Table 4.11 summarizes this comparison.

Table 4.11: Performance Comparison on DoS Attack Dataset

Model	Accuracy	Precision	Recall	F1-Score
CNN (My Model)	99.2%	0.978	0.997	0.988
Related Article Average	98.5%	0.99	0.995	0.995

4.2.5 PortScan Attack

The related article presents results using a Decision Table classifier on the PortScan dataset, achieving an accuracy of 99.82%, precision of 0.997, recall of 0.999, and F1-score of 0.998[34]. My Random Forest model attains a comparable performance with an accuracy of 99.8%, precision of 0.999, recall of 0.998, and F1-score of 0.998. This indicates that both models perform very closely on this task, with minor differences in precision and recall metrics. Table 4.12 provides a detailed comparison.

Table 4.12: My Random Forest model against Decision Table model on PortScan Attack Dataset

Model	Accuracy	Precision	Recall	F1-Score
Random Forest (My Model)	99.8%	0.999	0.998	0.998
Decision Table (Related Article)	99.82%	0.997	0.999	0.998

4.2.6 Comparison Limitations and Additional Analysis

For the Infiltration and Web Attack datasets, it was challenging to find directly comparable studies due to differences in dataset composition and classification approaches. Many related works either use mixed datasets combining multiple attack types or perform multiclass classification rather than the binary classification approach used in this study. These differences hinder direct performance comparisons on these specific attack types.

Fortunately, a recent article conducted a similar project focusing on these attack types and reported accuracy values for their models, which allows at least a partial comparison[35]. Although the article does not provide detailed metrics such as precision, recall, or F1-score, their reported accuracy can be compared with the results of our models for an overall performance perspective. You will find in this comparisons that our models yielded better overall accuracy, but take into consideration that the F1 score isn't presented by the article, so their model might have a rare chance that their performance might be more reliable, which I highly doubt, as our models yielded great F1 score results overall.Tables 4.13,4.14 summarizes this comparison.

Table 4.13: CNN Accuracy Comparison: Article vs. My Model

Threat Type	**CNN (Article)**	**CNN (My Model)**
Botnet	99.695	99.3
Brute Force	99.867	99.7
DDoS	99.998	99.3
DoS (Average)	94.90	99.2
Infiltration	94.444	99.8
Portscan	99.980	99.6
Web Attacks (Average)	93.447	99.4

Table 4.14: Random Forest Accuracy Comparison: Article vs. My Model

Threat Type	**Random Forest (Article)**	**Random Forest (Our Model)**
Botnet	97.253	97.4
Brute Force	98.009	99.8
DDoS	99.857	99.8
DoS (Average)	97.17	99.1
Infiltration	91.167	99.9
Portscan	99.917	99.8
Web Attacks (Average)	97.596	99.1

Chapter 5

Conclusion and Future Work

This chapter summarizes the main objectives and achievements of this project. It also outlines potential directions for future research that can build upon the work presented in this thesis.

5.1 Main Contributions

In this project, several key contributions were made, including but not limited to:

1. Proposing a novel approach based on machine learning and deep learning models, including Decision Trees,Random Forest,CNN,LSTM and Auto Keras, for detecting and classifying network intrusion attacks with enhanced accuracy using optimized feature selection.
2. Conducting a comprehensive experimental study to evaluate the model on various attack types.
3. Demonstrating improved performance in terms of accuracy, precision, recall, and F1-score compared to baseline approaches.

5.2 Future Work

Several avenues exist for extending and improving upon the current project:

1. Conducting experiments on additional datasets to assess the generalizability of the proposed model.
2. Exploring the integration of other deep learning architectures or ensemble techniques to further boost performance.
3. Investigating the response time for the model in real case scenarios and documenting the time for comparisons with other models where time was used as a performance metric.
4. Incorporating advanced feature engineering methods or dimensionality reduction techniques to improve computational efficiency.
5. Extending the model to perform multi-class classification across a wider variety of cyberattack types.
6. Incorporate automated triggers that execute specific procedures or routines upon detection of a threat, such as alert generation, system isolation, or countermeasures.

References

[1] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, 2021.

[2] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (iaas)," *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60–63, 2010.

[3] R. Yasrab, "Platform-as-a-service (paas): The next hype of cloud computing," *arXiv preprint arXiv:1804.10811*, 2018.

[4] S. Satyanarayana, "Cloud computing: Saas," *Computer Sciences and Telecommunications*, no. 4, pp. 76–79, 2012.

[5] Y. Li, Y. Lin, Y. Wang, K. Ye, and C. Xu, "Serverless computing: State-of-the-art, challenges and opportunities," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1522–1539, 2022.

[6] H. Sharma, "The evolution of cybersecurity challenges and mitigation strategies in cloud computing systems," *International Journal of Computer Engineering and Technology*, vol. 15, no. 4, pp. 118–127, 2024.

[7] C. Zuo, Z. Lin, and Y. Zhang, "Why does your data leak? uncovering the data leakage in cloud from mobile apps," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 1296–1310.

[8] M. A. M. Ariffin, M. F. Ibrahim, and Z. Kasiran, "Api vulnerabilities in cloud computing platform: Attack and detection," *International Journal of Engineering Trends and Technology*, vol. 1, pp. 8–14, 2020.

[9] O. R. Arogundade, "Addressing cloud computing security and visibility issues," *IARJSET*, vol. 10, no. 3, 2023.

[10] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (ddos) resilience in cloud: Review and conceptual cloud ddos mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.

[11] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE transactions on information forensics and security*, vol. 8, no. 12, pp. 1947–1960, 2013.

[12] S. Sangeetha, B. Gayathri Devi, R. Ramya, M. Dharani, and P. Sathya, "Signature based semantic intrusion detection system on cloud," in *Information Systems Design and Intelligent Applications: Proceedings of Second International Conference INDIA 2015, Volume 1*, Springer, 2015, pp. 657–666.

[13] S. O. Olabanji, Y. Marquis, C. S. Adigwe, S. A. Ajayi, T. O. Oladoyinbo, and O. O. Olaniyi, "Ai-driven cloud security: Examining the impact of user behavior analysis on threat detection," *Asian Journal of Research in Computer Science*, vol. 17, no. 3, pp. 57–74, 2024.

[14] R. Vadisetty, A. Polamarasetti, S. Prajapati, J. B. Butani, *et al.*, "Ai-driven threat detection: Enhancing cloud security with generative models for real-time anomaly detection and risk mitigation," *Available at SSRN 5218294*, 2023.

[15] M. I. Khan, A. Arif, and A. Khan, "Ai-driven threat detection: A brief overview of ai techniques in cybersecurity," *BIN: Bulletin of Informatics*, vol. 2, no. 2, pp. 248–61, 2024.

[16] A. Bhattarai, "Ai-enhanced cloud computing: Comprehensive review of resource management, fault tolerance, and security," *Emerging Trends in Machine Intelligence and Big Data*, vol. 15, pp. 39–50,

[17] P. Nina and K. Ethan, "Ai-driven threat detection: Enhancing cloud security with cutting-edge technologies," *International Journal of Trend in Scientific Research and Development*, vol. 4, no. 1, pp. 1362–1374, 2019.

[18] D. Gupta and S. Gupta, "An efficient approach of trigger mechanism through ids in cloud computing," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, IEEE, 2017, pp. 68–72.

[19] W. C. Leite Junior, C. C. de Moraes, C. E. de Albuquerque, R. C. S. Machado, and A. O. de Sá, "A triggering mechanism for cyber-attacks in naval sensors and systems," *Sensors*, vol. 21, no. 9, p. 3195, 2021.

[20] R. Selvam and S. Velliangiri, "An improving intrusion detection model based on novel cnn technique using recent cic-ids datasets," in *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)*, IEEE, 2024, pp. 1–6.

[21] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1192, 2019, p. 012 018.

[22] B. Reis, E. Maia, and I. Praça, "Selection and performance analysis of cicids2017 features importance," in *International Symposium on Foundations and Practice of Security*, Springer, 2019, pp. 56–71.

[23] Y.-B. Ho, W.-S. Yap, and K.-C. Khor, "The effect of sampling methods on the cicids2017 network intrusion data set," in *IT Convergence and Security: Proceedings of ICITCS 2021*, Springer, 2021, pp. 33–41.

[24] M. Azalmad, R. El Ayachi, and M. Biniz, "Unveiling the performance insights: Benchmarking anomaly-based intrusion detection systems using decision tree family algorithms on the cicids2017 dataset," in *International Conference on Business Intelligence*, Springer, 2023, pp. 202–219.

[25] T. Markovic, M. Leon, D. Buffoni, and S. Punnekkat, "Random forest based on federated learning for intrusion detection," in *IFIP international conference on artificial intelligence applications and Innovations*, Springer, 2022, pp. 132–144.

[26] N. Dash, S. Chakravarty, A. K. Rath, N. C. Giri, K. M. AboRas, and N. Gowtham, "An optimized lstm-based deep learning model for anomaly network intrusion detection," *Scientific Reports*, vol. 15, no. 1, p. 1554, 2025.

[27] V. R. Varanasi and S. Razia, "Cnn implementation for ids," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, IEEE, 2021, pp. 970–975.

[28] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "A survey of cnn-based network intrusion detection," *Applied Sciences*, vol. 12, no. 16, p. 8162, 2022.

[29] D. Sudyana, Y.-D. Lin, M. Verkerken, *et al.*, "Improving generalization of ml-based ids with lifecycle-based dataset, auto-learning features, and deep learning," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[30] A. F. Jabbar and I. J. Mohammed, "Development of an optimized botnet detection framework based on filters of features and machine learning classifiers using cicids2017 dataset," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 928, 2020, p. 032 027.

[31] S. Singh Panwar, Y. Raiwani, and L. S. Panwar, "Evaluation of network intrusion detection with features selection and machine learning algorithms

on cicids-2017 dataset," in *International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India*, 2019.

[32] M. I. Kareem and M. N. Jasim, "Fast and accurate classifying model for denial-of-service attacks by using machine learning," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1742–1751, 2022.

[33] A. F. Al-zubidi, A. K. Farhan, and S. M. Towfek, "Predicting dos and ddos attacks in network security scenarios using a hybrid deep learning model," *Journal of Intelligent Systems*, vol. 33, no. 1, p. 20 230 195, 2024.

[34] M. N. Jasim, A. M. A. Rahman, and M. J. Abdulredhi, "Machine learning classification-based portscan attacks detection using decision table," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 29, no. 3, pp. 1466–1472, 2023.

[35] Z. Pelletier and M. Abualkibash, "Evaluating the cic ids-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language r," *Science*, vol. 5, no. 2, pp. 187–191, 2020.