People's Democratic Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and scientific researche
وزارة التعليم العالي و البحث العلمي
National Higher School of Advanced Technologies
المدرسة الوطنية العليا للتكنولوجيات المتقدمة

Electrical Engineering and Industrial Computing Department

Final Year Project to Obtain the Diploma of

Engineering

- Field -

Telecommunication

- Specialty -

Telecommunication Systems and Networking

- Subject -

Machine Learning-Based DDoS Attacks Detection and Mitigation in O-RAN Enabled 5G Networks

Realized by

AISSA Manel Fatima Zohra & HAOUA Rania

Members of the Jury

CHIALI Imane	Chair
BOUTOUTA Dalila	Examiner
BEGHAMI Sami	Examiner
BENDOUDA Djamila	Supervisor

Algiers, **June 24th 2025**

Academic year 2024-2025

Dedication

To myself
For the perseverance, the silent battles, and the determination that brought me here. This achievement is a tribute to my personal growth.

To my dear father
Your unwavering support and strength have been a constant source of motivation. May these pages carry the echo of my deepest gratitude and love.

To my dear mother
Your infinite patience, care, and encouragement have illuminated my path throughout this journey. This work reflects all that you have given me.

To my beloved sister Ines and my dear brother Youssef
Your presence, affection, and joy have been my anchors. You are part of every step I take forward.

To all my family
For your love, warmth, and belief in me — even in silence.

To my friends
Your encouragement and kindness have lightened the road. May success always accompany you.

To all those I love
Thank you for being part of this journey.

Manel

Dedication

At this very moment, as I write these words, I am deeply aware that I would not have reached this point without the guidance and grace of God.

My deepest gratitude goes to those whose presence, love, and support have shaped this journey.

To my beloved parents, your love, your care, and your unwavering support have been the foundation of everything I have achieved. From my very first steps to this final milestone, you have stood beside me, and for that, I am endlessly grateful.

To my dear siblings, your constant encouragement and support have always given me strength, even in the hardest times.

To my friends, thank you all for being part of this journey. A special thank you to Maria, who has always stood by my side with kindness and loyalty.

To everyone who helped me, in small or big ways, since the very beginning, your impact is forever part of this accomplishment.

Thank you all.

And finally, I dedicate this work to myself, for never giving up, for battling through every challenge, even when it was hard to.

Rania

Acknowledgment

First and foremost, we would like to express our deepest gratitude to our god for granting us the strength, patience, and determination needed to complete this project.

We are especially grateful to our supervisor, **BENDOUDA Djamila**, for her unwavering support, generous guidance, and continuous encouragement. Her expertise, availability, and trust have been invaluable throughout every stage of this work. Her insightful advice has significantly enriched the quality of our project.

We would also like to extend our heartfelt thanks to the members of the jury **CHIALI Imane**, **BOUTOUTA Dalila**, and **BEGHAMI Sami** for taking the time to examine and evaluate our work. Their thoughtful feedback and valuable suggestions are sincerely appreciated.

We are also profoundly thankful to our families and friends, who have been a constant source of support and motivation throughout the completion of this thesis. Their encouragement, understanding, and belief in us were instrumental in helping us overcome challenges and achieve our goals.

Finally, to all those who contributed directly or indirectly to the success of this project, thank you. Your support has meant a great deal to us.

الملخص

يمثل دمج شبكة O-RAN في شبكات الجيل الخامس (5G) تحولًا جذريًا في مجال الاتصالات المتنقلة، حيث يُدخل مفاهيم الانفتاح والبرمجة والتحكم الذكي من خلال وحدات التحكم الذكية المركزية (RIC) في الشبكة الراديوية. واستنادًا إلى مبدأ الشبكات المعرفة بالبرمجيات ،(SDN) تكمن الابتكارية الأساسية في O-RAN في فصل منطق التحكم عن مستوى البيانات، مما يمكّن من إدارة ديناميكية وذكية للشبكة. ومع ذلك، فإن هذه التطورات يصاحبها تحديات أمنية كبيرة، وعلى رأسها التهديد المتزايد لهجمات حجب الخدمة الموزعة .(DDoS) ولمعالجة هذه الثغرات، يقترح هذا المشروع إطارًا ذكيًا للكشف عن هجمات DDoS والتصدي لها في الزمن الحقيقي داخل شبكة O-RAN للجيل الخامس.

يعتمد جوهر الحل المقترح على منصة اختبار متكاملة وقابلة للتجزئة (slice-aware) ومبنية بالكامل باستخدام الحاويات، تم إنشاؤها بواسطة OpenAirInterface (OAI) وتُدار عبر وحدة التحكم .FlexRIC تتيح هذه البنية التحتية إدارة مرنة وقابلة للبرمجة للشبكة، مما يسمح بالتحكم الدقيق في اتصال المستخدم وتجزيء الشبكة على مستوى النفاذ الراديوي .(RAN) لتعزيز قدرات الكشف، قمنا بتصميم وتقييم مجموعة من نماذج التعلم الآلي، شملت: غابة عشوائية (Random Forest) ،الشبكة العصبية الالتفافية ،(CNN) الشبكة العصبونية التكرارية ثنائية الاتجاه ،(BiLSTM) بالإضافة إلى نهج تجميعي .(ensemble) وقد أظهر نموذج الغابة العشوائية أفضل أداء، محققًا دقة بنسبة 99.0٪ أثناء التدريب، و90٪ أثناء محاكاة حركة مرور في الزمن الحقيقي، مع الحفاظ على تكلفة حسابية منخفضة. تم دمج نظام الكشف هذا داخل مكون وظيفة مستوى المستخدم ،(UPF) مما يسمح بتحليل لحظي لحركة المرور واكتشاف السلوكيات الشاذة. وعند التعرف على نشاط ضار، تقوم وحدة xApp مخصصة، منشورة ضمن RIC القريب من الزمن الحقيقي (Near-RT RIC) عبر FlexRIC، بإرسال أمر "إفراج RRC RRC") (Release لفصل الجهاز المتأثر على الفور.

تؤكد النتائج التجريبية فعالية نظامنا. حيث يحقق اكتشافًا دقيقًا وفوريًا لهجمات DDoS مع انخفاض معدل الإيجابيات الخاطئة وزمن تأخير منخفض باستمرار، حتى في ظروف الهجوم، مع الحفاظ على إنتاجية عالية في الاتصال الصاعد. وهذا يثبت قدرة الإطار على الحفاظ على أداء المستخدمين الشرعيين. بشكل عام، يقدم هذا العمل بنية عملية وقابلة للتوسع تعزز بشكل كبير أمان وقدرة شبكات 5G على الصمود في وجه التهديدات السيبرانية الناشئة.

كلمات مفتاحية: أمن الجيل الخامس، ،O-RAN هجمات حجب الخدمة ،(DDoS) التعلم الآلي، ،OpenAirInterface FlexRIC.

Abstract

The integration of O-RAN into 5G networks marks a major paradigm shift in mobile communications, introducing openness, programmability, and AI-driven control through centralized RAN Intelligent Controllers (RIC). Inspired by Software-Defined Networking (SDN), O-RAN's core innovation is the decoupling of control logic from the data plane, enabling dynamic and intelligent network management. However, these advancements also come with significant security challenges, particularly the increasing threat of Distributed Denial of Service (DDoS) attacks. To address these vulnerabilities, this project introduces an intelligent framework for real time DDoS detection and mitigation within a next generation Open RAN (O-RAN) 5G network.

At the core of our solution is a fully containerized and slice aware 5G Simulated Network, built using OpenAirInterface (OAI) and orchestrated through the FlexRIC controller. This architecture enables dynamic and programmable network management, offering precise control over user connectivity and network slicing at the RAN level. To enhance detection capabilities, we designed and evaluated a suite of machine learning models. These include Random Forest, Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and an ensemble approach. Among them, the Random Forest model demonstrated the best performance, achieving 99.0% accuracy during training and 90% accuracy under real time traffic simulations, all while maintaining low computational overhead. This detection system was integrated into the User Plane Function (UPF), allowing real time traffic analysis and anomaly detection. When malicious activity is identified, a dedicated xApp deployed in the Near-RT RIC via FlexRIC triggers an RRC Release command, immediately disconnecting the compromised User Equipment (UE).

Experimental results validate the effectiveness of our system. It achieves accurate, real time DDoS detection with a low false positive rate and consistently low latency, even under attack conditions, while maintaining high uplink throughput. This demonstrates the framework's ability to preserve performance for legitimate users. Overall, this work presents a practical and scalable architecture that significantly strengthens the security and resilience of 5G networks against emerging cyber threats.

Keywords: 5G Security, O-RAN, DDoS , Machine Learning, OpenAirInterface, FlexRIC.

Résumé

L'intégration de l'architecture O-RAN dans les réseaux 5G marque un tournant majeur dans le domaine des communications mobiles, en introduisant ouverture, programmabilité et contrôle intelligent via des RAN Intelligent Controllers (RIC) centralisés. Inspirée des principes du Software Defined Networking (SDN), l'innovation fondamentale de l'O-RAN réside dans la séparation entre la logique de contrôle et le plan de données, permettant une gestion dynamique, fine et intelligente du réseau. Toutefois, ces avancées technologiques s'accompagnent de nouveaux défis de sécurité, notamment la montée en puissance des attaques par déni de service distribué (DDoS). Afin de répondre à ces vulnérabilités, ce projet propose un cadre intelligent pour la détection et la mitigation en temps réel des attaques DDoS au sein d'un réseau 5G de nouvelle génération basé sur l'O-RAN.

Au cœur de notre solution se trouve un banc d'essai 5G entièrement conteneurisé et compatible avec le slicing, conçu à l'aide de OpenAirInterface (OAI) et orchestré via le contrôleur FlexRIC. Cette architecture offre une gestion réseau dynamique et programmable, avec un contrôle précis de la connectivité des utilisateurs et du découpage réseau (network slicing) au niveau de la RAN.Pour renforcer les capacités de détection, nous avons conçu et évalué une suite de modèles d'apprentissage automatique, incluant Random Forest, Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM) ainsi qu'une approche par ensemble (ensemble learning). Parmi ces modèles, Random Forest a montré les meilleures performances, atteignant 99,0% de précision en phase d'apprentissage et 90% sous trafic réel simulé, tout en conservant une empreinte computationnelle réduite.Le système de détection a été intégré à la User Plane Function (UPF), permettant une analyse du trafic en temps réel et la détection d'anomalies. En cas d'activité malveillante, une xApp dédiée, déployée dans le Near-RT RIC via FlexRIC, déclenche automatiquement une commande RRC Release, déconnectant immédiatement l'équipement utilisateur (UE) compromis.

Les résultats expérimentaux confirment l'efficacité de notre système. Il permet une détection précise et en temps réel des attaques DDoS avec un faible taux de faux positifs et une latence constamment réduite, même en conditions d'attaque, tout en maintenant un débit élevé en liaison montante. Cela démontre la capacité du framework à préserver les performances pour les utilisateurs légitimes. Globalement, ce travail propose une architecture pratique et évolutive qui renforce significativement la sécurité et la résilience des réseaux 5G face aux cybermenaces émergentes.

Mots-clés: Sécurité 5G, O-RAN, attaque DDoS, apprentissage automatique, OpenAirInterface, FlexRIC.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
5G	5th Generation
AMF	Access and Mobility Management Function
ANNs	Artificial Neural Networks
AUSF	Authentication Server Function
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DDoS	Distributed Denial of Service
DL	Deep Learning
DN	Data Network
eMBB	Enhanced Mobile Broadband
gNB	gNodeB (5G base station)
GTP-U	GPRS Tunneling Protocol - User Plane
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
LSTM	Long Short-Term Memory
MAC	Medium Access Control
ML	Machine Learning
mMTC	Massive Machine-Type Communications
NGAP	Next Generation Application Protocol
NRF	Network Repository Function

NSSF	Network Slice Selection Function
O-CU	Open Centralized Unit
O-DU	Open Distributed Unit
O-RAN	Open Radio Access Network
O-RU	Open Radio Unit
OAI	OpenAirInterface
PCAP	Packet Capture (File Format)
PCF	Policy Control Function
PDU	Protocol Data Unit
PRB	Physical Ressource Block
QoS	Quality of Service
RAN	Radio Access Network
RC	RAN Control
RIC	RAN Intelligent Controller
RLC	Radio Link Control
RRC	Radio Resource Control
SMF	Session Management Function
SMO	Service Management and Orchestration
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable Low Latency Communications
xApps	External Applications (in O-RAN architecture)

Chapter 1

GENERAL INTRODUCTION

1.1 Introduction

In the era of hyper connectivity, 5G networks form the backbone of next generation digital infrastructure, delivering ultra fast speeds, low latency, and the capacity to support billions of connected devices. These capabilities power transformative applications such as autonomous vehicles, smart factories, and remote healthcare. Unlike previous generations, 5G is not just a communication upgrade, it represents a complete architectural shift, designed to support the integration of digital and physical systems in real time [1].

However, with this transformation comes a greater exposure to cybersecurity risks. As 5G becomes deeply embedded in critical services, it becomes a lucrative target for cyber threats, particularly Distributed Denial of Service (DDoS) attacks. These threats exploit the decentralized, high bandwidth nature of 5G to overload network functions, disrupt services, and potentially cause large scale operational failures [1, 2].

A critical target of such attacks is autonomous vehicles and Vehicle-to-Everything (V2X) systems, which depend on real time alerts to avoid collisions. DDoS related delays can disrupt these communications, leading to severe accidents. Ensuring low latency, reliable connectivity is therefore essential to avoid these problems.

Traditional security mechanisms often fall short in this environment due to their static nature and limited adaptability. The dynamic features of 5G, including massive device connectivity, network slicing, and edge computing, expand the attack surface and require new, more flexible defenses [1].

To address these challenges, 5G architecture is evolving through the adoption of the Open Radio Access Network (O-RAN) paradigm. Inspired by Software Defined Networking(SDN), O-RAN introduces modularity and programmability by decoupling the control and data planes within the Radio Access Network [3]. This design splits traditional RAN elements into distinct, interoperable components that are orchestrated by an intelligent RAN controller, the near Real-Time RIC, which enables the deployment of custom applications known as xApps that can dynamically monitor, analyze, and control RAN behavior.

Crucially, the programmability of O-RAN provides a natural foundation for incorporating machine learning into network operations in a meaningful way. Instead of relying solely on fixed rules or manual oversight, machine learning can help the network recognize patterns, spot unusual behavior, and respond to potential threats more intelligently. In the context of this work, machine learning is used to enhance the detection of DDoS attacks making it possible to react faster and more effectively as threats emerge.

This thesis proposes a machine learning (ML) based security framework that leverages the flexibility of O-RAN to detect and mitigate DDoS attacks in real time. We implement and evaluate this framework using a fully containerized 5G Network based on OpenAirInterface (OAI) [4], extended with the FlexRIC controller [5] to support near-RT RIC operations. This setup provides a realistic and modular environment for simulating real world network behavior and evaluating intelligent, adaptive security mechanisms.

By combining the openness of O-RAN with the analytical power of machine learning, this research aims to make real time, intelligent threat detection and response a native capability of future mobile networks ultimately ensuring the resilience, safety, and reliability of critical digital services.

1.2 Research Motivation

The evolution of 5G networks has introduced not only faster and more reliable connectivity but also a shift in how networks are architected and managed. A major innovation in this transformation is the O-RAN, which embraces openness, modularity, and programmability by decoupling traditional monolithic RAN components. This openness is a game changer, enabling innovation, vendor interoperability, and the deployment of custom xApps for real time control and optimization.

However, this flexibility comes at a cost. The O-RAN architecture widens the attack surface by exposing more interfaces, increasing software complexity, and introducing new trust boundaries. Unlike traditional, closed RAN systems, O-RAN components can be targeted independently, and security can no longer rely solely on isolation or proprietary implementations [6].

Simultaneously, With the growth of 5G networks, the amount and variety of data traffic have increased significantly, coming from many types of devices and services. This traffic is not only larger in volume but also faster and more dynamic, making it much harder to monitor and protect. Traditional security methods are no longer effective because they cannot keep up with the speed and complexity of 5G traffic. This is especially true for DDoS attacks, which have become more difficult to detect and stop in real time, as the malicious traffic often blends in with normal network activity.

These challenges highlight the pressing need for smarter and more adaptive defense mechanisms, especially ones that can operate within the open and programmable nature of O-RAN. Leveraging Machine Learning (ML) and Deep Learning (DL) presents a promising direction for understanding complex traffic behavior and identifying malicious patterns more effectively. With the right integration, these intelligent techniques can enhance the overall security of next generation networks without compromising performance or flexibility.

1.3 Research Objectives

The primary objective of this research is to design, implement, and validate an intelligent machine learning-based framework for the detection and mitigation of DDoS attacks in O-RAN Enabled 5G Networks. The proposed framework will leverage real time data processing capabilities to significantly enhance network security.

To achieve this aim, the research is structured around the following specific objectives:

1. **Emulate a 5G Slicing Network environment :** Develop and deploy an (O-RAN) 5G slicing environmen using OpenAirInterface (OAI), integrated with the FlexRIC controller, to simulate realistic network environment and enable dynamic control and monitoring.

2. **Construct and Preprocess an Enhanced Dataset for Model Training:** Augment the CICDDoS2019 benchmark dataset by injecting emulated 5G traffic captured from the simulated 5G network, creating a more representative dataset (CICDDoS-5G). The enhanced dataset will undergo thorough preprocessing to extract relevant features from both benign and malicious traffic, ensuring its suitability for training effective ML/DL models.

3. **Develop and Implement ML/DL Algorithms:** Design and apply advanced ML and DL algorithms tailored for DDoS detection in 5G traffic. These models will be optimized to handle the complex patterns inherent to 5G networks.

4. **Design a Real Time Detection System:** Create and implement a system capable of capturing live network traffic and detecting potential DDoS attacks in real time using the trained models.

5. **Implement Mitigation Mechanisms:** Develop and deploy automated strategies to mitigate detected DDoS attacks, while ensuring minimal disruption to legitimate network services.

6. **Evaluate Model and System Performance:** Conduct a comprehensive evaluation of both the trained models and the overall defense framework using key performance metrics, and the impact on network performance.

1.4 Project Structure

This project is structured into four key chapters, each building towards an effective solution for DDoS detection and mitigation in O-RAN enabled 5G networks:

- **Chapter 2** provides the foundational context, introducing 5G and O-RAN architectures, DDoS attack taxonomy, and relevant ML/DL models. It also includes a review of related works, highlighting the latest research in the field.

- **Chapter 3** outlines the simulation framework, enhanced dataset preparation, and training models (Random Forest, 1D-CNN, BiLSTM, and ensemble). It also presents the integration of a real time detection and mitigation system in simulated 5G environment.

- **Chapter 4** presents a comparative analysis of model performance and evaluates the defense system within the 5G simulated environment.

- **Chapter 5** concludes the study by summarizing achievements and proposing future research directions to advance intelligent DDoS defense.

Chapter 2

BACKGROUND AND RELATED WORK

2.1 Introduction

This chapter provides an overview of the key concepts and technologies relevant to our study. It begins by describing the 5G network architecture and the principles of O-RAN and network slicing, which are essential for understanding the system's structure and challenges. It then explores the growing threat of DDoS attacks in 5G environments and discusses the limitations of traditional detection methods. To address these issues, the chapter introduces machine learning and deep learning techniques as promising solutions and also covers key ML concepts such as activation functions, loss functions, regularization techniques, and optimization algorithms, which are essential for building accurate detection systems in the complex and high speed environment of 5G networks. Finally, it reviews recent research in the field to identify existing approaches and limitations, helping us build a clear roadmap for the design and implementation of our proposed solution.

2.2 5G Network Architecture

The 5G network architecture introduces a modern and flexible framework designed to support high-speed data transmission, ultra-low latency, and massive device connectivity. Developed under the guidance of the 3rd Generation Partnership Project (3GPP), it was introduced globally in 2019 [7]. Unlike previous generations, 5G is based on a Service-Based Architecture (SBA), software-defined networking (SDN) and network function virtualization (NFV) for enhanced scalability [8].

5G networks rely on two main frequency ranges to deliver versatile connectivity. Frequency Range 1 (FR1), also known as Sub-6 GHz which serves as the backbone of 5G, balancing speed and coverage. Its low-band frequencies (below 1 GHz) stretch across vast rural areas and support massive IoT deployments, by prioritizing reliability over speed. Mid-band frequencies (2–6 GHz), strike a sweet spot—offering faster speeds for urban smartphone users and connected cars while maintaining reasonable coverage. the second range (mmWave,FR2) operates at 24 GHz and above, acting as a speed powerhouse for dense cities. though its signals struggle with obstacles like walls, requiring dense networks of small cells. Together, these frequencies allow 5G to meet diverse needs: seamless video calls, mission-critical factory automation via ultra-reliable low-latency links, and

blazing-fast mmWave for immersive tech. [9].

It is broadly divided into three main sections: the User Equipment (UE), the Radio Access Network (RAN), and the Core Network.

2.2.1 User Equipment

User Equipment refers to any device that connects to the 5G network, such as smartphones, IoT sensors, tablets, and autonomous vehicles. The UE connects to the RAN via advanced wireless technologies like massive MIMO and beamforming [1].

2.2.2 Radio Access Network

The 5G RAN is the intermediary layer between the UE and the 5G Core (5GC). The primary component of the RAN is the gNodeB, which connects devices to the network. Its key functionalities include UE admission control, radio resource management, mobility control, routing of user/control-plane packets, Quality of Service (QoS) management, and support for network slicing [10].

O-RAN Architecture Overview

Traditional RAN architectures have been reliable but face major drawbacks in the 5G era due to their rigid, vendor-specific designs, where hardware and software are tightly coupled and controlled by a single manufacturer. This limits flexibility, slows innovation, and prevents operators from customizing their networks. In response to these challenges, The Open RAN was introduced to promote openness, interoperability, and intelligence in the RAN domain, breaking down the RAN into modular components like the Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU) with open interfaces between them, allowing different vendors to work together. One of the key innovations in O-RAN is the introduction of the RAN Intelligent Controller (RIC), which allows operators to run real-time optimization apps(xApps) for tasks like mobility management, load balancing, and even DDoS mitigation. This open and programmable framework empowers network operators with unprecedented control and visibility over the RAN, enhancing both performance and adaptability in 5G networks.

As shown in Figure 2.1, the key elements of the O-RAN architecture are:

- **Service Management and Orchestration (SMO):** The SMO is the control center of the O-RAN system. It's responsible for managing and coordinating all the parts of the network, including the RAN components (RU, DU, CU), the RICs (Near-RT and Non-RT), and the open interfaces between them.

- **Non-Real-Time RAN Intelligent Controller (Non-RT RIC):** The Non-RT RIC is part of the Service Management and Orchestration (SMO) framework. which operates on a timescale of more than 1 second, and focuses on non-real-time tasks like AI model training, network analytics, and orchestration.It hosts rApps, which gather performance metrics, train models, and issue strategic policies via the A1 interface to the Near-RT RIC [11].

- **Near-Real-Time RAN Intelligent Controller (Near-RT RIC):** This controller operates on a 10ms to 1s timescale and enables real-time optimization of

RAN elements like the O-CU and O-DU through the E2 interface. It hosts xApps, which are software modules that can be deployed to perform that perform dynamic tasks such as radio resource management, and network slicing assurance using AI/ML models and UE/cell metrics [11].

- **Open Centralized Unit (O-CU):** At the top of the RAN stack lies the O-CU, which performs higher-layer RAN functions such as radio resource control (RRC), packet data convergence protocol (PDCP), and service data adaptation (SDAP). These layers manage session handling, QoS enforcement, and secure communication. The O-CU is typically split into two subcomponents,**O-CU-CP** (control plane) and **O-CU-UP** (user plane), which communicate over the E1 interface. This split allows for independent scaling and optimization of control and user traffic. Importantly, both the O-DU and O-CU are designed to support network slicing, enabling the creation of virtual RAN slices tailored to specific service requirements. Through their modular design and standardized interfaces, the O-RU, O-DU, and O-CU collectively bring openness, scalability, and multi-vendor interoperability to the 5G RAN ecosystem [11].

- **Open Distributed Unit (O-DU):** The O-DU is responsible for baseband processing and implements critical functions of the RAN protocol layers, including the high-PHY, MAC (Medium Access Control), and RLC (Radio Link Control). It serves as the main processing unit that allocates radio resources, schedules transmissions, and maintains radio link reliability. it connects to the O-CU via the F1 interface [11].

- **Open Radio Unit (O-RU):** The O-RU is positioned closest to the antenna and is tasked with handling the transmission and reception of radio signals over the air interface. It performs low-level physical layer (PHY) functions such as digital-to-analog conversion, beamforming, and signal amplification. The O-RU connects to the O-DU through the open fronthaul interface, allowing flexibility in vendor selection and deployment scenarios [11].

- **Open Interfaces (e.g., E2, A1, O1):** These standardized interfaces connect the various O-RAN components. For instance, the E2 interface links the Near-RT RIC to CU/DU, the A1 interface connects the Non-RT RIC to the Near-RT RIC, and the O1 interface allows the SMO to manage network elements [11].

2.2.3 Core Network Functions

The 5G Core (5GC) adopts a Service-Based Architecture (SBA), where network functions (NFs) are split up by service and communicate using APIs rather than rigid, predefined interfaces [8]. The functions of the 5G Core include:

- Access and Mobility Management Function (AMF)

  The AMF is responsible for managing user mobility, registration and network access. it serves as the main entry point for user connections, interacting with the Authentication Server Function (AUSF) to authenticate devices [2].

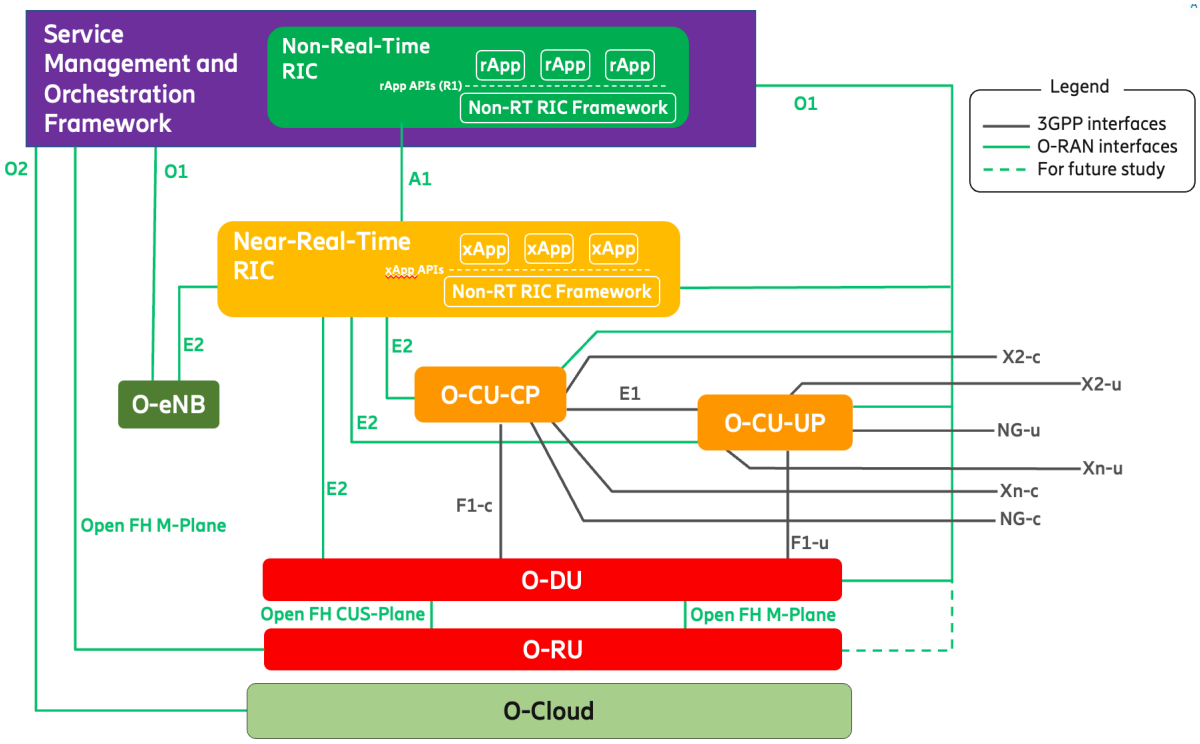


Figure 2.1: O-RAN Architecture [3]

- Session Management Function (SMF)

  The SMF handels subscriber session management,including IP address allocation, data session control and mobility management. It also works closely with the Policy Control Function (PCF) to enforce network policies and Quality of Service (QoS) rulescite [2, 12].

- User Plane Function (UPF)

  The UPF is responsible for routing and forwarding user data packets between the 5G Core and external Data Networks (DN) such as the internet [2].

- Policy Control Function (PCF)

  This network function controls integrated policy enforcement across the entire 5G Core network. It provides policy rules to other control plane Functions, ensuring optimal network performance [2].

- Network Slice Selection Function (NSSF)

  The NSSF select the most suitable network slice based on the service requirements of the user. It provides AMF selection assistance by ensuring that the requested services are directed to the correct network slice. [2]

- Authentication Server Function (AUSF)

  The AUSF Stores authentication keys for Users accessing the 5G Core network. It collaborates with UDM to store and validate user credentials [2].

- Unified Data Management (UDM)

  UDM is responsible for data management .including managing user subscriptions,storing authentication credentials [2].

- Network Repository Function (NRF)

  The NRF maintains a registry of all 5G Core Network Functions (NFs), allowing dynamic service discovery. providing interworking mechanisms such as authentication token management and service monitoring [2].

- Network Exposure Function (NEF)

  The NEF is responsible for storing internal and external service functions. It also exposes securely network capabilities and events to the application Function (AF) and enables the AF to access these services [2].

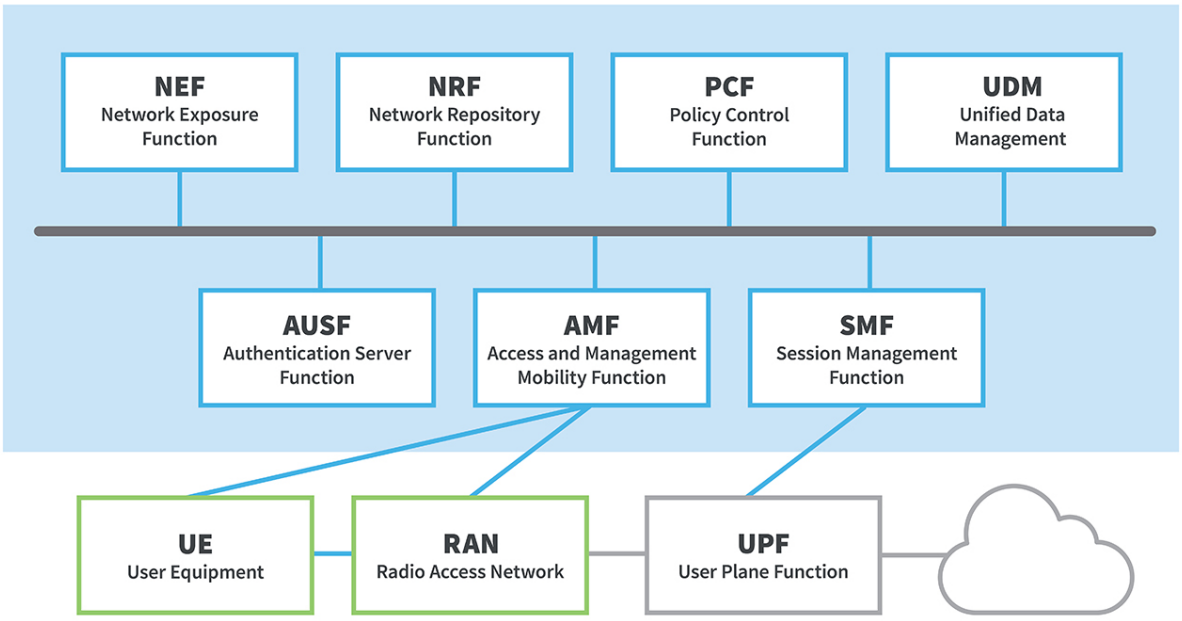


Figure 2.2: 5G Architecture [13]

2.3 5G Network Slicing

5G network slicing is one of the most transformative features introduced in the fifth generation of mobile networks. It enables a single physical infrastructure to be logically partitioned into multiple virtual networks, called *slices*, where each slice is tailored to serve a specific type of service, application, or industry requirement. This allows telecom operators to run multiple isolated network environments on the same shared physical resources, each with its own architecture, quality of service (QoS), and security mechanisms [12].

According to the 3GPP standard, 5G network slicing is typically categorized into three main types, each corresponding to a major family of use cases:

- **Enhanced Mobile Broadband (eMBB)**: This slice is designed for high data rates and large bandwidth applications such as ultra-HD video streaming, virtual reality (VR), and high-speed internet access. eMBB focuses on throughput and user experience in densely populated areas like stadiums or city centers [12].

- **Ultra-Reliable Low-Latency Communications (URLLC)**: This slice supports mission-critical applications that require extremely low latency and very high reliability. Use cases include autonomous vehicles, remote surgery, industrial automation, and real-time control systems. URLLC slices prioritize stability and response time over data throughput [12].

- **Massive Machine-Type Communications (mMTC)**: Also known as massive IoT, this slice type supports connectivity for a large number of low-power, low-data-rate devices such as sensors, smart meters, and environmental monitors. This slice is optimized for scalability and energy efficiency, rather than speed or latency [12].

the complexity and dynamic nature of 5G slices can introduce new cybersecurity challenges. Since multiple slices often coexist on shared infrastructure, a vulnerability in one slice could potentially affect others especially if isolation mechanisms fail or are misconfigured.

2.4 5G Network Vulnerabilities

5G networks introduce significant advancements but also new security challenges. Below are key vulnerabilities that make 5G networks susceptible to DDoS attacks:

- **Massive IoT Device Proliferation:** 5G supports an extensive number of IoT devices, many of which lack robust security mechanisms. This makes them easy targets for attackers who can compromise IoT devices to form large-scale botnets. These botnets can then launch volumetric DDoS attacks, overwhelming 5G network resources, leading to congestion and service degradation [14].

- **Distributed Architecture with Multi-access Edge Computing (MEC):** 5G relies on a highly distributed network architecture with Multi-access Edge Computing (MEC) at remote locations. Each MEC node becomes a potential DDoS target, especially since it holds low-latency and latency-sensitive services (like V2V communication) [14].

- **Service-Based Architecture (SBA) and Signaling:** Unlike 4G LTE, where control tasks were centralized, 5G relies on a distributed Service-Based Architecture (SBA), using HTTP/2-based signaling between Network Functions (NFs). While this enhances scalability, it also increases the attack surface, making the network susceptible to signaling DDoS attacks. Attackers can flood the network with fake session registration, attach/detach requests, or service discovery messages, overwhelming critical NFs like AMF, SMF, and AUSF [2, 8].

- **5G Network Slicing:** One of the most significant innovations in 5G, network slicing, allows the creation of multiple virtual networks on shared infrastructure to support different services. However, this introduces new security challenges. A DDoS attack targeting one slice can exhaust shared network resources, indirectly impacting other slices [12].

- **GTP-U Transport Protocol:** GTP-U (GPRS Tunneling Protocol - User Plane) is a protocol used to transport user data between the gNodeB and the User Plane

Function (UPF) in the 5G core network.it does not verify whether the traffic source is legitimate or not .So any attacker generating GTP-U traffic can impersonate a valid UE and overwhelm the UPF [15]

- **O-RAN Architecture and its Open Interfaces:** The O-RAN architecture, while enabling openness and intelligent control via RAN Intelligent Controllers (RIC), also introduces new DDoS vulnerabilities due to its disaggregated nature. By separating traditional monolithic RAN components into modular and virtualized units (O-RU, O-DU, O-CU) and exposing standardized open interfaces such as E2 and A1, O-RAN expands the potential attack surface. These open interfaces, although crucial for programmability and multi-vendor interoperability, can be exploited by attackers to inject malicious traffic. A DDoS attack targeting the E2 interface can overwhelm the communication channel between the RIC and the RAN nodes, saturate network resources, and disrupt the real-time control loop. The RIC, which is central to managing functions like mobility, scheduling, and handover decisions, becomes a critical point of failure when flooded with DDoS Attacks, leading to degraded performance or complete loss of control in the RAN [6, 16].

2.5 DDoS Attacks on 5G Networks

A Distributed Denial of Service (DDoS) attack is one of the most widespread attacks that attempt to disrupt the availability of a targeted network by overwhelming it with a flood of excessive traffic. Unlike a standard Denial of Service (DoS) attack, which originates from a single source, a DDoS attack leverages a network of infected devices controlled by an attacker (a botnet). The goal is to exhaust network resources like bandwidth, CPU, or session capacity by flooding the network with malicious traffic, often using massive botnets made up of compromised IoT devices, rendering it unavailable to legitimate users [17, 18].

As 5G networks promise unprecedented speed, capacity, and low latency, they also introduce new vulnerabilities, especially to DDoS attacks. In the 5G era, the situation becomes even more critical: the proliferation of connected devices, and the use of cloud-native, software-defined infrastructure all contribute to a dramatically expanded attack surface. DDoS attacks are now not only more frequent, but also more sophisticated often combining multiple attack vectors such as SYN floods, UDP amplification, and ICMP floods. They can reach speeds of hundreds of gigabits per second, making them hard to detect and even harder to mitigate in real time. Traditional defenses struggle to cope with these evolving threats, which is why researchers are increasingly exploring the integration of Machine Learning to enable more intelligent defense mechanisms tailored for the 5G environment [18].

2.5.1 Types of DDoS Attacks

DDoS attacks are broadly categorized into three types, each targeting different layers of a network:

- **Volumetric Attacks**

  A volumetric DDoS attack is the most well-known and widespread type of DDoS attacks.

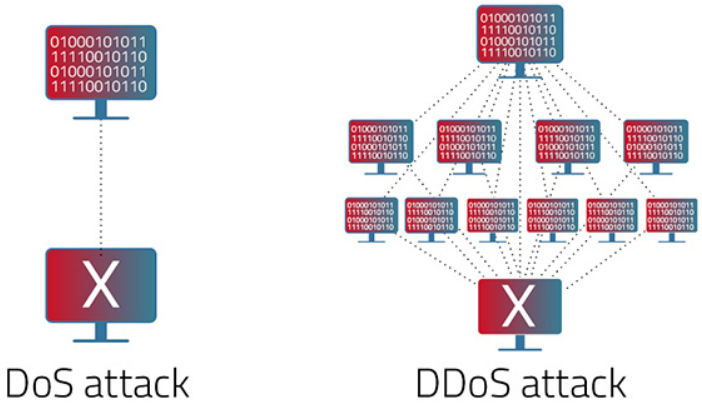


Figure 2.3: DoS/DDoS Attacks [19]

The principle behind a volumetric attack is straightforward: saturate the target's bandwidth by sending an enormous volume of traffic. Attackers often achieve this using amplification techniques, with DNS amplification being one of the most commonly used methods. these attacks can also be created using botnets made up of exploited IoT devices. Connected devices usually lack basic security defenses, but because they're connected to the Internet and can execute code, they can be easily exploited. the attack's intensity is measured in bits per second (bps) or gigabits per second (Gbps) [17].

UDP Flood

A UDP flood is a volumetric attack where the attacker sends a large number of UDP packets to random ports on the target system. Because UDP is connectionless, the server must process each packet, often responding with ICMP "Destination Unreachable" messages. The attacker typically spoofs source IP addresses in the UDP packets, ensuring that all ICMP responses are sent to a fake address instead of the attacker. This will lead to a Denial-of-Service (DoS) condition that blocks legitimate traffic [17].

- **Network Protocol Attacks**

  Protocol Attacks are a type of DDoS attacks that target vulnerabilities in Layer 3 and Layer 4 of the OSI model, such as flaws in TCP/IP protocols. By flooding targets with malicious requests (e.g., SYN flood) or malformed packets (e.g., Ping of Death), they exhaust the target's ressources causing systems to become unresponsive [17].

  SYN Flood

  An example of such an attack is the SYN flood attack that aims to exhaust a server's connection resources by exploiting the TCP three-way handshake process through sending a high volume of SYN (synchronize) packets to the target server, often with spoofed IP addresses. In response, the server sends SYN-ACK packets and waits for the final ACK to complete the connection. However, the attacker never sends the ACK, leaving numerous connections half-open. As the server continues to allocate resources for these incomplete connections, its memory, CPU, and open ports become overwhelmed. Once all available connection slots are occupied, the

server can no longer process legitimate requests, resulting in a Denial of Service (DoS) condition [17].

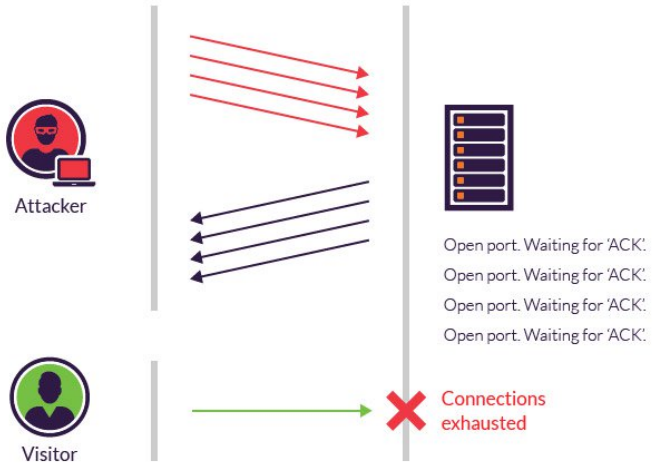


Figure 2.4: SYN Flood Attack [20]

- **Application Layer Attacks** While volumetric attacks and protocol attacks compromise a service with the sheer number of requests, application layer attacks, target an edge server that executes a web application. These threats are harder to detect because attackers usually make requests like legitimate users. Consequently, these attacks often show up as smaller traffic spikes and do not require the assistance of a botnet [17].

  HTTP Flood
  HTTP flood attacks belong to Layer 7 DDoS attacks, which exploit the HTTP protocol used for loading webpages and processing online forms. Unlike network layer attacks, Layer 7 attacks are more difficult to mitigate because malicious traffic closely resembles normal user behavior. To maximize their impact, attackers often use botnets to generate massive volumes of HTTP requests aimed at overwhelming the target [17].

2.6 Traditional Detection Methods

The deployment of 5G networks marks a significant evolution in mobile communications, enabled by advanced technologies such as network slicing. These innovations enhance flexibility and scalability, but simultaneously broaden the attack surface, introducing new and complex security vulnerabilities. Among the most critical threats in this context are DDoS attacks, which exploit the dynamic and decentralized nature of 5G infrastructures to overwhelm network resources with illegitimate traffic. This can severely degrade service quality, disrupt availability, or even cause complete outages in targeted slices or core components. Given the potential impact on network performance, there is a pressing need for robust and adaptive detection and mitigation strategies. Traditionally, DDoS defense is organized into three key stages: detection, source identification, and response. Detection involves identifying abnormal traffic patterns or behavioral anomalies indicative of an attack. Source identification aims to trace the origin of malicious traffic, which is

particularly challenging in distributed environments. Finally, response mechanisms are implemented to neutralize the threat in real time by blocking malicious or spoofed traffic as close to the source as possible [21].

In the following section, we examine and discuss various techniques used for detecting DDoS attacks.

2.6.1 Static Methods

Static methods constitute one of the earliest approaches to detecting DDoS attacks. These techniques operate based on predefined statistical rules to identify abnormal traffic behaviors. Rather than learning from historical data, they use mathematical models and statistical tests to compare real-time network traffic against a reference baseline of normal activity.

A widely adopted technique in this category is entropy analysis, which quantifies the degree of randomness or unpredictability within specific traffic attributes. In typical network conditions, fields such as source IP addresses or destination ports exhibit a relatively stable distribution. However, a sudden spike or drop in entropy may indicate a DDoS attempt. For instance, an abrupt increase in the diversity of source IP addresses could suggest a botnet-based attack, while a drop in entropy may reflect a volumetric attack originating from a single source. Another commonly used static technique is the chi-square test, which measures how closely the current distribution of selected traffic features aligns with a predefined baseline. Significant deviations between the observed and expected distributions may signal malicious activity. For example, an unexpected surge in traffic directed to a specific port could be flagged through this method [22].

2.6.2 Threshold-Based Detection

Threshold-based detection identifies potential DDoS attacks by observing network or system activity and comparing it to established normal patterns. If certain metrics like traffic volume or connection requests go beyond set threshold limits, it indicates abnormal behavior that might signify an attack. These limits are typically learned over time from regular, attack-free operations. While this method helps uncover new or evolving attacks, it can also produce false alarms if thresholds are set too low or miss attacks if they're set too high [23].

2.6.3 Signature-Based Detection

Signature-based detection is a widely used technique in intrusion detection systems (IDS), where incoming traffic is analyzed against a library of predefined attack signatures. These signatures are built from distinct packet attributes that have been associated with past malicious activities, such as abnormal protocol usage, fixed payload sizes, or repeated patterns in packet structure. While this method can effectively detect known DDoS attack types, it struggles to identify new or slightly modified threats due to its reliance on previously observed behaviors. Moreover, if attackers use encryption to conceal botnet communications, this approach becomes ineffective, as it cannot interpret the encrypted data [21, 24].

However, the high-speed and service-oriented nature of 5G networks necessitates intelligent and automated defense strategies capable of adapting to evolving attack vectors.

This limitation of conventional methods has driven increasing interest in ML and DL approaches, which offer the potential for real-time, data-driven decision-making and early detection of sophisticated threats.

2.7 Modern Detection Methods

The increasing frequency and complexity of DDoS attacks demand the implementation of more intelligent, adaptive, and dynamic security solutions. In this context, Machine Learning (ML) has emerged as one of the most promising approaches for attack detection. Unlike traditional detection systems, which are often based on predefined rules or static signatures, ML algorithms are capable of analyzing large volumes of data in real time and automatically learning both normal and abnormal system behaviors. This enables them to detect not only known attacks but also new and evolving threats that may bypass conventional methods [25].

2.7.1 Machine Learning Overview

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) in which machines use algorithms to make predictions or classifications by learning from data, whether labeled or unlabeled [26, 27]. These algorithms can range from simple methods like linear regression to more complex ensemble techniques. They identify patterns, make decisions, and improve their performance over time and in real-world scenarios, all without the need for explicit programming [27].

Deep Learning (DL) is a branch of ML that leverages the architecture of multilayered Artificial Neural Networks (ANNs), which are computational models inspired by the functioning of biological neurons [28]. A typical neural network consists of three main components: an input layer, one or more hidden layers, and an output layer. Each layer is composed of interconnected processing units known as neurons.

The input layer receives raw data, where each neuron corresponds to a specific feature extracted from the dataset. These features are then propagated through the hidden layers, where complex nonlinear transformations are applied. Each neuron computes a weighted sum of its inputs, passes it through an activation function (such as ReLU, sigmoid, or tanh), and outputs a value that is forwarded to the next layer. If the output exceeds a certain threshold, the neuron is activated and contributes to subsequent computations.

The output layer produces the final prediction, which can be a classification (e.g., attack vs. benign traffic) or a regression output depending on the application. Learning in neural networks occurs through a process called backpropagation, where errors between the predicted and actual outcomes are propagated backward to adjust the weights, minimizing the loss function over successive training iterations [29].

2.7.2 Random Forest

Random Forest is an ensemble learning method used for classification and regression that builds multiple decision trees during training, combining their predictions through majority voting (classification) or averaging (regression). Each tree is trained on a random subset of the data (bootstrap aggregating) and considers only a random subset of features at each split, enhancing diversity and reducing overfitting. This approach improves generalization, with the model's error converging to a limit as the number of trees increases,

ensuring strong performance even with high-dimensional data, though this comes with reduced interpretability compared to single decision trees or linear models [30]. Random Forests mitigate overfitting by combining diverse trees trained on randomized subsets of data and features [31]. In the field of cybersecurity, Random Forest has proven to be particularly effective for detecting DDoS attacks [31].

2.7.3 Convolutional Neural Networks

The Convolutional Neural Network is one of the most prominent architectures in the field of deep learning [32]. CNN have shown remarkable effectiveness in a variety of domains, including image reconstruction [33] and natural language processing [34]. In recent years, CNN have also gained increasing attention in the field of cybersecurity [35], particularly for critical applications such as DDoS attack detection [25,36]. By leveraging components such as convolutional, pooling, and fully connected layers, CNN can automatically learn spatial feature hierarchies. These powerful pattern recognition capabilities make CNN especially suitable for analyzing complex datasets, including traffic in 5G networks [37].

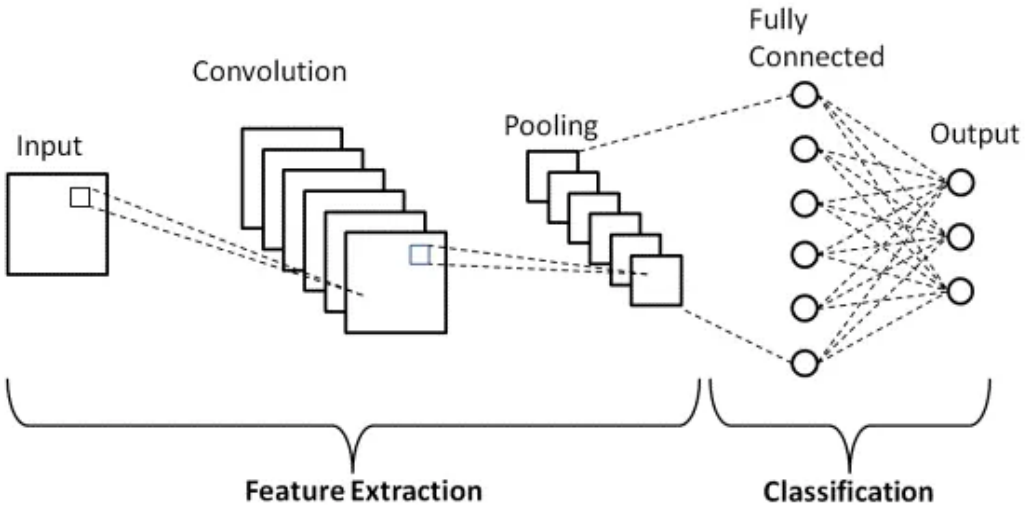


Figure 2.5: CNN Architecture [38]

As depicted in Figure 2.5 , the convolutional layers play a fundamental role in feature extraction. These layers utilize a set of trainable filters, also referred to as kernels, which slide across the input data to identify local patterns such as edges, contours, and textures. At each position, the filter performs an element-wise multiplication with the corresponding segment of the input, followed by a summation. The result of this operation is a feature map, which captures both spatial and contextual information present in the input. To introduce non-linearity into the model, each feature map is subsequently passed through an activation function—most commonly the Rectified Linear Unit (ReLU). This non-linear transformation enables the network to learn and represent more complex patterns [39]. Notably, the convolutional layer benefits from sparse connectivity and shared weights, which contribute to a significant reduction in both computational cost and memory requirements [40].

After the convolutional and activation phases, pooling layers are introduced to reduce the spatial dimensions of the feature maps. This process, known as downsampling, helps to minimize the number of parameters and operations in the network, while retaining

the most critical features. Max pooling, which selects the highest value within a defined window, and average pooling, which computes the mean, are the most frequently used strategies [40].

In the final stages of the CNN , the output of the convolutional and pooling layers is passed to one or more fully connected layers. These layers are responsible for aggregating the extracted features and making final predictions. For classification tasks, the network typically concludes with a SoftMax output layer, which produces a probability distribution over the target classes, thereby yielding the final decision of the model [39].

2.7.4 Bidirectional Long Short-Term Memory

Bidirectional Long Short-Term Memory networks are an advanced extension of traditional models, designed to capture both past and future context in sequential data. While standard LSTM networks process information in a single direction—from past to future—BiLSTM incorporates an additional LSTM layer that reads the input sequence in reverse. This dual-processing mechanism enables the network to have a more comprehensive understanding of the entire sequence, which is especially valuable in tasks where context on both sides of a token is essential, such as in speech recognition, sentiment analysis, or named entity recognition [41, 42]. BiLSTM has been successfully applied in the detection of cyberattacks, including DDoS attacks, intrusions, and anomalous traffic behavior in networks [43].

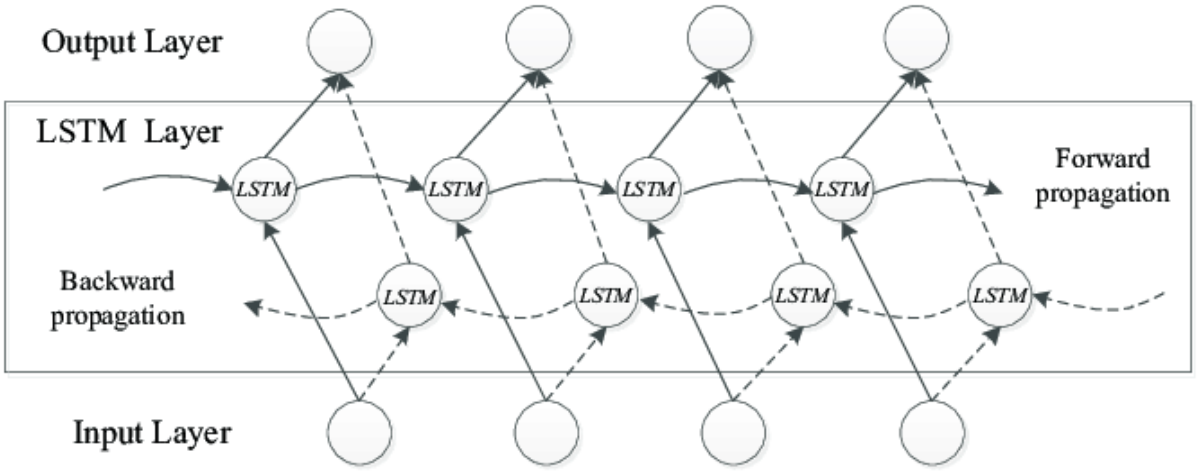


Figure 2.6: BiLSTM Architecture [44]

The architecture of a BiLSTM consists of two LSTM layers, as depicted in Figure 2.6, running in parallel. The forward LSTM processes the input sequence in its natural order (from the first to the last time step), while the backward LSTM reads the same sequence in reverse (from the last to the first time step). At each time step, the outputs from both directions are concatenated or combined, resulting in a richer representation that incorporates information from both past and future contexts. This feature distinguishes BiLSTM from unidirectional models, which rely solely on historical inputs for prediction.

Internally, each LSTM unit within the forward and backward layers contains memory cells and three types of gates—input, forget, and output gates—that regulate the flow of information through time. These gates determine which information to keep, forget, or pass to the next state. By doing so, LSTM networks are able to preserve relevant

information over long sequences, avoiding issues like vanishing gradients that typically affect vanilla Recurrent Neural Networks [45].

The output of a BiLSTM layer at each time step is a vector that merges the outputs of both the forward and backward LSTM. This enriched representation is particularly effective for tasks requiring detailed sequence labeling or classification. For example, in the BiLSTM-CRF architecture, a Conditional Random Field (CRF) layer is often added on top of the BiLSTM to model dependencies between output labels, thereby improving sequence prediction accuracy [46].

2.7.5 Activation Functions

An activation function is a mathematical transformation applied to the output of each neuron in ANN models. Its primary role is to introduce non-linearity into the model, allowing the network to learn complex relationships between input data and target outputs. Without a non-linear activation function, a deep network would behave like a simple linear model, incapable of solving complex tasks such as image recognition or sequence prediction [47]. There are several types of activation functions; the most commonly used are presented below.

ReLU (Rectified Linear Unit)

ReLU is the most widely used activation function in the hidden layers of DL models. It is simple, computationally efficient, and helps mitigate the vanishing gradient problem. ReLU does not saturate for positive values, enabling efficient learning. However, it can lead to the "dying ReLU" problem when outputs consistently remain zero for $x < 0$ [29].

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.1}$$

Sigmoid (Logistic Function)

The sigmoid function maps the input into a range between 0 and 1. It is especially suitable for binary classification problems and is typically used in the output layer. However, it suffers from the vanishing gradient problem in deep networks, which slows down learning [29].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Softmax

The softmax function transforms a vector of raw scores (logits) into a probability distribution, where the sum of all output values is equal to 1. It is used in the output layer of multi-class classification networks, allowing each output to be interpreted as the probability associated with a particular class [29].

$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad \text{for } i = 1, 2, ..., n \tag{2.3}$$

2.7.6 Loss Functions

The loss function is a mathematical function that quantifies the discrepancy between a model's predictions and the actual ground truth values. In supervised learning, it serves as a learning signal by providing a numerical measure of the model's performance on specific training examples [29].

In the context of DL, the loss function is a critical component during training. It is used to compute gradients that guide the optimization algorithm (such as gradient descent) in updating the model's weights to minimize prediction errors. A wide range of loss functions exist, each tailored to specific tasks, data types, and optimization goals [29]. At a high level, they are categorized into regression loss functions—such as Mean Squared Error—and classification loss functions, such as Cross-Entropy Loss [48]. This project focuses on classification loss functions, which assess prediction errors for discrete class labels.

Cross-Entropy Loss

A commonly used classification loss is the cross-entropy loss, which originates from the concept of entropy and measures the uncertainty in a system. It quantifies the difference between the predicted probability distribution and the true distribution. The loss is minimized when the predicted probabilities align closely with the actual labels. Cross-entropy is formally linked to the Kullback–Leibler divergence, which measures how one probability distribution diverges from another. Thus, minimizing cross-entropy reduces the difference between the predicted and actual class probability distributions [48].

Binary Cross-Entropy

Binary cross-entropy is used for binary classification problems. The model outputs values between 0 and 1, while the true labels are either 0 or 1. This loss penalizes both incorrect predictions and low-confidence predictions.

$$\mathcal{L}_{\text{binary}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.4}$$

This function strongly penalizes confident but wrong predictions due to the logarithmic scale. It encourages the model to make both accurate and confident predictions [48].

Categorical Cross-Entropy

Categorical cross-entropy is used in multi-class classification tasks. It measures the distance between the true class distribution (usually one-hot encoded) and the predicted class distribution (typically obtained via a softmax activation function in the output layer) [48].

$$\mathcal{L}_{\text{categorical}} = -\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij}) \tag{2.5}$$

Where:

- n: number of examples,

- C: number of classes,
- y_{ij}: true label (0 or 1),
- $\hat{y}_{ij}$: predicted probability for class j.

This function is especially suitable for tasks where each input belongs to one of multiple exclusive categories.

2.7.7 Overfitting and Regularization Techniques

In the field of ML, the primary objective is to build a model capable of generalizing to unseen data. However, a common issue encountered during model training is overfitting, which occurs when the model learns not only general patterns but also the noise and anomalies specific to the training set. This leads to low error on the training data but high error on validation or real-world data [49]. Common causes of overfitting include:

- A model that is too complex (e.g., too many parameters),
- A dataset that is too small,
- A lack of noise or diversity in the data.

Regularization refers to a set of techniques aimed at limiting model complexity to prevent overfitting. Its main goal is to improve the model's ability to generalize [29]. The following are widely adopted regularization techniques:

L1 Regularization

L1 regularization adds a penalty proportional to the sum of the absolute values of the model's weights. It promotes sparsity, meaning some weights may become exactly zero, effectively performing feature selection [29].

L2 Regularization

L2 regularization adds a penalty proportional to the square of the model's weights. It prevents the weights from growing too large, which stabilizes the learning process and enhances generalization [29].

Dropout

During training, dropout randomly deactivates a subset of neurons, as illustrated in Figure 2.7. This prevents the network from becoming overly reliant on specific neurons, encouraging redundancy and improving generalization [29].

Early Stopping

Early stopping involves monitoring the model's performance on a validation set during training and halting training when the performance begins to deteriorate. This strategy helps avoid overfitting by preserving the model state that best generalizes to unseen data [29].

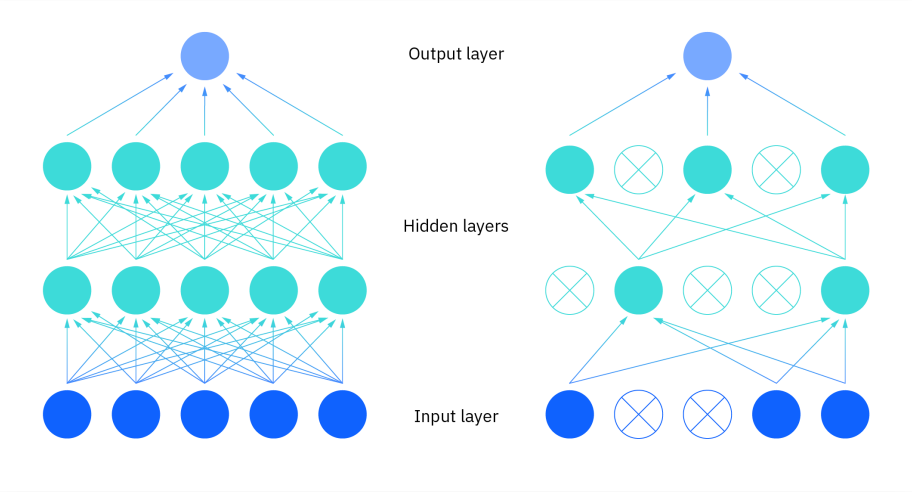


Figure 2.7: Illustration of Dropout technique [50].

2.7.8 Optimization Algorithms

In the context of ML and DL, optimization algorithms are essential for effectively training models. Their main objective is to minimize the loss function by adjusting the model parameters. There are many different optimizers, among which the most commonly used include:

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a classical method that updates the model parameters using only a single sample or a small batch (mini-batch) of data at each iteration. Unlike traditional batch gradient descent, which requires scanning the entire dataset before each update, SGD offers more frequent updates and helps accelerate learning, especially for large datasets.

However, gradients computed from a single example are noisy, which can make the optimization trajectory unstable. To address this, techniques such as momentum or dynamic learning rate adjustments are often applied alongside SGD [29].

Adaptive Moment Estimation

The Adam algorithm is a more advanced optimization method that combines the advantages of two techniques: momentum, which accelerates descent using the history of gradients, and RMSProp, which adapts the learning rate based on recent gradients. Adam computes moving averages of the first and second moments of the gradients for each parameter, allowing for a different learning rate per weight. Adam is widely used in deep learning models due to its ability to converge quickly and stably, even in situations with complex loss functions or noisy data. It requires little manual tuning and performs well by default, making it a preferred choice in most modern neural network architectures [29].

2.7.9 Performance Metrics

Assessing the performance of ML models, particularly in classification tasks, necessitates more than a simple evaluation of accuracy. Classification models produce discrete outputs, and as such, require metrics capable of capturing the correctness of these categorical predictions in a structured manner.

The confusion matrix is a fundamental evaluation tool that presents a tabular comparison between the predicted class labels and the actual ground-truth labels. It provides essential insight into the model's behavior by detailing true positives, false positives, true negatives, and false negatives (see Figure 2.8) [51].

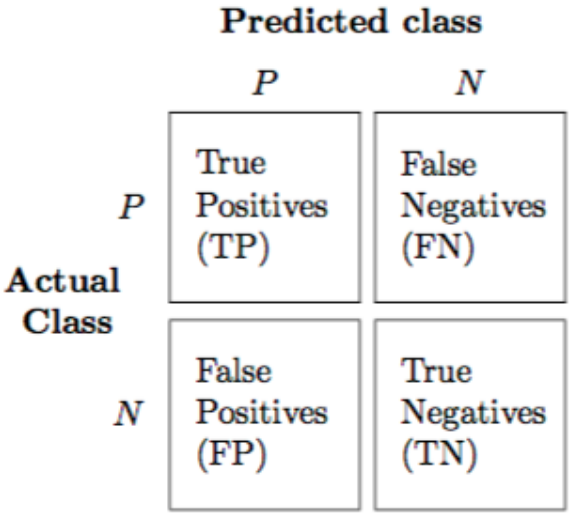


Figure 2.8: Confusion Matrix Example [52].

Accuracy

Accuracy measures the overall correctness of a model by calculating the proportion of true results (both true positives and true negatives) among the total number of cases examined [51]. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

Precision

Precision evaluates how many of the instances predicted as positive are actually positive. It is especially useful when false positives are costly [51]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.7}$$

A high precision indicates a low false positive rate, which is critical in avoiding unnecessary alerts and system disruptions.

Recall

Recall, also known as sensitivity or true positive rate, measures the ability of a model to detect actual positive cases. In the context of attack detection, it indicates how well the system identifies real attacks [51].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.8}$$

High recall is essential in ensuring that malicious activities are not overlooked. However, recall alone is insufficient if it comes at the expense of precision.

F1-Score

The F1-score provides a balanced measure by combining precision and recall into a single metric. It is the harmonic mean of precision and recall [51]:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

The F1-score is particularly useful when there is a trade-off between precision and recall and is widely used in evaluating models for DDoS detection, where both false positives and false negatives can be critical.

2.8 Related Work

This literature review was one of the most important and helpful parts during our thesis preparation, as it provided valuable insights into how recent research has approached DDoS detection and mitigation using ML-based methods. The goal of this review is to examine previous studies that focus on the application of machine learning techniques to detect and mitigate DDoS attacks in various types of networks, with particular emphasis on 5G networks. To ensure the relevance and timeliness of our analysis, we concentrated on research published from the year 2020 onward, prioritizing studies that closely align with the objectives and context of our thesis. In this section, we present a review of such papers, highlighting their key contributions and how they relate to our work.

Khan (2023). In his thesis, Khan [12] focused on the detection of DoS and DDoS attacks specifically targeting 5G network slices using deep learning methods. He proposed a simulation framework built with Free5GC and UERANSIM to replicate real-world 5G slicing scenarios. To address the lack of datasets tailored to this context, a new datasets were generated using both VM-based and container-based implementations. The study introduced two deep learning models CNN and BiLSTM to classify and detect these attacks. The BiLSTM model achieved the highest detection accuracy of 99.96%. This work highlights the importance of dataset generation and real-time detection in securing 5G network slicing environments.

Elzayn et al. (2023). In [53], the authors proposed a reinforcement learning-based approach for mitigating DDoS attacks in 5G-V2X networks. They designed a realistic environment using OpenAirInterface, FlexRIC, and robot cars equipped with 5G modules to simulate both benign and attack traffic. The architecture includes two network slices:

a benign slice for normal users and a sinkhole-type slice with limited resources to isolate suspected attackers. A deep learning agent detects the attack, while a reinforcement learning agent—trained using Q-learning, DQN, A2C, and PPO algorithms decides when to release the user from isolation. Among all models, Q-learning showed the most stable performance, achieving less than 3.2% average error in attack duration estimation using real testbed data. This work demonstrates the effectiveness of combining real-world 5G-V2X environments with intelligent RL-based mitigation strategies.

Sheikhi and Kostakos (2024) [54] proposed an unsupervised Federated Learning (FL) approach to detect DDoS attacks in 5G core networks. Their work addresses the limitations of centralized machine learning models, particularly privacy concerns and scalability issues. They built a realistic 5G testbed using Open5GS and UERANSIM, where SYN and UDP flood attacks were simulated on separate 5G Core instances. Network traffic was captured and preprocessed using Tshark, and an Autoencoder model was trained locally on each node. Instead of sharing raw data, each node sent its learned model parameters to a global server for aggregation. The final model achieved a detection accuracy of up to 99%, showing the viability of FL-based DDoS detection in 5G environments where data privacy is crucial.

Park et al. (2022) In this article [2], the authors address a critical gap in existing research by focusing on signaling DDoS attacks in the 5G Core Network (CN), rather than the Radio Access Network (RAN), which has traditionally received more attention. Their study introduces a machine learning-based intrusion detection system designed to detect attacks targeting CN functions such as AMF, SMF, and UPF. They propose a dual-method approach using Entropy-Based Analysis (EBA) and Statistical-Based Analysis (SBA) to preprocess traffic data and extract meaningful features. The dataset was generated with a Spirent Landslide 5G traffic generator, simulating 625,000 users. The system was evaluated using SVM, Naive Bayes, and Random Forest, with Random Forest achieving the highest accuracy of 98.4%. Their approach demonstrates high efficacy in differentiating between normal and attack traffic based on statistical and entropy features of session behavior, reinforcing the potential of ML in safeguarding CN infrastructure.

Saini et al. (2020) In [55], the authors introduce a machine learning-based detection framework for DDoS attacks. the study relys on an existing dataset that contain some of the latest types of attacks like SIDDoS and Smurf attacks. Features such as average packet size, bit rate, packet size, inter arrival time...etc are extracted and used to train the models via the Weka platform. The evaluated algorithms include Naive Bayes, Random Forest, MLP, and J48, with the latter achieving the highest detection accuracy of 98.64% . This study underscores the practical applicability of ML techniques in live traffic environments and the effectiveness of decision tree based methods for DDoS detection.

Reddy (2024) in [56] conducted a comprehensive study on DDoS detection in 5G networks using machine learning and deep learning techniques. The research developed and validated multiple models, including BERT, BiLSTM, Custom CNN, Random Forest, SVM, XGBoost, and an ensemble of Random Forest and XGBoost, within a simulated 5G environment created using free5GC and UERANSIM. The ensemble model achieved the highest accuracy (99.72%), demonstrating superior performance in detecting TCP

flood, UDP flood, and other DDoS attacks. However, the study also highlighted the trade-off between the ensemble model's improved accuracy and its increased computational complexity, suggesting that traditional ML models like Random Forest (99.59% accuracy) may be more practical for resource-constrained environments. The work addressed a critical gap by evaluating modern DL techniques alongside traditional ML and ensemble methods, providing a balanced analysis of their effectiveness in real-time DDoS detection for 5G networks. The simulation-based approach enabled rigorous testing under controlled attack scenarios while maintaining ethical standards through synthetic data usage.

Rana & Filippo (2024) in [57] propose a real-time DDoS detection framework for 5G networks that combines P4-programmable data planes with deep learning. Their system collects telemetry data through P4-based UPF and DPUs, processing it in real-time using Fastcapa and DPDK workers. A CNN model analyzes traffic patterns continuously, achieving 98.6% detection accuracy with sub-millisecond latency, triggering immediate mitigation upon attack detection.

Bomidika S(2024) in [31] explores the application of both ML and DL techniques for the detection of DDoS attacks within 5G networks. The findings indicate that ensemble-based models achieve superior accuracy compared to individual ML or DL approaches. The research is conducted in a simulated 5G environment utilizing Free5GC and UERANSIM, which enables the generation of realistic traffic patterns for model evaluation.

Abdoul & Ataro (2024) in [58] propose a hybrid deep learning approach for detecting and mitigating DDoS attacks targeting 5G core network Virtual Network Functions (VNFs). The method combines XGBoost for feature extraction with a deep neural network for classification. To preserve data privacy in distributed environments, Federated Learning (FL) is employed for model training. Evaluated using the CICDDoS2019 dataset, the framework demonstrated high accuracy.

Morteza et al. (2022) in [59] propose a distributed SDN-based architecture to detect and mitigate DDoS attacks in 5G networks. The system uses multiple dispersed controllers managed by a master controller, which dynamically balances the processing load among them. This hierarchical approach enhances scalability, reliability, and detection efficiency. Simulation results demonstrate the system's effectiveness in distributing workload and improving DDoS mitigation performance.

Bousalem et al. (2022) in [60] proposed a deep learning-based method to detect and mitigate DDoS attacks in 5G networks using dynamic network slicing. When an attack is detected, the system isolates the malicious users by moving them to a low-resource "sinkhole" slice. Their solution was tested on a real 5G prototype using OpenAirInterface and FlexRAN, with traffic generated from both real and virtual users. Among 100 trained models, the best one achieved around 97% accuracy and less than 4% false positive rate. Unlike previous works, this approach targets the RAN and is validated in a realistic environment, offering an effective and automated response to DDoS attacks.

Wen et al. (2022) in [16] introduced MobiFlow, a telemetry system tailored for enhancing security in 5G Open RAN networks. Drawing inspiration from NetFlow, MobiFlow captures detailed, real-time data from user devices, RAN components, and core network elements, and makes this information available to xApps on the near-real-time RIC. This rich telemetry stream enables a range of security applications, such as intrusion detection, rogue base station identification, and dynamic RAN control. What sets MobiFlow apart from existing solutions is its focus on fine-grained, low-level data—going beyond basic KPIs to support more responsive and intelligent security mechanisms. The authors argue that this level of detail is critical for building robust and scalable security services in the flexible, software-driven O-RAN environment.

Wen et al. (2024) in [61] introduced *5G-SPECTOR*, the first comprehensive framework designed to detect Layer-3 cellular protocol exploits within the O-RAN architecture. Unlike traditional security solutions that operate on coarse-grained metrics or limited vantage points, 5G-SPECTOR leverages a novel fine-grained telemetry stream called *MobiFlow*, which captures detailed RRC and NAS-level events from user equipment and base stations. It integrates with a programmable xApp, *MobiExpert*, that allows operators to define rule-based detection logic using the P-BEST language. The system was validated on a real O-RAN testbed using SD-RAN and OpenAirInterface, demonstrating its effectiveness in detecting both known and previously unseen attacks with low overhead. Furthermore, its modular design allows it to be extended with AI/ML-based analytics, such as MobiWatch, making it a flexible and forward-looking platform for cellular intrusion detection in programmable 5G networks.

Awad et al. (2024) in [62] presented a novel 5G prototype framework that integrates machine learning-driven xApps within the O-RAN architecture to detect and mitigate DDoS attacks in V2X scenarios. Their approach uses two key xApps: an Attack Detection (AD) xApp that employs deep learning to monitor live traffic and identify malicious flows in real time, and a Resource Control (RC) xApp that dynamically reallocates network resources to limit the impact of detected attackers. The system was implemented on a hybrid testbed combining physical and virtual UEs, using srsRAN and Open5GS, and demonstrated the effectiveness of xApps in ensuring reliable V2X communication even under DDoS attacks. By leveraging the Near-Real-Time RIC and modular O-RAN components, the framework ensures rapid detection and mitigation without disrupting service to legitimate users.

2.8.1 Comparative Analysis of Related Works

Table 2.1 presents a structured comparison of recent contributions to DDoS detection and mitigation in 5G networks, focusing on core attributes that are highly relevant to the design of our proposed framework. These include the type of datasets used, model selection and performance, simulation platforms, and the level of O-RAN integration.

This comparative review was instrumental in identifying the most suitable components for our architecture. In particular, it helped clarify which models consistently perform well in 5G settings, which datasets offer realistic traffic characteristics, and how xApps can be practically deployed for near-real-time defense. Based on these insights, we built a roadmap that balances effectiveness, modularity, and realism for 5G specific threat detection and response.

Table 2.1: Comparison of Related Works with Our Proposed Framework

Study	**Dataset Used**	**Models Evaluated**	**Best Performing Model**	**5G Simulation Tools**	**Mitigation Strategie**
Khan (2023) [12]	Custom + CICDDoS2019	CNN, BiLSTM	BiLSTM (99.96%)	Free5GC, UERANSIM	No
Elzayn et al. (2023) [53]	Real testbed traffic	DQN, A2C, PPO, CNN	Q-learning variant	OAI, FlexRIC, Robot Cars	Yes (O-RAN integration/xApp)
Park et al. (2022) [2]	Spirent traffic (625k users)	RF, SVM, Naive Bayes	RF (98.4%)	Simulated CN only	No
Awad et al. (2024) [62]	Hybrid real/virtual	DL + xApp AD	AD xApp DL model	srsRAN, Open5GS	Yes (2 xApps)
Reddy (2024) [56]	Simulated 5G traffic (Attack-/normal)	RF (99.59%), CNN (98.78%), BiLSTM (99.52%), Ensemble	Ensemble (RF+XGB: 99.72%)	Free5GC, UERANSIM	No
Rana and Filippo (2024) [57]	Real-time telemetry (P4-DPU)	CNN	CNN (98.6%)	P4 UPF, Fastcapa, DPDK	Yes (mitigation with programmable P4 switch UPF)

Key Observations and Roadmap Implications:

- **Dataset Strategy:** While many works rely on custom or synthetic data, CICDDoS2019 remains a standard benchmark due to its wide attack coverage. Our decision to enhance it with real 5G simulated traffic makes it more representative of live network behavior without sacrificing reproducibility.
- **Model Selection:** Most studies focus on deep learning (CNN, BiLSTM), but Random Forest consistently appears among top performers for its speed, interpretability, and robust performance. Our roadmap adopted all three (RF, CNN, BiLSTM)
- **Simulation Tools:** While some works adopt OpenAirInterface for its realism and integration capabilities, our selection of OAI and FlexRIC was primarily driven by their native compatibility with the O-RAN architecture. Specifically, their support for the E2 interface and xApp programmability made them ideal for implementing a near-real-time control and mitigation framework aligned with O-RAN standards.
- **Mitigation Support:** Active mitigation is rare in the reviewed literature. When it appears, it is often limited to high-level strategies like slice isolation. Our framework

directly targets and disconnects malicious UEs through the mitigation xApp, an approach not seen in previous works.

- **xApp Design and Scope:** While Awad and Elzayn leverage xApps, their focus is limited to specific domains such as V2X. In contrast, our system introduces a general purpose detection xApp and a dedicated mitigation xApp designed to act autonomously in real time.

Conclusion: This analysis confirms that while DDoS detection in 5G is widely studied, few frameworks implement actionable, real time mitigation. Even fewer are designed with O-RAN compliance in mind. Our proposed system fills this gap by delivering an interpretable, xApp-driven architecture that integrates both detection and mitigation within a fully simulated 5G/O-RAN environment. This makes it especially relevant for modern network deployments requiring fast, scalable, and standards compliant defense mechanisms.

Chapter 3

METHODOLOGY

3.1 Introduction

The related work survey served as a crucial foundation for our project. It helped us understand the strengths and limitations of existing approaches and guided our selection of tools, datasets, and machine learning models. By analyzing prior solutions, we were able to build a structured roadmap for implementing a practical and efficient DDoS detection and mitigation framework tailored specifically for a simulated 5G environment.

As illustrated in Figure 3.1, our methodology begins with the simulation of an O-RAN enabled 5G network using OpenAirInterface (OAI) and the FlexRIC near-real-time RIC. The simulated setup includes a containerized core network, a gNB, and user equipment (UE) configured through Linux namespaces. This allowed us to generate realistic 5G traffic, both normal and malicious.

The generated traffic was then used to enhance the CICDDoS2019 dataset, resulting in a new version referred to as **CICDDoS-5G**, which is more representative of realistic 5G network conditions. Prior to model training, the **CICDDoS-5G** dataset underwent a comprehensive preprocessing phase, including data cleaning, label encoding, and feature extraction. This step was essential to ensure data consistency and improve the quality of the training input for machine learning algorithms.

Based on the refined dataset, Based on our earlier analysis, we designed and trained three machine learning models, Random Forest, CNN, and BiLSTM for traffic classification. These models were carefully selected for their ability to handle complex patterns and temporal dependencies present in 5G traffic, ensuring robust detection of both benign and malicious flows.

Next, we developed a detection framework that operates on the UPF, where it captures real-time traffic and identifies anomalies. Upon detecting an attack, it immediately sends alerts to a mitigation framework implemented as an xApp on the FlexRIC controller. This xApp then responds by disconnecting the malicious UEs, ensuring timely and automated mitigation of DDoS threats.

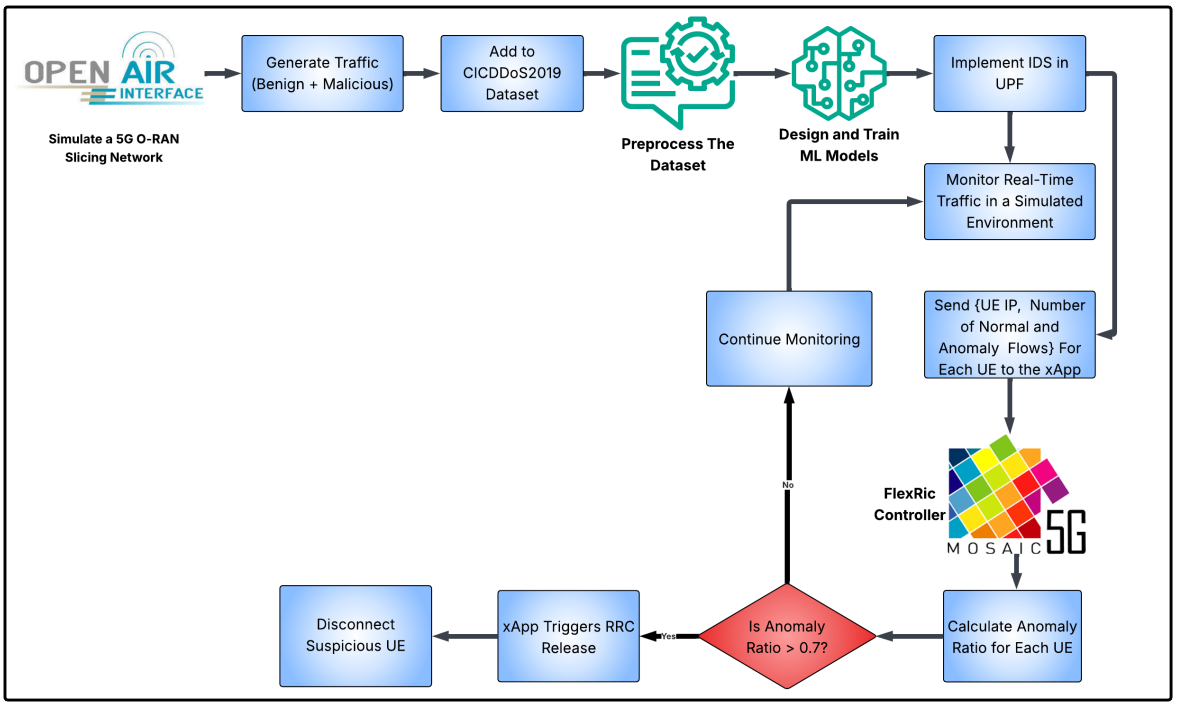


Figure 3.1: Proposed Method Strategy

3.2 5G Architecture Implementation

3.2.1 Environment Overview

The implementation was carried out on a native Linux environment running Ubuntu 22.04 due to its compatibility with the required software and its official support for 5G networks simulation tools. The entire 5G network was fully simulated on a physical machine without the use of virtual machines. The hardware and software specifications of the machine are summarized in the table below3.1.

Software Prerequisites

Before deploying the 5G simulation environment, several essential software components were installed following the official tutorial provided by the OpenAirInterface project on GitLab [63]. These tools were chosen to ensure compatibility with the simulation software and to provide a stable foundation for running containerized services.

The key software prerequisites included are:

- **putty:** `PuTTY` is a free and open-source terminal emulator widely used for remote access to servers, particularly through SSH (Secure Shell) and Telnet protocols. It provides a simple interface for connecting to remote machines.

- **Docker:** Docker is a platform that simplifies application development and deployment by using containers, a lightweight, isolated environments that package everything an application needs to run. In this 5G simulation, all core components such as AMF, SMF, UPF...etc are running inside containers. This made the setup easy to manage without relying on virtual machines or external infrastructure [64].

- **Docker Compose:** Docker Compose is a tool used to define and manage multi-container Docker applications. It allows users to configure all services and networks required by an application in a single YAML file. This simplifies the deployment and orchestration of complex systems, making it particularly useful for managing containerized 5G network components. In this setup, Docker Compose was used to efficiently coordinate and launch all the core services using one docker compose file [65].

These steps ensured that the environment was ready to run OpenAirInterface and FlexRIC services for the 5G network simulation.

Table 3.1: Implementation Environment Specifications

Category	**Specification**
Hardware	
Device	Dell Laptop
Processor	Intel Core i5 (8th Generation)
RAM	8 GB
Storage	SSD (256 GB)
Software	
Operating System	Ubuntu 22.04 LTS
5G simulated network	OpenAirInterface (OAI)
RIC Platform	FlexRIC (near-RT RIC)

3.2.2 OpenAirInterface Deployment

OpenAirInterface (OAI) [4] is an open source platform that provides a full stack implementation of the 5G network, supporting both software-based emulation and integration with radio hardware. In this project, OAI was used to simulate a complete 5G network. The 5G Core components were deployed in a containerized environment using Docker Compose, while the gNB was built natively on the host system. Multiple UEs were instantiated in isolated Linux network namespaces, enabling realistic simulation of the Radio Access Network (RAN), as illustrated in the simulated 5G architecture shown in Figure 3.2.

Core Network

The 5G Core Network in this setup is composed of the essential core components such as the AMF, SMF, UPF, NRF, and NSSF each running in isolated Docker containers to ensure modularity and scalability. Docker Compose was used to orchestrate and manage these services efficiently.

Network Slicing Implementation

To enable network slicing within the 5G testbed, the deployment was extended to include two independent slices. This was achieved by deploying two instances of SMF and two instances of UPF, each responsible for handling a specific data slice. These instances were built using the official Docker images provided by OpenAirInterface.

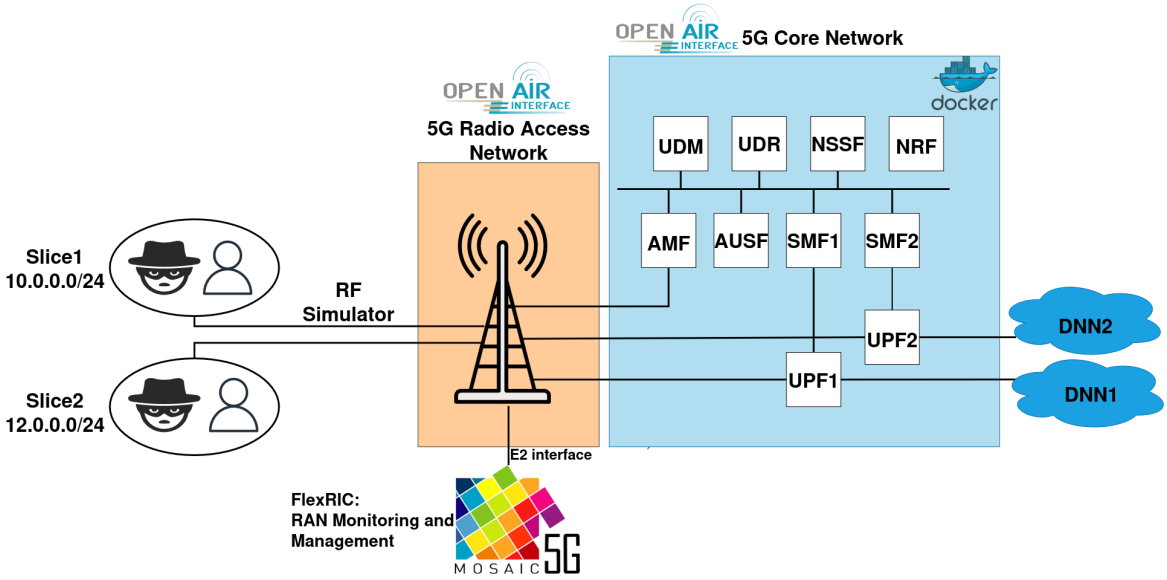


Figure 3.2: Simulated 5G Network

Both slices belong to the same Slice/Service Type (SST), specifically eMBB, but they are distinguished by different Slice Differentiators (SD), allowing the system to route traffic logically between slices. Each SMF-UPF pair was configured to serve a unique Data Network Name (DNN), representing different service types or applications.

To support this setup:

- The ‘docker compose‘ configuration was modified to launch the two SMF and two UPF containers.
- Each SMF was linked to its corresponding UPF and assigned a unique DNN.
- The amf config file was updated to include mappings between S-NSSAI (SST/SD) and DNN values.
- UE configuration files were adapted to request specific S-NSSAI values to connect to the appropriate slice.

Additionally, the Network Slice Selection Function (NSSF) was configured to handle the slice selection process, ensuring that each UE is directed to the correct slice based on the requested SD value. This setup enabled dynamic slice assignment, isolated traffic handling, and realistic simulation of differentiated services all running on shared physical infrastructure.

To run the core network services, the following Docker Compose command was used from the OAI CN5G directory:

```
1 docker compose up -d
```

This brings up all defined services (AMF, SMF, UPF, NRF, AUSF, etc.) in detached mode as shown in figure 3.3.

```
ensta@rania:~/5G-testbed/oai-cn5g$ docker compose up -d
[+] Running 12/12
 ✔ Container oai-ext-dn      Running                                    0.0s
 ✔ Container oai-nssf        Running                                    0.0s
 ✔ Container oai-nrf         Running                                    0.0s
 ✔ Container mysql           Running                                    0.0s
 ✔ Container oai-udr         Running                                    0.0s
 ✔ Container oai-udm         Running                                    0.0s
 ✔ Container oai-ausf        Running                                    0.0s
 ✔ Container oai-amf         Running                                    0.0s
 ✔ Container oai-smf-slice1  Running                                    0.0s
 ✔ Container oai-upf-slice1  Running                                    0.0s
 ✔ Container oai-smf-slice2  Running                                    0.0s
 ✔ Container oai-upf-slice2  Running                                    0.0s
```

Figure 3.3: 5GCN running components

Next Generation NodeB (gNB)

The gNB represents the 5G base station responsible for handling radio communication between the UE and the 5G core. In this simulated environment, the gNB was built directly on bare-metal by following the official OpenAirInterface (OAI) tutorial and documentation [4]. This approach provided full access to low-level configurations and enabled the integration of an E2 agent for O-RAN programmability.

To ensure compatibility with O-RAN xApps and the FlexRIC platform, we focused on compiling the gNB with E2 support. The steps followed for building both gnb and user equipments are listed below:

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g
cd openairinterface5g
git checkout slicing-spring-of-code
cd cmake_targets
./build_oai -c -C -w SIMU --gNB --nrUE --build-e2 --ninja
cd ../
git checkout develop
cd cmake_targets
./build_oai --build-lib telnetsrv
```

The initial checkout to the ‘slicing-spring-of-code‘ branch enables slicing and E2 agent support. The flag `--build-e2` is essential, as it compiles the gNB with the E2 interface, a cornerstone of O-RAN’s openness and programmability. This E2 agent allows external controllers (like the Near-RT RIC) to access and modify RAN behavior at runtime. Additionally, the flag `--nrUE` ensures that the NR User Equipment (UE) is also compiled, allowing full-stack simulations involving both the gNB and UE in a standalone setup.

After building the RAN components, including the E2 agent and the Telnet server library, the gNB was launched from the `build` directory using the following command:

```
sudo ./nr-softmodem \
  -O ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb.sa.band78.fr1.106PRB.
     ↪ usrpb210.conf \
  --sa \ --rfsim \ -E \
  --gNBs.[0].min_rxtxtime 6 \
  --telnetsrv \
  --telnetsrv.shrmod rrc
```

This command runs the gNB in **standalone (SA) mode** using the `--sa` flag, meaning it connects directly to the 5G Core Network (5GC) without relying on a 4G EPC fallback.

The `--rfsim` option enables the **RF simulator**, allowing simulation of radio frequency behavior entirely in software eliminating the need for physical RF hardware.

The `-E` flag activates the compiled E2 agent, enabling communication with the FlexRIC Near-RT RIC and allowing programmable control of RAN functions such as PRB allocation and RRC management.

The executable `nr-softmodem` is a central software component that implements the 5G NR protocol stack (PHY and MAC layers) in real-time. The configuration file specified by the `-O` flag contains essential radio settings such as the frequency band (Band 78), numerology, subcarrier spacing, and number of physical resource blocks (106 PRBs in this case), which emulate a realistic mid-band 5G deployment.

The `--telnetsrv` flag starts the Telnet server, which enables **runtime monitoring and debugging**. The additional option `--telnetsrv.shrmod rrc` specifically enables the observation and adjustment of the Radio Resource Control (RRC) layer during operation.

After initialization, the gNB sends an **NGAP (Next Generation Application Protocol)** connection request to the AMF. This request initiates the signaling interface between the RAN and the Core. Upon receiving a successful response from the AMF, the gNB transitions to a ready state and starts listening for User Equipments (UEs) attempting to attach to the network.

This NGAP-based communication establishes the NG interface, which is essential for handling signaling and mobility management procedures between the gNB and the core. Once the NG interface is active, the gNB can handle RRC (Radio Resource Control) connection requests from UEs and forward relevant messages to the AMF for registration, authentication, and session establishment.

User Equipments

To emulate User Equipments in our 5G environment, we used the namespace-based approach provided by OpenAirInterface (OAI). OAI offers two primary methods for running multiple UEs simultaneously: one based on Docker containers and the other using Linux network namespaces. For this setup, we opted for the namespace method due to its simplicity, lower resource overhead, and seamless integration with the host system. Compared to Docker, namespaces eliminate the need for container orchestration or image management, making development, debugging, and monitoring much easier.

The process begins by creating a dedicated network namespace for the UE using the `multi-ue.sh` script provided by OAI. This script not only sets up the isolated namespace but also creates a virtual Ethernet pair, one interface remains on the host while the other is moved into the UE's namespace. An IP address is then assigned to each end of the pair, establishing a direct communication link. This link is later used by the UE to connect to the RF simulator (rfsim) server, enabling simulated radio access.

the following command creates one UE namespace (`-c1`) and opens a shell inside it

(-e):

```
sudo /home/ensta/5G-testbed/openairinterface5g/tools/scripts/multi-ue.sh -c1 -
    ↪ e
```

Once the namespace was set up, the UE was launched inside it using the following command:

```
sudo LD_LIBRARY_PATH=. ./nr-uesoftmodem \
--rfsimulator.serveraddr 10.201.1.100 \
-r 106 \
--numerology 1 --band 78 -C 3619200000 \
--rfsim \
-O /home/ensta/5G-testbed/openairinterface5g/targets/PROJECTS/GENERIC-NR-5GC/
    ↪ CONF/ue.conf -E
```

This command starts the UE in RF simulator mode (‘–rfsim‘) and connects it to the simulated gNB via the specified server address. The configuration file defines the UE’s parameters, including band, carrier frequency, and numerology. Once launched, the UE initiates the registration procedure with the gNB and subsequently with the Core Network, completing the full attach process and enabling end-to-end data communication within the 5G architecture.

As shown in Figures 3.4a and 3.4b, each UE successfully initiates a PDU session establishment procedure with the 5G Core Network. The NAS signaling indicates that the UEs send a ‘Pdu Session Establishment Request‘ to the AMF, which then forwards the request to the appropriate SMF based on the requested S-NSSAI. After validation, the core responds with a ‘PDU Session Establishment Accept‘ message, confirming successful session creation. Each UE is assigned a distinct IPv4 address from its respective slice subnet.

3.2.3 FlexRIC Controller Integration

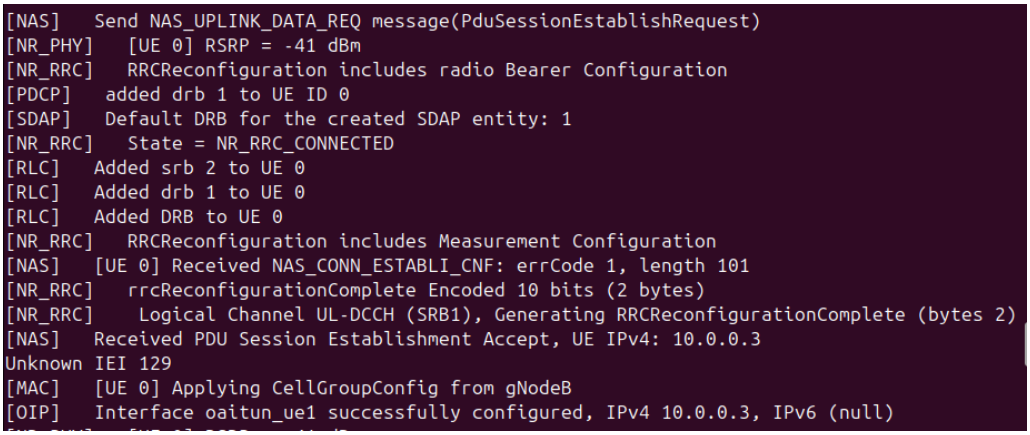
To enable intelligent and programmable control over the RAN, the simulated 5G network was extended with the integration of the FlexRIC controller. FlexRIC is an open and modular near-real-time RAN control platform developed to support the O-RAN architecture. It provides the necessary components for managing and interacting with RAN elements through the standardized E2 interface, allowing the implementation of intelligent applications (xApps) that can observe and control RAN behavior dynamically.

The installation and configuration of FlexRIC were carried out by following the official instructions available on the project’s GitLab repository [5]. Once integrated, FlexRIC established E2 connectivity with the gNB.

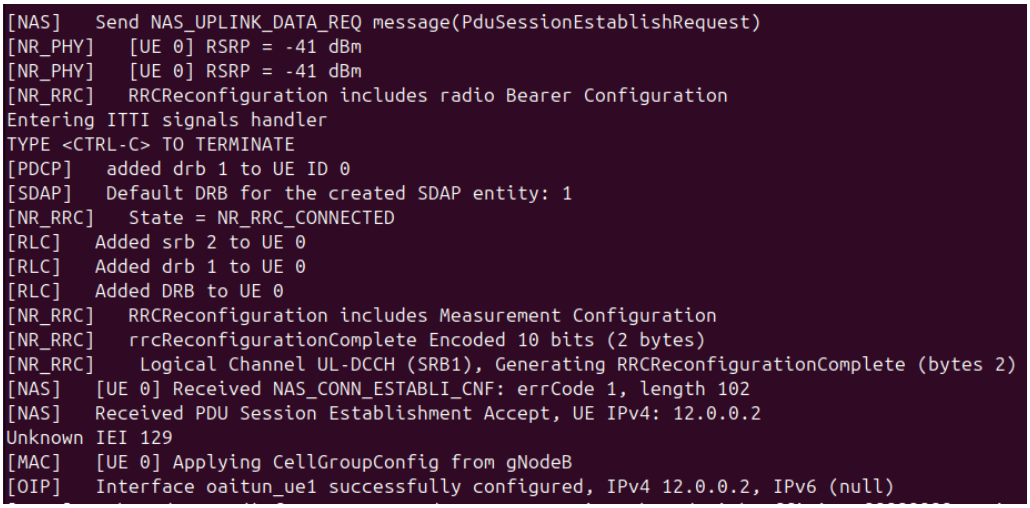
The integration architecture consists of the following key components:

- **E2 Agent:** This runs on the OAI gNB and acts as the interface between the gNB and the controller. It exposes RAN data and accepts control commands from the FlexRIC controller.

- **E2 Termination (E2T):** A central component within the FlexRIC platform that handles communication with multiple E2 Agents. It manages the setup and maintenance of E2 connections.



(a) UE connected via Slice 1 (SST=1, SD=0X000001)



(b) UE connected via Slice 2 (SST=1, SD=0X000005)

Figure 3.4: UEs connected to distinct slices with active PDU sessions

- **xApps:** Modular control or monitoring applications that plug into FlexRIC. These can implement functions such as traffic steering, slice orchestration, anomaly detection, or handover optimization.

FlexRIC supports a range of **E2 Service Models**, which define the type of information exchanged between the controller and the RAN node. These service models include:

- **KPM (Key Performance Measurement)**: Provides statistics and performance metrics from the gNB, such as throughput and latency.
- **RC (RAN Control)**: Enables control of specific RAN behaviors, including handovers and scheduling decisions.
- **RAN Slice Information**: Allows slicing-related data to be reported and configured, facilitating slice-aware control.

The integration process involved compiling FlexRIC from source and configuring it to recognize and communicate with the E2 Agent embedded within the OAI gNB. Once the E2 connection was established, the gNB began transmitting real time RAN metrics (such

as radio link quality, UE state, or load) to the controller. In return, FlexRIC through its xApps was able to send control commands back to the gNB to adjust behaviors such as scheduling, slicing policies, or resource allocation.

```
[E2 AGENT]: E2 SETUP REQUEST timeout. Resending again (tx)
[E2-AGENT]: E2 SETUP RESPONSE rx
[E2-AGENT]: Transaction ID E2 SETUP-REQUEST 4 E2 SETUP-RESPONSE 4
```

Figure 3.5: gNB E2 Agent log: successful E2 connection

```
ensta@rania:~$ ./start-ric.sh
[UTIL]: Setting the config -c file to /usr/local/etc/flexric/flexric.conf
[UTIL]: Setting path -p for the shared libraries to /usr/local/lib/flexric/
[NEAR-RIC]: nearRT-RIC IP Address = 127.0.0.1, PORT = 36421
[NEAR-RIC]: Initializing
[NEAR-RIC]: Loading SM ID = 143 with def = RLC_STATS_V0
[NEAR-RIC]: Loading SM ID = 142 with def = MAC_STATS_V0
[NEAR-RIC]: Loading SM ID = 144 with def = PDCP_STATS_V0
[NEAR-RIC]: Loading SM ID = 148 with def = GTP_STATS_V0
[NEAR-RIC]: Loading SM ID = 2 with def = ORAN-E2SM-KPM
[NEAR-RIC]: Loading SM ID = 146 with def = TC_STATS_V0
[NEAR-RIC]: Loading SM ID = 3 with def = ORAN-E2SM-RC
[NEAR-RIC]: Loading SM ID = 145 with def = SLICE_STATS_V0
[iApp]: Initializing ...
[iApp]: nearRT-RIC IP Address = 127.0.0.1, PORT = 36422
[NEAR-RIC]: Initializing Task Manager with 2 threads
[E2AP]: E2 SETUP-REQUEST rx from PLMN   1. 1 Node ID 3584 RAN type ngran_gNB
[NEAR-RIC]: Accepting RAN function ID 2 with def = ORAN-E2SM-KPM
[NEAR-RIC]: Accepting RAN function ID 3 with def = ORAN-E2SM-RC
[NEAR-RIC]: Accepting RAN function ID 142 with def = MAC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 143 with def = RLC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 144 with def = PDCP_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 145 with def = SLICE_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 146 with def = TC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 148 with def = GTP_STATS_V0
[iApp]: E42 SETUP-REQUEST rx
[iApp]: E42 SETUP-RESPONSE tx
```

Figure 3.6: FlexRIC controller log showing registration of multiple Service Models and E2 connection acceptance

As shown in Figures 3.5 and 3.6, the gNB E2 agent initially reports a timeout while attempting to send the E2 SETUP REQUEST. This occurred because the FlexRIC controller had not yet been launched. Once the controller was started, the gNB successfully received the corresponding E2 SETUP RESPONSE, establishing the E2 connection.

On the controller side, the FlexRIC log confirms the successful loading and registration of multiple RAN function IDs, corresponding to service models such as KPM, RC, MAC, RLC, and SLICE. These enable both monitoring and control functionalities over the RAN.

Furthermore, the last two lines of the log

```
[iApp]: E42 SETUP-REQUEST rx
[iApp]: E42 SETUP-RESPONSE tx
```

appear after launching the xApp, indicating that it has connected to the FlexRIC controller and established its own E2 Application Protocol (E42) session. This confirms that the xApp is now actively communicating with the RIC and is ready to subscribe to service models, receive data, or send control commands to the gNB.

This end to end integration —gNB with E2 Agent, FlexRIC as the Near-RT RIC, and the xApp— enables full programmability and dynamic control of the RAN, in line with the O-RAN architecture's goals of openness, flexibility, and disaggregation

3.3 Simulating Attack Scenarios

To evaluate the effectiveness of our proposed machine learning-based defense framework, we conducted a series of controlled DDoS attack simulations within a containerized 5G network environment built using OpenAirInterface (OAI) and the FlexRIC controller. This setup allowed us to emulate realistic network behavior while maintaining the flexibility to control and monitor attack parameters.

We designed two types of attack scenarios to analyze the impact of volumetric DDoS attacks on network performance and slice isolation:

- **Intra-slice attack scenario**: Both User Equipments (UEs) belong to the same network slice. One UE generates legitimate traffic while the second launches an attack. This helps us study how malicious traffic affects other users within the same slice.

- **Inter-slice attack scenario**: The attacking and legitimate UEs are assigned to different slices. This setup is used to assess how well network slicing isolates services and whether DDoS traffic in one slice spills over to affect another.

The attacks were executed from the UEs toward a key core network component like the User Plane Function (UPF) using the `hping3` traffic generation tool. Below, we describe the two specific attack types simulated in the simulated 5G network.

3.3.1 SYN Flood Attack

To evaluate the resilience of our defense framework, we simulated a TCP SYN flood attack using the well known packet crafting tool `hping3`. This type of attack is designed to exhaust server resources by rapidly sending a large number of TCP connection requests, ultimately aiming to overwhelm the target and degrade service quality.

The attack was launched using the following command:

```
hping3 -S -d 100000000000000000 -I oaitun_ue1 --flood 192.168.70.134
```

This command sends a continuous stream of TCP SYN packets (`-S` flag), with an abnormally large payload size specified by the `-d` flag, over the interface `oaitun_ue1` the tunnel interface used by the UE. The `--flood` option ensures that packets are sent as fast as possible, creating an intense flood toward the target IP address (in this case, the UPF at `192.168.70.134`).

3.3.2 UDP Flood Attack

In addition to TCP based attacks, we also simulated a UDP flood attack to evaluate how our defense framework handles different types of volumetric DDoS threats. Unlike TCP SYN floods, which aim to exhaust server-side connection tables, UDP flood attacks overwhelm the target with a high volume of stateless datagrams. These attacks consume bandwidth and processing resources, often causing disruption to legitimate services due to their intensity and randomness.

The attack was launched from one of the UEs in the simulated 5G environment using the `hping3` tool. The UE, assigned to a dedicated slice, sent a continuous stream of UDP packets at high speed toward a component of the 5G core (e.g., the UPF or DNS server). The following command was used to initiate the flood:

```
hping3 --udp -d 1200 -I oaitun_ue1 --flood 192.168.70.134 -p 80
```

This command generates UDP packets (`--udp`) with a payload size of 1200 bytes (`-d 1200`), transmitted through the `oaitun_ue1` interface. The packets are sent to the target IP address `192.168.70.134` on port 80 (`-p 80`), using the `--flood` option to maximize the packet rate.

This scenario helps us analyze the impact of high volume, connectionless traffic on network behavior, especially in the context of 5G slice isolation. It also tests the responsiveness of our defense framework in detecting and mitigating UDP based threats, which are typically more difficult to track due to their stateless nature.

3.4 Dataset Preprocessing

To build a reliable and high-performing DDoS detection model, the quality, diversity, and volume of the training data are just as essential as the choice of ML algorithm. In this study, we selected the CICDDoS2019 dataset, which provides a rich and varied set of network traffic data including both benign and multiple DDoS attack scenarios. However, raw data from such datasets often contains noise, redundancy, and inconsistencies that can negatively impact model performance. Therefore, we have applied a thorough preprocessing pipeline to refine and prepare the data before feeding it into our ML models. The steps of this preprocessing pipeline are detailed below.

3.4.1 Description of Dataset

The CICDDoS2019 dataset, developed by the Canadian Institute for Cybersecurity, consists of 18 CSV files containing flow-based analysis results generated by CICFlowMeter-V3. It offers over 80 features, including timestamps, IP addresses, ports, and protocols. The dataset contains more than 12.7 million labeled traffic records collected over a two-day period in a controlled testbed environment. It includes 19 types of recent, real-world-like DDoS attacks, categorized into reflection-based and exploitation-based types, as illustrated in Figure 3.7.

3.4.2 Dataset Enhancement for 5G DDoS Detection

We initially trained our detection model using the CICDDoS2019 dataset, which is a rich and diverse resource for studying various DDoS attack types. However, this dataset was

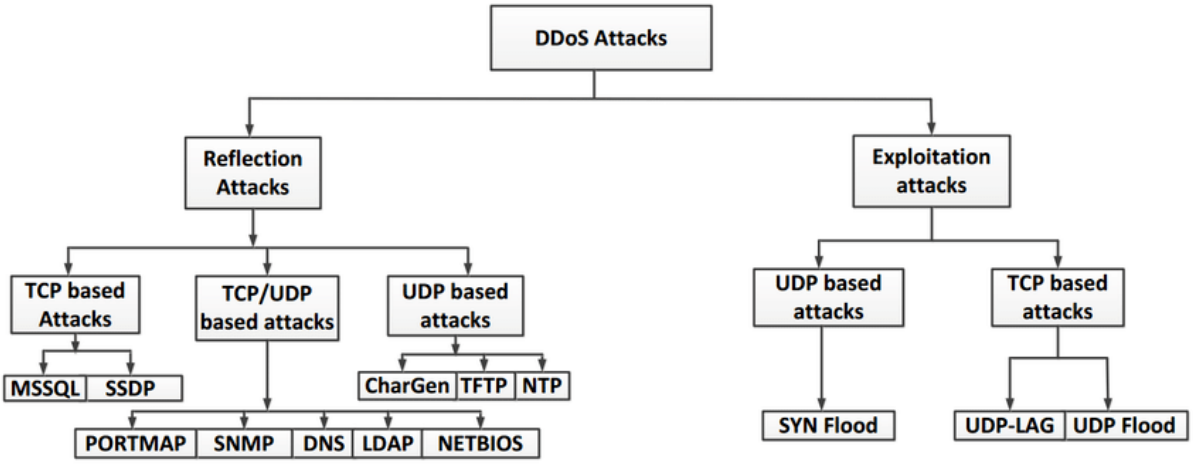


Figure 3.7: Types of DDoS attacks in CICDDoS2019 [66].

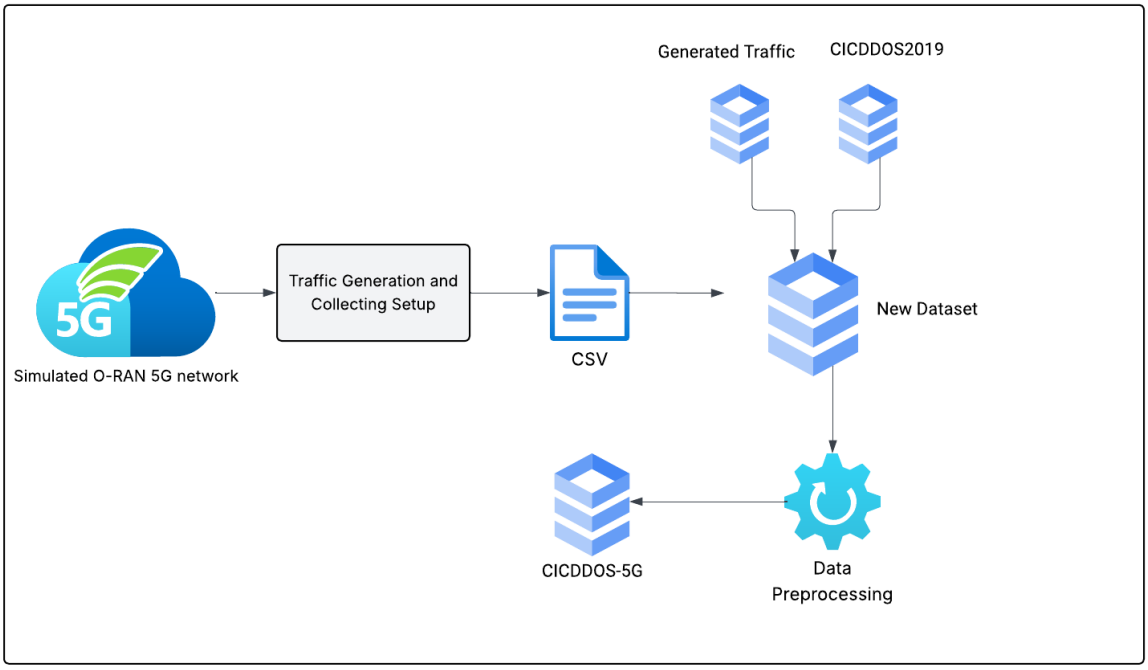


Figure 3.8: Dataset Enhancement for 5G DDoS Detection

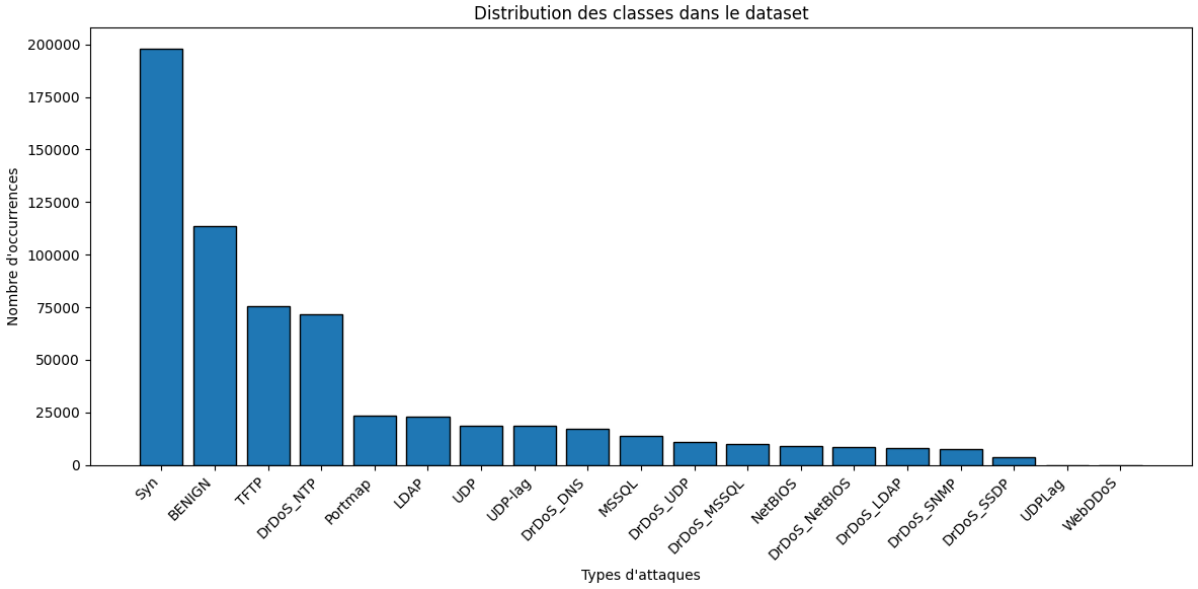


Figure 3.9: Class distribution in the CICDDoS2019 dataset

not designed specifically for 5G network environments and does not reflect the unique characteristics of O-RAN 5G network slices. As a result, when deploying our model in a simulated 5G network environment using the OAI simulator, the model struggled to effectively detect DDoS attacks, revealing a key limitation in the dataset's applicability to modern 5G systems.

To overcome this limitation, we implemented a 5G network slicing scenario using OIA and generated new traffic data tailored to O-RAN 5G environments. For benign traffic, ICMP ping was used, while various DDoS attacks such as UDP flood and SYN flood were simulated using the hping3 tool. The traffic was captured using tcpdump in PCAP format and subsequently converted into CSV format using CICFlowMeter, extracting 84 relevant flow-based features. These new data samples, comprising both malicious and benign traffic specific to 5G network slices, were integrated with the CICDDoS2019 dataset.

The result is a new enriched dataset, named **CICDDoS-5G**, which more accurately reflects the dynamic behavior and architectural specificities of modern 5G networks. The distribution of traffic in our final dataset is presented in Table 3.2. The overall process of dataset enhancement is illustrated in Figure 3.8.

Table 3.2: Total Traffic Before Preprocessing

Property	**Value**
CICDDoS2019 traffic	632.474
Real-time traffic	100.883

3.4.3 Preprocessing Pipeline

The dataset employed in this study exhibits a pronounced class imbalance, with benign (normal) flows significantly underrepresented compared to attack instances. To ensure robust binary classification (Attack vs. Normal), this imbalance was addressed using

a controlled downsampling strategy. Instead of enforcing strict class parity, the number of DDoS samples was reduced according to a fixed ratio relative to benign flows. This approach preserved a representative distribution of both classes while minimizing information loss. The initial class distribution is illustrated in Figure 3.9.

Dataset Cleaning

We began by cleaning the dataset to improve its quality prior to training our model. Unnecessary columns such as Timestamp, IP addresses, and Flow ID were removed, as they do not contribute to the detection of attacks.

All infinite values were replaced with NaN. Columns or rows containing an excessive number of missing values were dropped, and the remaining missing values were imputed using the mean of each respective column. Additionally, we removed columns with more than 90% zero values and excluded rows containing negative values in features such as Flow Duration, which should always be positive.

```
# Data Cleaning and Feature Removal
to_drop = ['Timestamp', 'Source IP', 'Destination IP', 'Flow ID']
df.drop(columns=to_drop, inplace=True)
df.replace([np.inf, -np.inf], np.nan, inplace=True)

# Remove rows with negative values in key columns
positive_cols = ['Flow Duration', 'Total Length of Fwd Packets']
exists = [c for c in positive_cols if c in df.columns]
if exists:
    df = df[(df[exists] >= 0).all(axis=1)]
```

Listing 3.1: Data Cleaning and Feature Removal

Next, we converted the categorical columns into numbers using one-hot encoding. This step was necessary because most ML and DL algorithms can only work with numerical inputs.

```
df = pd.get_dummies(df, columns=cat_cols)
```

Listing 3.2: One-hot Encoding

We then used the `StandardScaler` from Scikit-learn to scale all the feature values. This standardization step is important because it puts all features on the same scale, which helps the DL model treat each feature fairly.

`StandardScaler` transforms each feature so that it has a mean of 0 and a standard deviation of 1. This is done by subtracting the mean and dividing by the standard deviation for each value. Features with large ranges (like byte counts or durations) can otherwise dominate the learning process, so standardizing ensures better model performance and faster convergence during training.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Listing 3.3: Feature Standardization

Finally, we converted the target column Label into binary values. We replaced the BENIGN class with `0`, and grouped all attack types (such as SYN, TFTP, DrDoS_NTP, etc.) under the value `1`.

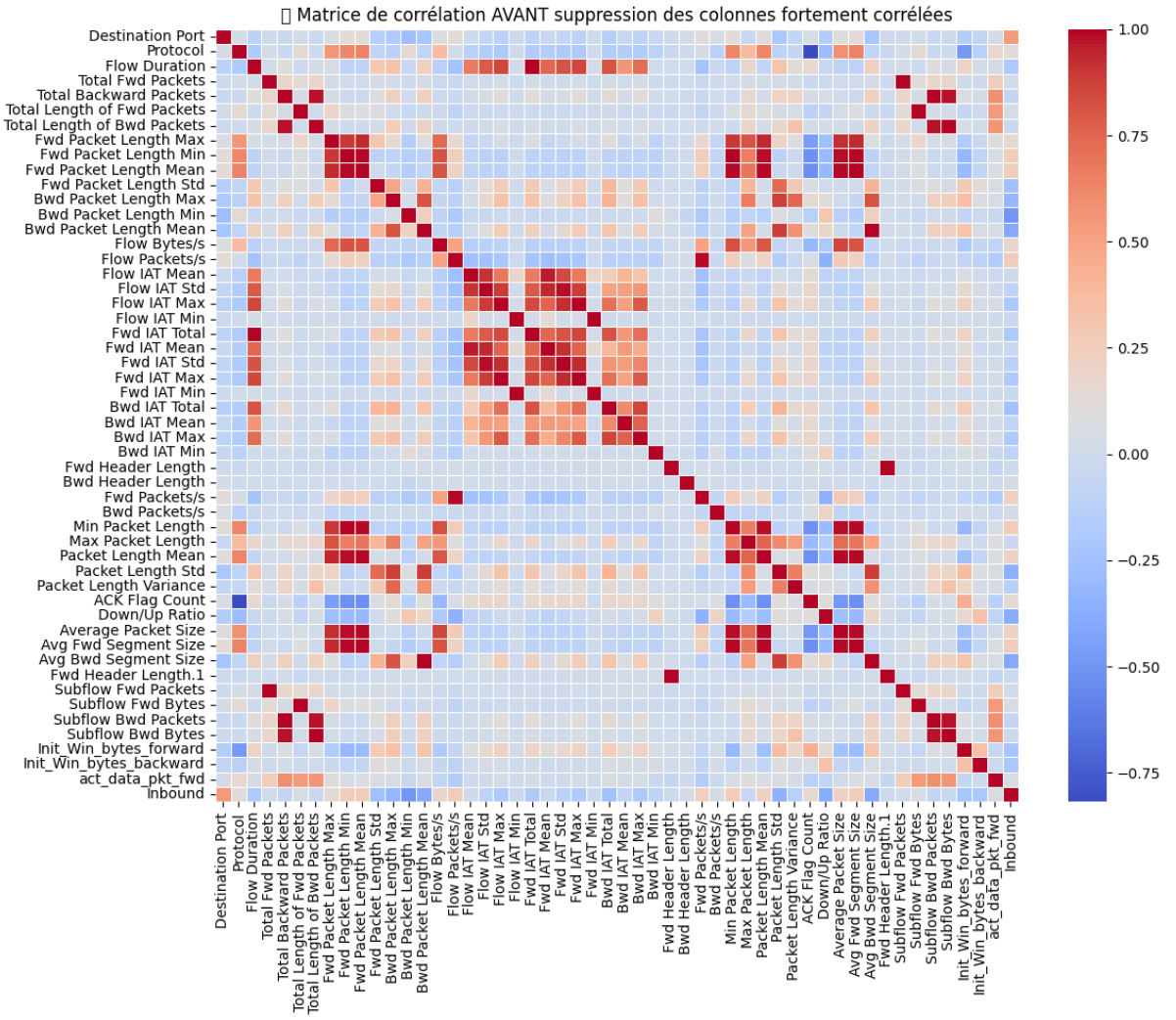


Figure 3.10: Correlation matrix between numerical features

```
df['Label'] = df['Label'].apply(lambda x: 0 if x.upper() == 'BENIGN' else 1)
```

Listing 3.4: Label Encoding

Feature Selection

Feature selection is a crucial step in building efficient and reliable ML models. By reducing the number of features, we can minimize noise in the dataset, reduce the risk of overfitting, and improve model performance. In our approach, we first calculated the correlation matrix between all numerical features. When two features had a correlation coefficient greater than 0.95, we removed one of them to avoid redundancy. The correlation matrix is shown in Figure 3.10.

We then applied two feature selection techniques: the Chi-Square (Chi^2) test and the Information Gain Ratio (IGR). The Chi^2 test evaluates the statistical dependence between each feature and the target variable, helping to identify the most relevant features for classification tasks. In contrast, IGR is based on information theory and measures how much information a feature contributes to the prediction of the target, taking into account the feature's entropy.

Table 3.3: Top 10 Selected Features for CNN and BiLSTM Models

Feature Name	Simple Description
Bwd IAT Mean	Average time between packets going backward
Bwd IAT Min	Smallest time between two backward packets
Flow IAT Mean	Average time between packets in a flow
Flow IAT Std	Variation in the time between packets in a flow
Fwd Header Length	Total header size of forward packets
Total Length of Fwd Packets	Total size of all forward packets
ACK Flag Count	Number of ACK flags in the flow
Protocol	Type of protocol used (e.g., TCP, UDP)
Flow Packets/s	Number of packets sent per second
Flow Bytes/s	Number of bytes sent per second

We applied the Chi-Square and Information Gain Ratio (IGR) methods using a custom Python function (see Listing 3.5) that selects the top 35 features from each method, intersects the two sets, and retains the 10 most relevant features.

```
def select_features(X, y):
    X_chi = X.copy()
    X_chi[X_chi < 0] = 0
    chi2_selector = SelectKBest(score_func=chi2, k=35)
    chi2_selector.fit(X_chi, y)
    chi2_features = X.columns[chi2_selector.get_support()].tolist()

    def information_gain_ratio(X_discretized, y):
        info_gain = mutual_info_classif(X_discretized, y, discrete_features=
            ↪ True)
        entropy_feature = [
            -np.sum((counts / np.sum(counts)) * np.log2(counts / np.sum(counts))
                ↪ )
            for _, counts in (np.unique(X_discretized[:, i], return_counts=True)
                ↪  for i in range(X_discretized.shape[1]))
        ]
        entropy_feature = np.where(np.array(entropy_feature) == 0, 1e-6,
            ↪ entropy_feature)
        return info_gain / entropy_feature

    kbin = KBinsDiscretizer(n_bins=10, encode='ordinal', strategy='uniform')
    X_binned = kbin.fit_transform(X)
    igr_scores = information_gain_ratio(X_binned, y)
    igr_features = X.columns[np.argsort(igr_scores)[-35:]].tolist()

    final_features = [f for f in igr_features if f in chi2_features]
    top10 = final_features[:10]

    return X[top10], top10
```

Listing 3.5: Feature Selection Function Combining Chi-Square and IGR

For the feature selection process to train random forest model , we applied a feature

Table 3.4: Top 15 Selected Features for Random Forest Model

Feature	**Description**
Destination Port	The destination port number used in the flow.
Inbound	Whether the packet is incoming to the internal network.
Min Packet Length	Minimum length of the packets in the flow.
URG Flag Count	Number of packets with the TCP URG flag set.
ACK Flag Count	Number of packets with the TCP ACK flag set.
Fwd Packet Length Min	Minimum packet length in the forward direction.
Flow Bytes/s	Number of bytes transmitted per second in the flow.
Init_Win_bytes_forward	Initial TCP window size in the forward direction.
Bwd Packets/s	Number of backward packets per second.
Packet Length Std	Standard deviation of the packet lengths.
Fwd Packet Length Mean	Average packet length in the forward direction.
CWE Flag Count	Count of packets with CWE (ECN-Echo) flag.
act_data_pkt_fwd	Number of forward packets carrying actual data.
Flow IAT Mean	Mean inter-arrival time between packets.
Total Backward Packets	Total number of packets in the backward direction.

selection process based on a Random Forest classifier. We trained the model on the full training dataset and leveraged its built-in feature importance scores, accessible via the `feature_importances_` attribute. The features were ranked by importance in descending order, and the top 15 most influential features were selected for the final training.

This selection method helped reduce overfitting, improved training efficiency, and maintained high classification accuracy. Table 3.4 lists the chosen features along with a brief description of each.

Dataset Balancing and Shuffling

To address the imbalance between benign and attack traffic in our dataset, we applied the **SMOTE** (Synthetic Minority Over-sampling Technique). This method synthetically generates new instances of the minority class, helping balance the number of samples in both categories.

After oversampling, we shuffled the dataset rows using the following Python instruction:

```
1 df = shuffle(df,random_state=42).reset_index(drop=True)
```

Listing 3.6: Shuffling the dataset

Shuffling ensures that the model does not learn any ordering or pattern from the data sequence, which could lead to biased training or overfitting. This randomization step is essential for improving the generalization ability of the model.The final distribution of the dataset is shown in Table 3.5.

3.5 Model Design and Training

Model training is a crucial phase in the machine learning pipeline, where various ML and DL algorithms are applied to the preprocessed dataset, as illustrated in Figure 3.11.

Table 3.5: Final Dataset Structure After Preprocessing

Property	**Value**
Total samples	1,032,105
Number of features	10
BENIGN instances	513,459
ATTACK instances	518,646

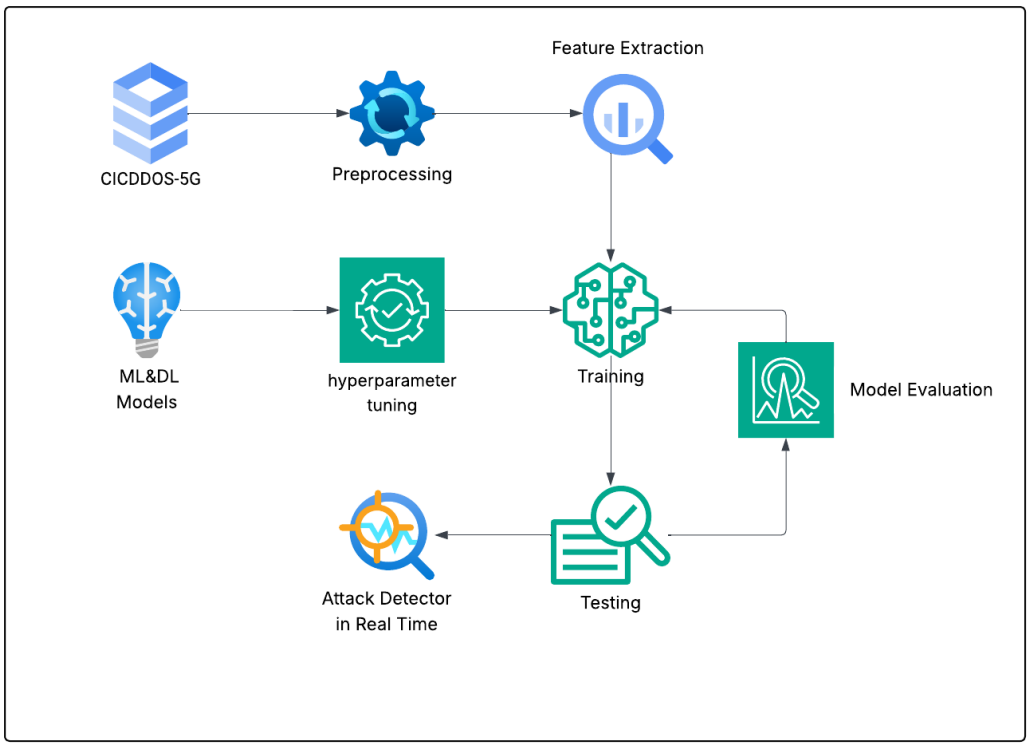


Figure 3.11: Training Pipeline

This study leverages a diverse set of models—Random Forest, CNN, BiLSTM, and an ensemble approach—each selected for its specific strengths in detecting DDoS attacks and analyzing network traffic patterns.

- **Random Forest** was chosen for its robustness, interpretability, and ability to handle high-dimensional data. As a tree-based ensemble method, it effectively captures non-linear relationships in traffic features and is resistant to overfitting due to its use of multiple decision trees.
- **CNN** is well-suited for identifying local spatial dependencies in data. In the context of network traffic, CNN can extract hierarchical patterns from input features, making it effective for detecting abrupt changes or anomalies in traffic behavior typically associated with DDoS attacks.
- **BiLSTM** was selected for its ability to learn from sequential and time-dependent data. Since network traffic is inherently temporal, BiLSTM networks can capture long-range dependencies in both past and future directions, which is valuable for identifying slow or stealthy DDoS attack patterns.
- **Ensemble Approach** combines the strengths of multiple models to improve detection accuracy and robustness. By aggregating predictions from different algorithms, the ensemble reduces the risk of relying on a single model's limitations and improves generalization performance across diverse attack types.

To ensure optimal performance, key hyperparameters such as the learning rate, number of layers, and dropout rate were carefully tuned to strike a balance between model complexity and generalization. Overfitting was mitigated through techniques such as cross-validation and regularization.

The dataset was divided into 70% for training, 20% for testing, and 10% for validation, allowing for effective learning, robust evaluation, and fine-tuned hyperparameter optimization. Model training involved experimentation with different epochs and batch sizes, while performance was monitored through loss and accuracy metrics, enabling real-time adjustments and improved training outcomes.

3.5.1 1D-CNN Based Model

In this project, we developed a CNN model specifically tailored for binary classification tasks, with a primary focus on detecting DDoS attacks. The architecture comprises two 1D convolutional layers with increasing filter sizes (32 and 64), each followed by max pooling and dropout layers to reduce overfitting and extract robust features. The output is then flattened and passed through a fully connected dense layer with 128 neurons, followed by another dropout layer. Finally, a single neuron with a sigmoid activation function generates the binary classification output.

This architecture, illustrated in Figure 3.12, is well-suited for processing structured and sequential input data, such as network traffic flows, where capturing temporal dependencies and local feature patterns is essential for accurate classification. The proposed CNN model leverages multiple layers to extract and learn hierarchical representations of input features, enhancing its ability to distinguish between benign and malicious traffic. A summary of the CNN architecture is presented in Table 3.6.

The model is compiled using the Adam optimizer, known for its adaptive learning rate and efficient convergence. The loss function selected is binary cross-entropy, which is most appropriate for binary classification tasks such as distinguishing between benign traffic and DDoS attacks.

Table 3.6: CNN Architecture for Binary DDoS Attack Detection

Layer Type	**Details**
Conv1D	32 filters, kernel size = 3, activation = ReLU
MaxPooling1D	Pool size = 2
Dropout	Rate = 0.25
Conv1D	64 filters, kernel size = 3, activation = ReLU
MaxPooling1D	Pool size = 2
Dropout	Rate = 0.25
Flatten	Converts 2D feature maps to 1D feature vector
Dense	128 units, activation = ReLU
Dropout	Rate = 0.5
Dense (Output)	1 unit, activation = Sigmoid

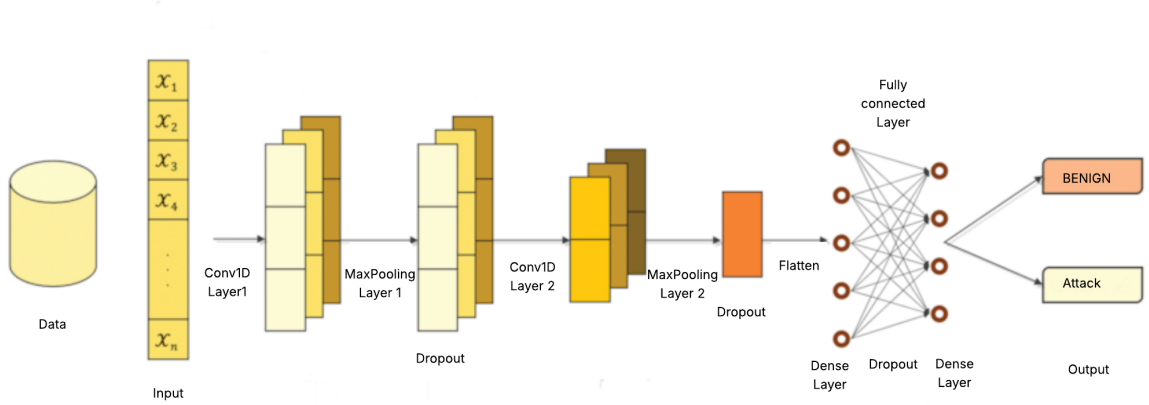


Figure 3.12: CNN Model Architecture

3.5.2 BiLSTM-Based Model

To capture the temporal dependencies present in sequential network traffic data, we implemented a BiLSTM model tailored for binary classification. This model is particularly effective in learning both forward and backward temporal patterns, thereby enhancing its capability to detect complex anomalies such as DDoS attacks.

The architecture, illustrated in Figure 3.13, begins with a BiLSTM layer comprising 64 units, enabling the model to learn patterns from both directions of the input sequence. This bidirectional mechanism significantly improves the model's ability to differentiate between normal and malicious traffic behaviors. To stabilize and accelerate training, batch normalization is applied to the output of the BiLSTM layer. A dropout layer with a rate of 30% follows, serving as a regularization mechanism to mitigate overfitting.

A `GlobalAveragePooling1D` layer is subsequently employed to reduce the temporal dimension by computing the average of each feature map across time steps, effectively summarizing the sequence into a fixed-length vector. This representation is then passed to a dense layer with 32 units and a LeakyReLU activation function, which allows a small gradient when the neuron is inactive, improving learning dynamics. A second dropout layer is added to further enhance generalization. The final layer is a dense layer with a single neuron and a sigmoid activation function, producing a probability that indicates whether the input traffic is benign or part of a DDoS attack.

A summary of the BiLSTM architecture is presented in Table 3.7.

Table 3.7: BiLSTM Architecture for Binary DDoS Attack Detection

Layer Type	**Details**
BiLSTM	64 units, outputs sequence for each timestep
Batch Normalization	Normalizes output to improve stability and training speed
Dropout	Rate = 0.3 (30%)
GlobalAveragePooling1D	Aggregates temporal features into a fixed-length vector
Dense	32 units, activation = LeakyReLU
Dropout	Rate = 0.3 (30%)
Output Layer	Dense (1 unit), activation = Sigmoid

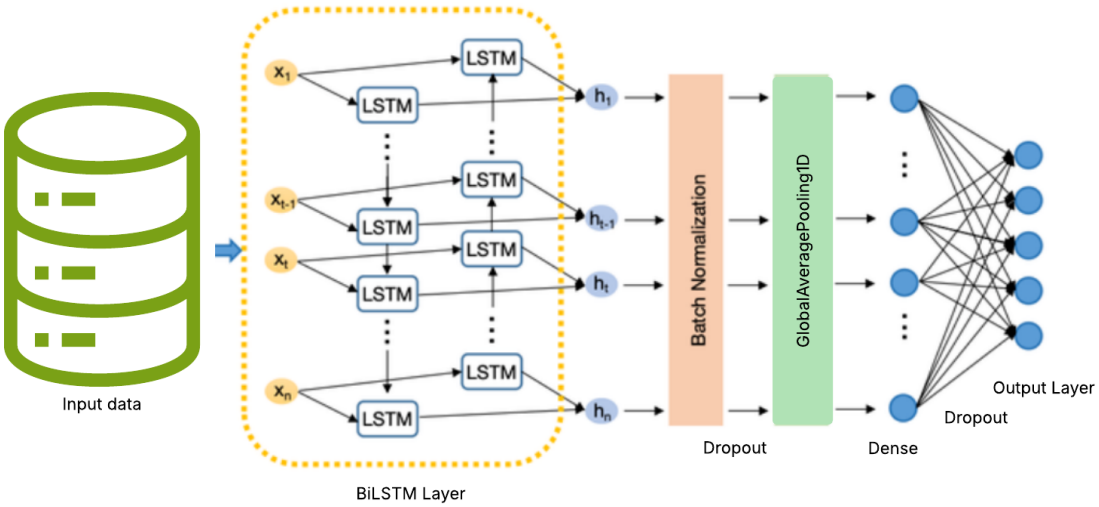


Figure 3.13: BiLSTM Model Architecture

The model is trained using the Adam optimizer due to its fast convergence properties and robustness. Binary cross-entropy is used as the loss function, which is suitable for binary classification tasks. Additionally, early stopping and learning rate reduction callbacks are employed to prevent overfitting and ensure efficient convergence.

3.5.3 Random Forest-Based Model

Our Random Forest model is specifically designed to handle the high-dimensional and structured nature of network traffic data. This architecture is particularly well-suited for tabular datasets such as network flow features due to its robustness to noise and its ability to handle both numerical and categorical data effectively.The model architecture is composed of the following components:

- **Input Layer:** Receives a fixed-size feature vector composed of 15 selected and normalized flow-based features. These features capture statistical and protocol-level characteristics of the network traffic.
- **Ensemble of Decision Trees:** The core of the Random Forest consists of 250 independently trained decision trees. Each tree is trained on a different bootstrap sample from the training dataset and uses a random subset of features at each split. This introduces diversity among the trees and reduces the risk of overfitting.
- **Majority Voting Layer:** For each input sample, predictions from all trees are aggregated using a majority voting scheme to produce the final class label. This ensemble decision process enhances prediction stability and improves generalization. Class labels are defined as 0 for `BENIGN` and 1 for `DDoS`.

3.5.4 Ensemble Model

In order to leverage the strengths of different machine learning approaches, we designed an ensemble architecture combining three complementary models: a Random Forest ,BiLSTM and CNN. This choice is based on our intention to capture:

- the explicit tabular features of network flows (effectively modeled by Random Forest).
- the local spatial patterns in tabular data (identified by the CNN).
- the temporal dynamics of flow sequences (learned by the BiLSTM).

Each model is trained independently using the same preprocessed dataset. During the inference stage, their outputs are fused using a weighted voting scheme to form the final decision, as illustrated in Figure 3.14 .

The ensemble score is computed using the following formula:

$$\text{Score}_{\text{ensemble}} = a \cdot \text{Score}_{\text{BiLSTM}} + b \cdot \text{Score}_{\text{RandomForest}} + c \cdot \text{Score}_{\text{CNN}} \tag{3.1}$$

where a, b, and c are the voting weights assigned to BiLSTM, Random Forest, and CNN respectively. These weights are constrained such that $a + b + c = 1$, and they are dynamically determined based on the validation accuracy of each individual model.

A classification threshold of 0.5 is then applied to the aggregated ensemble score to predict the traffic class: BENIGN (label 0) or DDoS (label 1). This ensemble method aims to mitigate the individual weaknesses of each model, enhance generalization performance, and improve robustness in handling dynamic traffic patterns, particularly within the challenging and heterogeneous context of 5G network environments.

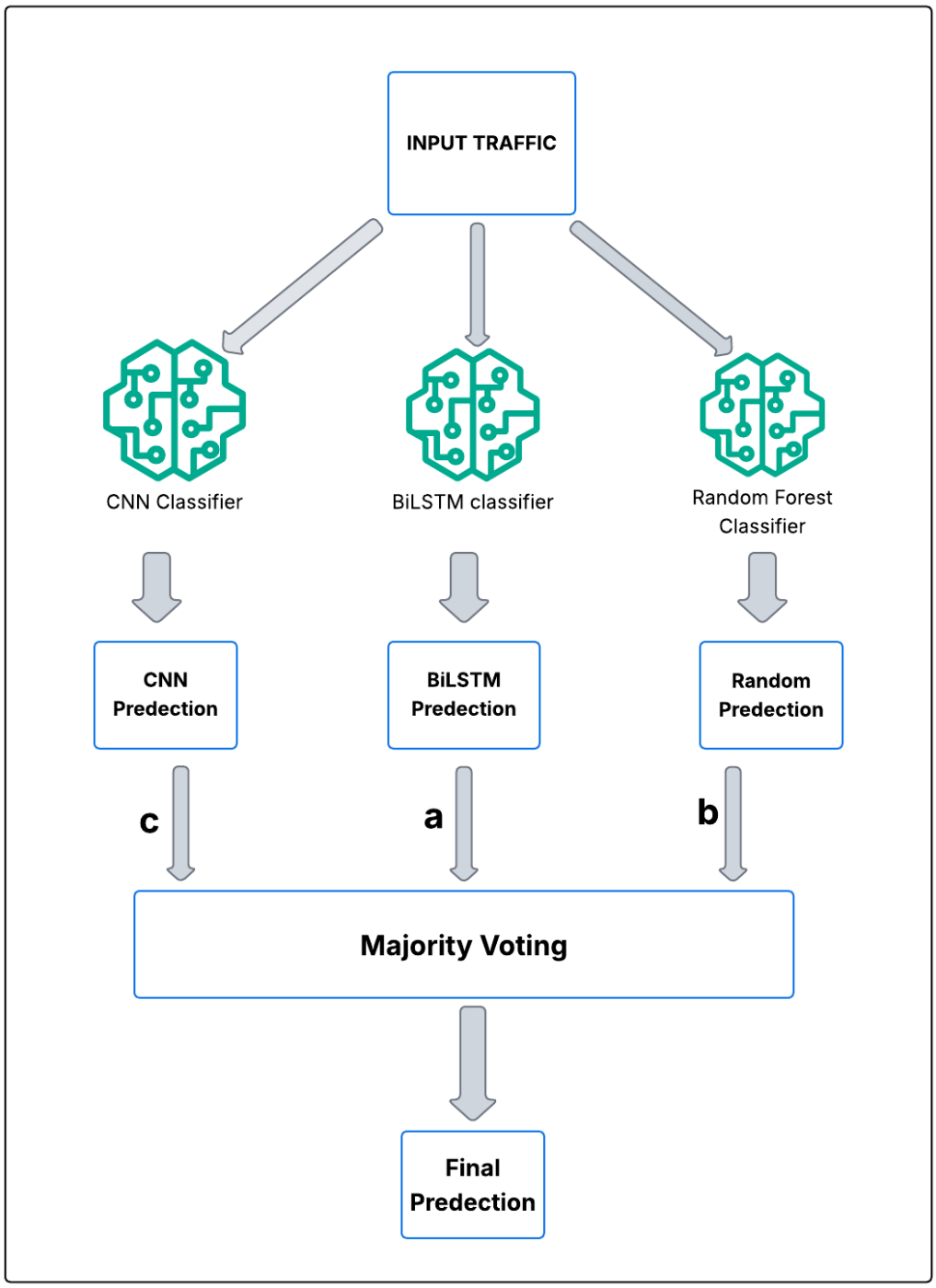


Figure 3.14: Ensemble architecture

Table 3.8: Comparative Table of Models for DDoS Attack Detection

Model Name	**Model Type**	**Features**	**Architecture**	**Optimizer**	**Loss Function**	**Overfitting Control**	**Output Type**
Random Forest	ML	15	250 decision trees with bootstrap sampling, majority voting prediction	N/A	N/A	Tree diversity, no backpropagation	Class label: 0/1
1D-CNN	DL	10	2×Conv1D (32,64), MaxPooling, Dropout (0.25,0.5), Dense(128), Sigmoid output	Adam	Binary Cross-Entropy	Dropout, pooling layers	Probability
BiLSTM	DL	10	BiLSTM(64), BatchNorm, Dropout(30%), GlobalAvgPooling, Dense(32, LeakyReLU), Sigmoid	Adam	Binary Cross-Entropy	Dropout, early stopping	Probability
Hybrid Ensemble	Hybrid Model	15	Weighted voting: a·BiLSTM + b·RF + c·CNN ($a+b+c=1$)	N/A	Binary Cross-Entropy	Model fusion strategy	Final decision 0/1

3.5.5 Comparative Overview of the Designed Models

In Table 3.8, we present a comparative analysis of the models we employed for DDoS attack detection. Each model was selected for its unique strengths in handling different aspects of network traffic. Our Random Forest model offers robustness and interpretability for structured data using an ensemble of decision trees. The 1D-CNN model allows us to capture local spatial patterns through convolutional and pooling layers, which is effective in identifying abrupt traffic anomalies. With the BiLSTM model, we were able to learn temporal dependencies in both directions of traffic flow, making it particularly suitable for detecting stealthy or evolving attack behaviors. To enhance detection accuracy and resilience, we designed a Hybrid Ensemble that combines the outputs of these three models using a weighted voting scheme. This ensemble approach enabled us to mitigate the limitations of individual models and improve our system's ability to generalize across diverse attack scenarios.

3.6 Detection Framework

Our detection framework, illustrated in Figure 3.15, integrates a ML-based Intrusion Detection System (IDS) within a simulated 5G network using OAI as the core platform. The IDS is embedded in each dedicated UPF instance per slice, enabling per-UE traffic monitoring and DDoS attack detection. It operates by capturing network traffic in real time, extracting relevant flow-level features, and classifying this data using a trained ML model. Upon detecting an attack, the system can trigger alerts.

This modular design ensures independent detection and response capabilities across different slices, promoting scalability, isolation, and efficient protection of network resources. Figure 3.16 illustrates the positioning of the IDS within the UPF and its inter-

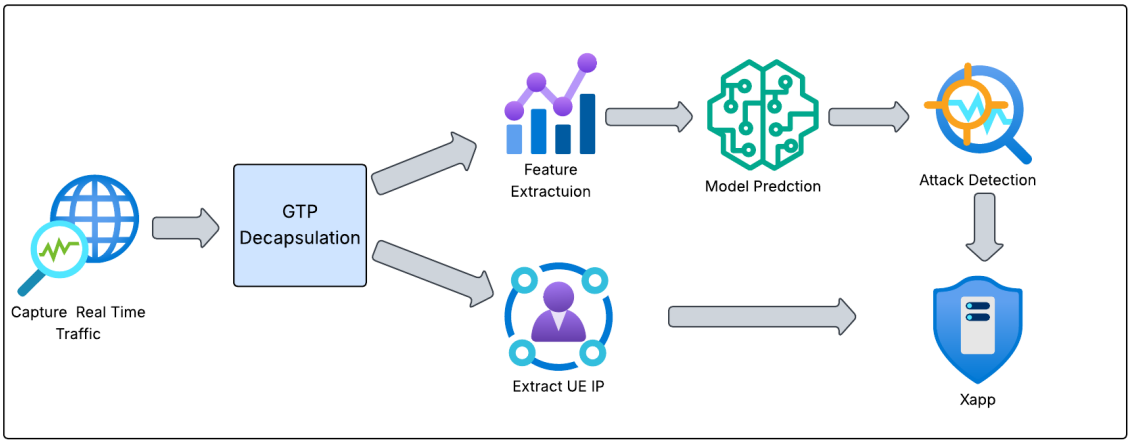


Figure 3.15: Detection Framework

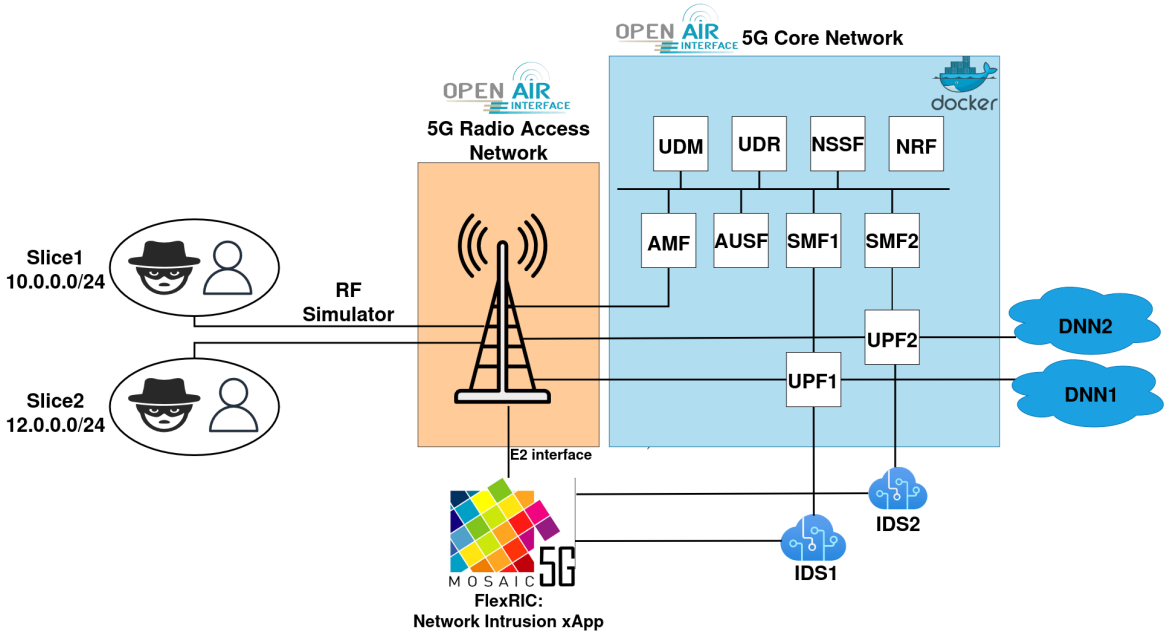


Figure 3.16: IDS Integration within the 5G Network

action with other components of the 5G network.

3.6.1 Packet Capture and Decapsulation

The IDS captures GTP-U packets from the `tun0` interface in the UPF of the 5G network. Specifically, it filters packets originating from the targeted network slice using a subnet filter. The following code snippet illustrates the packet capture loop:

Listing 3.7: Packet Capture Function

```
def capture_traffic():
    subnet_filter = "src net 10.0.0.0/24"
    while True:
        try:
            sniff(iface='tun0', filter=subnet_filter, prn=process_packet, store=
                ↪ False)
        except KeyboardInterrupt:
            logging.info("Packet capture stopped by user.")
            break
        except Exception as e:
            logging.error(f"Error during packet capture: {e}")
            time.sleep(10)
```

Once a GTP-U packet is captured, the IDS performs decapsulation to extract the inner IP packet, which contains the actual UE traffic. From this inner IP packet, the UE's IP address is retrieved and used as a key identifier for further analysis. The following code illustrates the GTP-U decapsulation and subsequent extraction of the IP address:

Listing 3.8: Packet Decapsulation and Processing

```
def process_packet(pkt):
    ip_pkt, teid = decapsulate_gtp(pkt)
    if not ip_pkt or IP not in ip_pkt:
        return

    ue_ip = ip_pkt[IP].src
    sliding_windows[ue_ip].append(ip_pkt)
```

For each UE, the IDS maintains a sliding window of 40 packets. When this window reaches its maximum size, it triggers the feature extraction and prediction processes. The window then advances with an overlap, enabling the system to analyze both flow-level behaviors and point anomalies effectively. This architecture ensures that the IDS can process multiple UE flows in real time, maintaining a continuous and efficient monitoring system within the 5G network infrastructure.

3.6.2 Feature Extraction

Once packets are captured from the network interface and grouped into sliding windows, the system extracts a set of features that capture the temporal and behavioral characteristics of the network traffic. This transformation of raw packet-level data into a structured set of features is essential for model-based analysis.

The feature extraction process leverages the capabilities of Scapy, a Python-based packet manipulation library, to analyze packets and compute a rich set of statistical,

protocol, and flow-level features. These features are designed to capture both the intrinsic properties of the packets and their aggregated flow behavior.

The key features computed per window include: Flow Duration, Protocol Flags, and Header Attributes. The following Python function illustrates the core feature extraction logic using Scapy:

Listing 3.9: Feature Extraction

```
import pandas as pd
from scapy.all import *

def extract_selected_features(packets):
    packet_sizes = [len(pkt) for pkt in packets]
    forward_packets = [pkt for pkt in packets if pkt[IP].src == ue_ip]
    inbound_packets = [pkt for pkt in packets if pkt[IP].dst == ue_ip]
    flow_duration = (packets[-1].time - packets[0].time) if packets else 1

    features = {}
    features['Min Packet Length'] = min(packet_sizes) if packet_sizes else 0
    features['ACK Flag Count'] = sum(1 for pkt in packets if TCP in pkt and pkt
        ↪ [TCP].flags & 0x10)
    features['Fwd Packet Length Min'] = min([len(pkt) for pkt in
        ↪ forward_packets]) if forward_packets else 0
    features['Flow Bytes/s'] = sum(len(pkt) for pkt in packets) / flow_duration
        ↪  if flow_duration else 0
    features['Bwd Packets/s'] = len(inbound_packets) / flow_duration if
        ↪ flow_duration else 0
    features['Packet Length Std'] = pd.Series(packet_sizes).std() if len(
        ↪ packet_sizes) > 1 else 0
    features['Total Backward Packets'] = len(inbound_packets)

    return pd.DataFrame([features])
```

This approach ensures that each window of packets is transformed into a structured feature vector that matches the model's expectations. Once features are extracted, they must be normalized and aligned with the model's input format. This step is critical because machine learning models are sensitive to feature scales and input order. The system uses a pre-trained 'StandardScaler', fitted during the training phase, to standardize the features. Before normalization, a verification step ensures that all required features (defined in FINAL_FEATURES) are present. The following function demonstrates the normalization process:

Listing 3.10: Feature Normalization and Alignment

```
scaler = joblib.load("scaler.pkl")
def normalize_and_align(df, required_columns):
    for col in required_columns:
        if col not in df:
            df[col] = 0
    df = df[required_columns].fillna(0)
    df_normalized = pd.DataFrame(scaler.transform(df), columns=required_columns
        ↪ )
    return df_normalized
```

This ensures that the input data is consistent with the model's training format, preventing issues such as mismatched feature dimensions or out-of-distribution values during real-time predictions.

3.6.3 Prediction and Action

A pre-trained machine learning model classifies aggregated network flows by analyzing key features extracted from packet windows, determining whether each flow represents normal traffic or a potential DDoS threat. For each source entity, the system continuously monitors and maintains a count of both normal and anomalous flows.

Upon detecting suspicious activity, the Intrusion Detection System (IDS) transmits a structured message to the xApps via a TCP socket, enabling real-time coordination and response.

Listing 3.11: Establishing TCP Connection and Capturing Traffic

```
def main():
    global sock
    while True:
        try:
            sock = socket.create_connection(("192.168.70.129", 8080))
            logging.info("Connected to the server.")
            break
        except ConnectionRefusedError:
            logging.error("Server not ready. Retrying in 5 seconds...")
            time.sleep(5)
    capture_traffic()
```

The transmitted message includes critical information such as the source UE identifier, the volume of anomalous and legitimate flows, and the corresponding network slice parameters, namely the SST and SD. This real-time communication facilitates rapid decision-making and supports dynamic mitigation strategies within the 5G architecture.

Listing 3.12: Formatted Message and Sending via Socket

```
message = f"sst:{sst},sd:{sd},normal:{total_normal_packets},anomaly:{
    ↪ total_ddos_packets}"
send_message_to_server(message)
```

3.7 Mitigation Framework

The mitigation framework we developed is designed to act as a reactive layer in our 5G enviroment, built on the FlexRIC platform. Its main objective is to respond to detected anomalies by taking direct RAN level actions to free up resources and protect the network infrastructure.

Specifically, the mitigation framework is implemented as a custom xApp that targets DDoS attacks on critical 5G core components such as the UPF. When an attack is detected on a specific network slice, specialized detection systems are used to define the responsible UE(s). These systems communicate the relevant information to the xApp through a dedicated socket interface.

Upon receiving this information, the xApp reacts using the E2SM-RC (Radio Control) service model. It triggers a targeted RRC Release procedure for the identified malicious UE, effectively disconnecting it from the network. This allows the xApp to isolate suspicious behavior at the UE level without compromising the rest of the slice's operations.

In parallel, our system enforces a static and balanced PRB allocation strategy by equally dividing them between the network slices, ensuring fairness and stability in the distribution of radio resources. This configuration remains fixed, providing predictable resource isolation even under attack scenarios.

The result is a responsive mitigation mechanism that operates close to the RAN, by enforcing real time control over UE connectivity. the full mitigation logic is illustrated in Figure 3.17.

3.7.1 Initial xApp from FlexRIC

As a foundation for our work, we adopted a reference xApp implementation provided by the FlexRIC project. This xApp was built using the standardized O-RAN E2SM-RC service model and was designed to demonstrate basic control functionalities, particularly those related to slice level PRB (Physical Resource Block) quota enforcement in the RAN. which governs how available radio resources are distributed among multiple network slices within a gNB.

Key Features of the xApp

- **Slice Level Resource Allocation**: The xApp enforces policies for distributing PRBs among network slices using three configurable ratios defined in the 3GPP specification:
  - **Minimum PRB Ratio**: Guarantees a minimum share of PRBs for a slice, ensuring it always has resources even under high load.
  - **Maximum PRB Ratio**: Caps the maximum PRBs a slice can use, preventing resource monopolization.
  - **Dedicated PRB Ratio**: Reserves a fixed portion of PRBs exclusively for a slice, which cannot be reallocated to other slices.
- **Static Configuration**: The xApp uses hardcoded values for slice configurations in its current implementation. Only the `dedicated` PRB ratio is explicitly defined, while both the `minimum` and `maximum` PRB ratios are set to `0`, indicating that they are unset and not enforced by the gNB.

```
1
2  const char* sst_str[] = {"1", "1"};
3  const char* sd_str[] = {"1", "5"};
4  int dedicated_ratio_prb[] = {70, 30};
5  for (int i = 0; i < num_slice; i++) {
6    gen_rrm_policy_ratio_group(&RRM_Policy_Ratio_List->ran_param_val.lst->
       ↪ lst_ran_param[i],
7                               sst_str[i],
8                               sd_str[i],
9                               0, dedicated_ratio_prb[i], 0);
10 }}
```

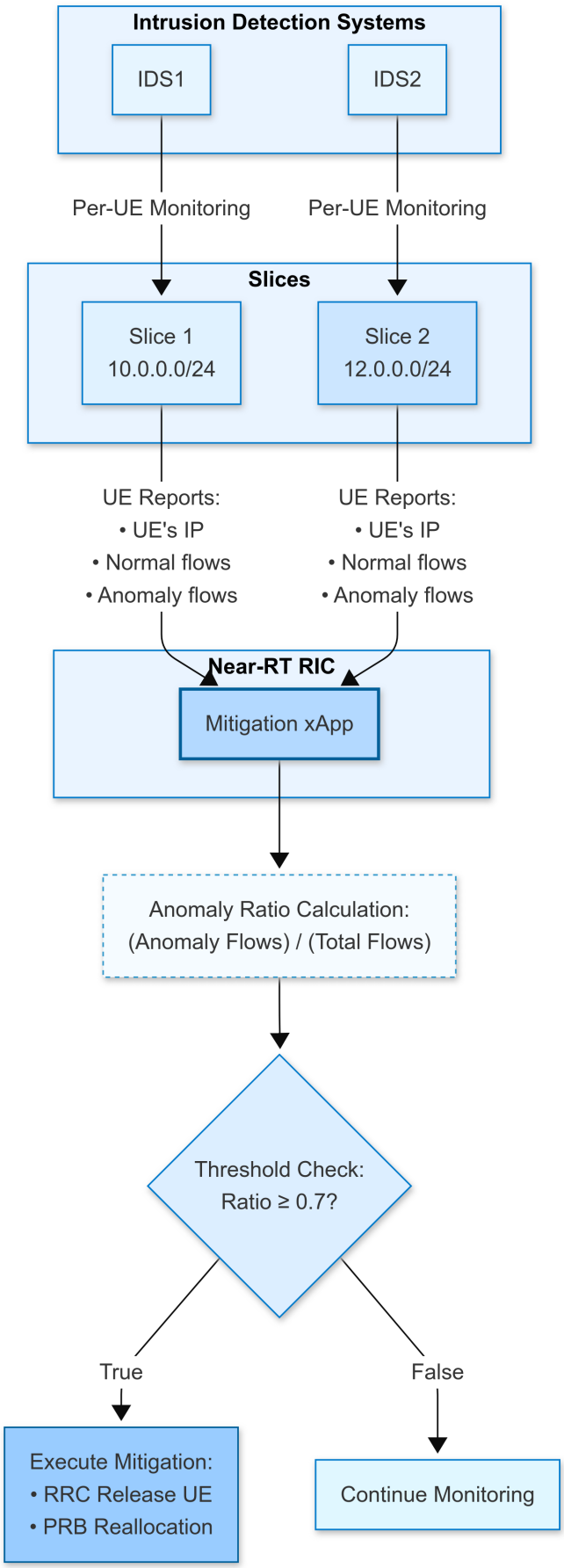


Figure 3.17: Mitigation strategy Diagram

These values allocate 70% of PRBs to the first slice (SST=1, SD=1) and 30% to the second slice (SST=1, SD=5).

- **E2SM-RC Control Procedures**: The xApp constructs and sends control messages to the gNB via the E2 interface. The messages adhere to the E2SM-RC Control Header (Format 1) and Control Message (Format 1) specifications.

Limitations and Extensions

- **Unequal PRB Allocation**: In its original form, the xApp applies hardcoded, unbalanced PRB ratios across slices (70/30), which may lead to unfair resource distribution and underutilization in lightly loaded slices.
- **No Security Mechanisms**: The initial xApp does not incorporate any form of security awareness or protection against malicious UEs, such as those involved in DDoS attacks targeting the core network.
- **Lack of Real-Time Response**: The control logic is static and does not respond dynamically to real-time events or changing network conditions at the UE level.
- **No External Integration**: The xApp operates in isolation and lacks connectivity with external analytics or anomaly detection engines that could provide valuable context for intelligent decision making.

In our extended implementation, we addressed these issues by enforcing equal PRB allocation across slices (50/50), which improves fairness and predictability. We also introduced a lightweight mitigation mechanism that integrates with two anomaly detection engines. Upon detecting suspicious UEs, the xApp is capable of issuing targeted RRC Release commands via the E2SM-RC, effectively disconnecting malicious devices in real time. These enhancements enable our system to provide better resource management and enhanced security at the RAN level, while maintaining the simplicity of static policy enforcement.

3.7.2 Custom Enhancements

To make the xApp truly effective for our DDoS attacks mitigation,we made several important enhancements. These changes were aimed at giving the xApp more awareness of network behavior and the ability to act quickly when something goes wrong.

We started by improving how the xApp communicates with the external IDS modules, enabling it to receive real time traffic data for each UE. We also built in multithreaded support so it can handle data from multiple IDS sources at the same time without delays or blocking. Most importantly, we added intelligent logic that allows the xApp to calculate anomaly levels per UE and take actions based on what it receives.

These custom improvements allow the xApp to go beyond passive monitoring and actively defend the network in real time.

Socket Communication and Multithreading

To support real time anomaly detection across multiple UEs, the xApp incorporates a multithreaded TCP server that manages concurrent communication with two independent

IDS clients. Each IDS monitors its own slice and periodically transmits per UE anomaly statistics to the xApp.

Upon startup, the `main()` function initializes the TCP server by creating a socket, binding it to a known address and port, and listening for incoming client connections. When an IDS client connects, a new thread is spawned using `pthread_create()` to handle communication for that client independently. This allows the xApp to process traffic insights from both slices.

Each thread receives a pointer to a `client_data_t` structure containing the client's socket and unique identifier. As connections are established, a global counter `clients_connected` is incremented to track how many IDS clients have joined. This counter is protected by a mutex to ensure thread safety. Once a client connects, the thread signals a condition variable, notifying the main thread of its arrival.

The main thread remains blocked on `pthread_cond_wait()` until all expected IDS clients have connected. This synchronization ensures that control operations don't proceed until all monitoring components are active.

```
pthread_mutex_lock(&lock);
clients_connected++;
pthread_cond_signal(&cond);
pthread_mutex_unlock(&lock);

pthread_t sniffer_thread;
if (pthread_create(&sniffer_thread, NULL, handle_ids, (void*)client_data) < 0)
    ↪ {
    perror("Could not create thread");
    return 1;}
pthread_detach(sniffer_thread);

pthread_mutex_lock(&lock);
while (clients_connected < IDS_CLIENTS) {
    pthread_cond_wait(&cond, &lock);}
pthread_mutex_unlock(&lock);
puts("Both IDS connected. Starting main loop...");
```

This multithreaded design ensures robust and scalable data ingestion. By separating client handling into parallel threads and synchronizing initialization, the xApp can reliably receive and process anomaly reports in real time. These statistics are later used to compute anomaly ratios and trigger targeted RAN control actions like RRC release based on per-UE behavior.

Client Handling

The `handle_ids()` function manages communication with each IDS client in its own thread, allowing the xApp to process data from multiple sources concurrently. Each thread runs a continuous loop, receiving structured messages from its assigned IDS client. These messages include essential information such as slice identifiers (SST and SD), the number of normal and anomalous packets, and the IP address of the UE.

When a message is received, it is parsed and the extracted values are stored in the shared `ue_data[]` structure. To ensure thread safety while multiple threads access this global data, updates are performed under a mutex lock.

Once the shared data is updated, the thread signals a condition variable to notify the main control logic that fresh data is available. This allows the xApp to react in real time if a UE shows a high anomaly ratio.

The core of this functionality is illustrated in the following loop:

```
while ((read_size = recv(sock, client_message, BUFFER_SIZE, 0)) > 0) {
    client_message[read_size] = '\0';

    printf("Received message from IDS %d: %s\n", ids_id + 1, client_message);

    int sst, sd, normal_count, anomaly_count;
    char ip[INET_ADDRSTRLEN] = {0};

    if (sscanf(client_message, "sst:%d,sd:%d,normal:%d,anomaly:%d,ue_ip:%15s",
              &sst, &sd, &normal_count, &anomaly_count, ip) == 5) {
        pthread_mutex_lock(&lock);
        ue_data[ids_id].sst = sst;
        ue_data[ids_id].sd = sd;
        ue_data[ids_id].normal_count = normal_count;
        ue_data[ids_id].anomaly_count = anomaly_count;
        strncpy(ue_data[ids_id].ip_address, ip, INET_ADDRSTRLEN);

        if (sd == 1) {
            if (strcmp(ip, "10.0.0.2") == 0) {
                ue_data[ids_id].rrc_ue_id = 1;
            } else if (strcmp(ip, "10.0.0.3") == 0) {
                ue_data[ids_id].rrc_ue_id = 2;
            }
        } else if (sd == 5) {
            if (strcmp(ip, "12.0.0.2") == 0) {
                ue_data[ids_id].rrc_ue_id = 3;
            } else if (strcmp(ip, "12.0.0.3") == 0) {
                ue_data[ids_id].rrc_ue_id = 4;
            }
        }
        messages_received++;

        pthread_cond_signal(&cond);
        pthread_mutex_unlock(&lock);

        printf("Parsed values for IDS %d: SST: %d, SD: %d, Normal: %d, Anomaly:
            ↪  %d, IP: %s, RRC UE ID: %d\n",
              ids_id + 1, sst, sd, normal_count, anomaly_count, ip, ue_data[
                  ↪ ids_id].rrc_ue_id);}
```

Static Mapping from IP Address to RRC UE ID A crucial part of this function is assigning a valid `RRC_UE_ID` to each UE. Since RRC Release messages require UE identifiers known to the gNB, and IDS clients only report UE IP addresses, we implemented a static mapping between IP addresses and RRC UE IDs. This was necessary because the gNB's internal identifiers are not directly accessible via IDS data.

The mapping is hardcoded and based on observations from **KPM measurement reports**, as shown in Figure 3.18. By monitoring KPM logs from the FlexRIC based xApp, we correlated UE IPs with their corresponding RRC UE IDs as recognized by the gNB.

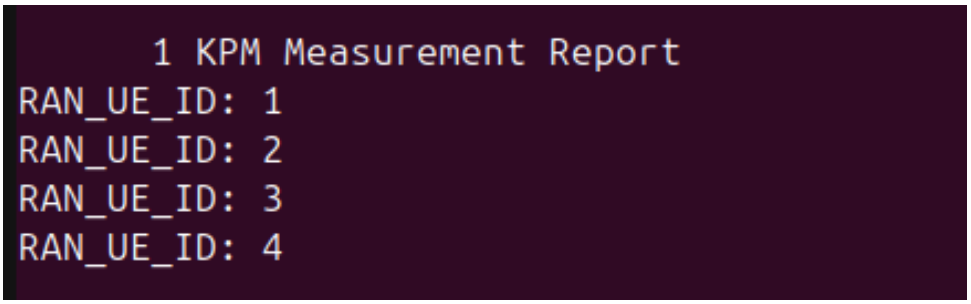


Figure 3.18: KPM Measurement Report Showing RAN_UE_IDs

This mapping allows the xApp to identify the correct RRC UE ID from the UE's IP, making it possible to issue an RRC Release via the E2 interface.

After updating the shared data, the function signals a condition variable to notify the main control thread that new IDS statistics are available.

Triggering RRC Release for Malicious UEs

In 5G networks, the RRC (Radio Resource Control) Release procedure is used to disconnect a UE from the radio network. This is typically triggered when a device becomes idle, moves out of coverage, or in our work exhibits malicious behavior. Releasing the RRC connection not only prevents the offending UE from continuing to impact the network but also frees up radio resources.

To implement this functionality, we extended the xApp to send RRC Release commands via the E2 interface using a simple yet effective mechanism. The setup includes a `telnet` server running on the E2 Agent, which listens for commands on TCP port 9090. The xApp uses the `netcat` (`nc`) utility to send commands to this server.

Once a UE is identified as malicious the xApp invokes the `rrc_release_ue()` function, passing the UE's RAN ID. This function spawns a new thread to handle the release asynchronously, ensuring that the main control loop remains responsive:

Listing 3.13: RRC Release Execution Logic

```
void rrc_release_ue(int ran_ue_id) {
    pthread_t thread;
    int* arg = malloc(sizeof(*arg));
    if (arg) {
        *arg = ran_ue_id;
        pthread_create(&thread, NULL, rrc_release_ue_thread, arg);
        pthread_detach(thread);
    }
}
```

The release thread constructs a shell command that echoes the release instruction and pipes it into `netcat`. The instruction takes the form `rrc release_rrc <RAN_UE_ID>` and is transmitted to the E2 Agent via TCP:

Listing 3.14: RRC Release Thread

```
void* rrc_release_ue_thread(void* arg) {
    int ran_ue_id = *((int*)arg);
    char command[256];
    snprintf(command, sizeof(command),
            "echo rrc release_rrc %d | nc %s 9090", ran_ue_id, SERVER_IP);
    system(command);
    free(arg);
    return NULL;
}
```

This mechanism enables real time control of UE disconnection. By tying this release logic to UE anomaly detection, we can rapidly respond to suspicious traffic behavior, protecting the network and its legitimate users from potential attacks.

Static PRB Allocation Enforcement

Once an attack is detected and the malicious UE is removed from the network through an RRC release, the xApp immediately reinforces static PRB allocation to rebalance resources across slices. This step is crucial to prevent the attack from continuing to affect the performance of other users.

The xApp achieves this by calling the `enforce_slicing()` function, which sends a control message to the gNB using the RC service model. This message instructs the gNB to reapply a fixed PRB split, typically a 50/50 distribution between the two slices. This ensures that each slice regains its intended share of radio resources, keeping the network fair and stable.

By tying together per UE anomaly detection with slice level PRB enforcement. the xApp maintains both security and service quality. Even in the presence of malicious traffic, this approach helps ensure that legitimate users continue to get the resources they need.

Per-UE Anomaly Ratio

To enable control decisions, the xApp computes an anomaly ratio for each UE based on data received from the IDS clients. Each IDS client is responsible for monitoring traffic flows associated with specific UEs and sends statistics to the xApp in the form of normal and anomalous flow counts. These values are parsed and stored in the global `ue_data[]` structure.

The following code block shows the main control logic that processes these statistics:

Listing 3.15: Anomaly Ratio Calculation and RRC Release Decision

```
void apply_rrc_release(e2_node_arr_xapp_t nodes) {
    for (int i = 0; i < IDS_CLIENTS; i++) {
        int total_flows = ue_data[i].normal_count + ue_data[i].anomaly_count;
        double anomaly_ratio = (total_flows > 0) ?
                            (double)ue_data[i].anomaly_count / total_flows : 0;

        if (anomaly_ratio > 0.7) {
            printf("Client %d (RAN UE ID: %d) has %.0f%% anomaly. Triggering RRC
                ↪ release.\n",
```

```
                i + 1, ue_data[i].rrc_ue_id, anomaly_ratio * 100);
            rrc_release_ue(ue_data[i].rrc_ue_id);
        }
    }

    enforce_slicing(nodes);
}
```

For each connected IDS client the xApp retrieves the total number of normal and anomaly flows, computes the **anomaly ratio**, then compares this ratio to a predefined threshold (0.7).

If a UE's anomaly ratio exceeds the threshold, the xApp invokes the `rrc_release_ue()` function with the corresponding RRC UE ID. This action instructs the RAN to release the UE, effectively removing it from the radio network and mitigating the threat it may pose.

This mechanism allows the xApp to autonomously detect and act upon UE-specific anomalies based on real-time traffic behavior, with decisions derived from decentralized IDS monitoring but enforced centrally through RRC control.

After each RRC release, the xApp also calls the `enforce_slicing()` function to reapply static PRB slicing policies across the slices. This ensures that once an attack is mitigated by disconnecting the malicious UE, each network slice continues to receive its guaranteed share of radio resources.

This mechanism allows the xApp to autonomously detect and respond to anomalies while maintaining fair and consistent resource isolation between slices.

Chapter 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter evaluated the performance of our proposed solution by testing the effectiveness of the ML/DL models on both the benchmark dataset CICDDoS2019 and the enhanced version, CICDDoS-5G. A comparison between the models was carried out to highlight the strengths and weaknesses of each one. We then moved on to present the results of both the detection and mitigation frameworks in real time scenarios.

To further demonstrate the effectiveness of our defense framework, we analyzed key network performance indicators such as round trip time (RTT) and throughput. These metrics clearly showed the significant impact DDoS attacks can have on the user experience, as well as the ability of our framework to effectively counter and mitigate those effects.

4.2 Model Evaluation

The performance of the models was evaluated using two datasets: the original CICDDoS2019 dataset and the enhanced dataset (CICDDoS-5G), which aims to improve the detection of DDoS attacks in a 5G environment by incorporating traffic patterns adapted to 5G network characteristics.

Model performance was assessed using well-established metrics: accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of each model's ability to correctly detect and classify DDoS attacks. Accuracy reflects the overall correctness of the model's predictions, while precision measures the proportion of true positives among all positive predictions. Recall (or sensitivity) indicates the model's ability to identify actual attack instances. The F1-score, which combines precision and recall into a single value, offers a balanced evaluation of the model's performance.

Table 4.1 summarizes the performance metrics for the four models.

Table 4.1: Accuracy Comparison

Dataset	**Random Forest**	**CNN**	**BiLSTM**	**Ensemble**
CICDDoS2019	99.90	98.6	98.83	99.0
CICDDoS-5G	90.38	73.92	84.18	84.24

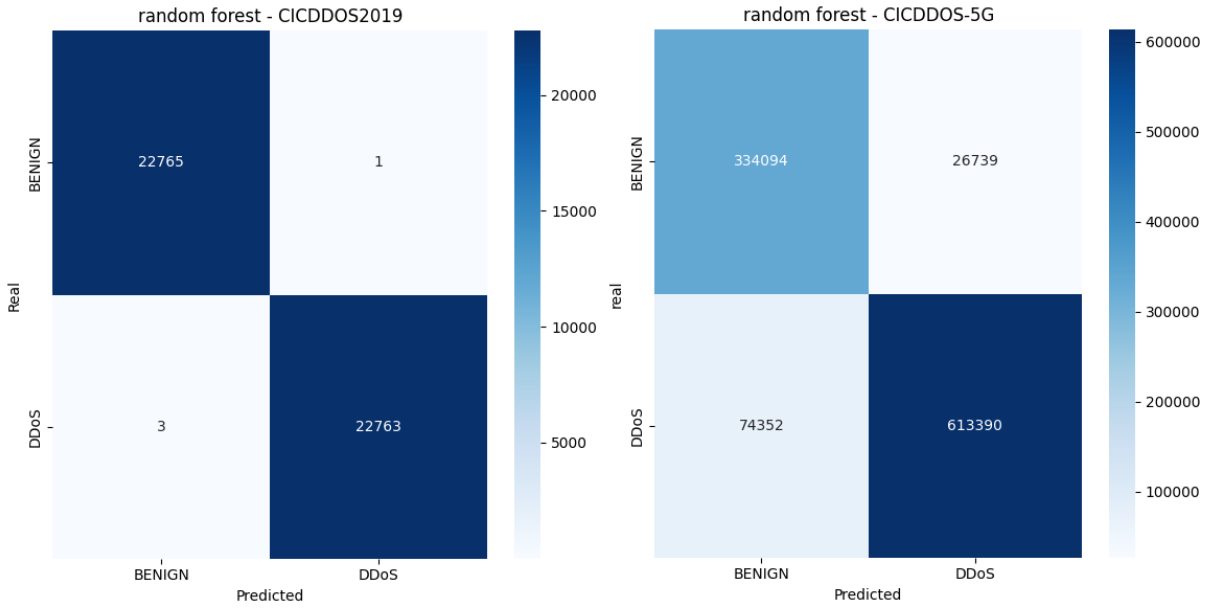


Figure 4.1: Confusion Matrix – Random Forest

4.2.1 Random Forest Analysis

The Random Forest model achieved a very high accuracy of 99.9% on the CICDDoS2019 dataset, demonstrating its strong capability in feature integration and decision-making for DDoS detection. The confusion matrix in Figure 4.1 for this dataset showed mostly correct predictions, with few false positives and false negatives, highlighting the model's ability to effectively distinguish between benign and DDoS traffic.

Random Forest is particularly appreciated for its robustness against overfitting, its ability to handle high-dimensional data, and its relatively low computational requirements compared to deep learning models. Its ensemble of decision trees provides a reliable assessment of feature importance, and the model is highly effective for classifying data with clear decision boundaries.

However, when applied to the enhanced dataset CICDDOS-5G comprising more diverse and realistic traffic, the confusion matrix in Figure 4.1 revealed an increase in false positives and false negatives, and the model's accuracy dropped to 90.3%. This performance gap illustrates Random Forest's limitations in adapting to more complex data without extensive feature engineering. Furthermore, while Random Forest offers better interpretability than deep learning models, it may struggle to handle the volume and velocity of real-time 5G traffic or to capture more subtle, high-level patterns.

4.2.2 CNN Model Analysis

The CNN model achieved an accuracy of 98.6% on the CICDDoS2019 dataset but dropped to 73.92% on the enhanced CICDDoS-5G dataset, as shown in Figure 4.2. This demonstrates its ability to learn from structured, feature-rich data while highlighting its limitations in handling diverse and complex traffic patterns. CNN is effective at capturing spatial and temporal dependencies within DDoS attacks, but it struggles to generalize when exposed to varied data. This suggests that additional feature engineering may be

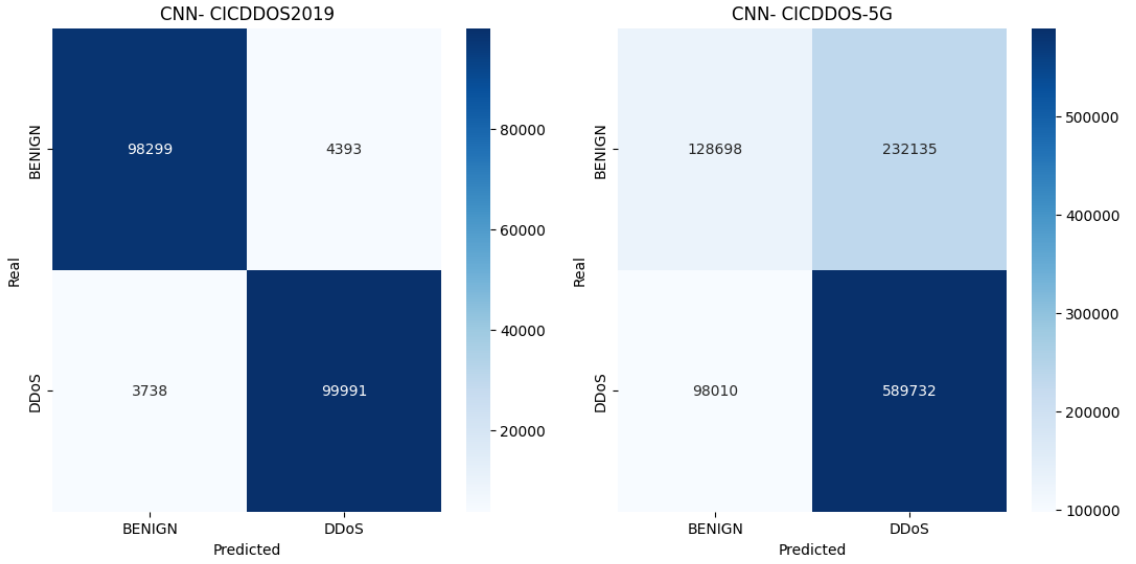


Figure 4.2: Confusion Matrix CNN

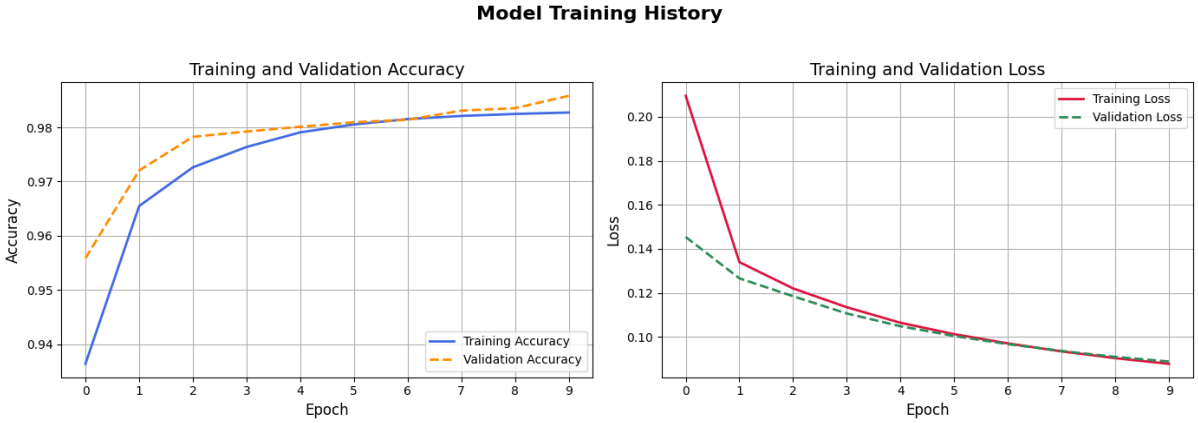


Figure 4.3: CNN Training and Validation Accuracy and Loss Curves

required to enhance its adaptability and robustness in real-world 5G scenarios.

The training and validation curves in Figure 4.3 illustrate the following key points:

- **Effective Learning:** Both accuracy and loss curves show steady improvement, indicating successful training.
- **Generalization:** Validation accuracy reaches 98.7%, slightly higher than training , suggesting good generalization without overfitting.
- **Stable Convergence:** The curves converge smoothly, with loss values decreasing consistently, confirming learning stability.
- **Intersection Insight:** At epoch 5, the training and validation curves intersect, highlighting the model's balanced learning.

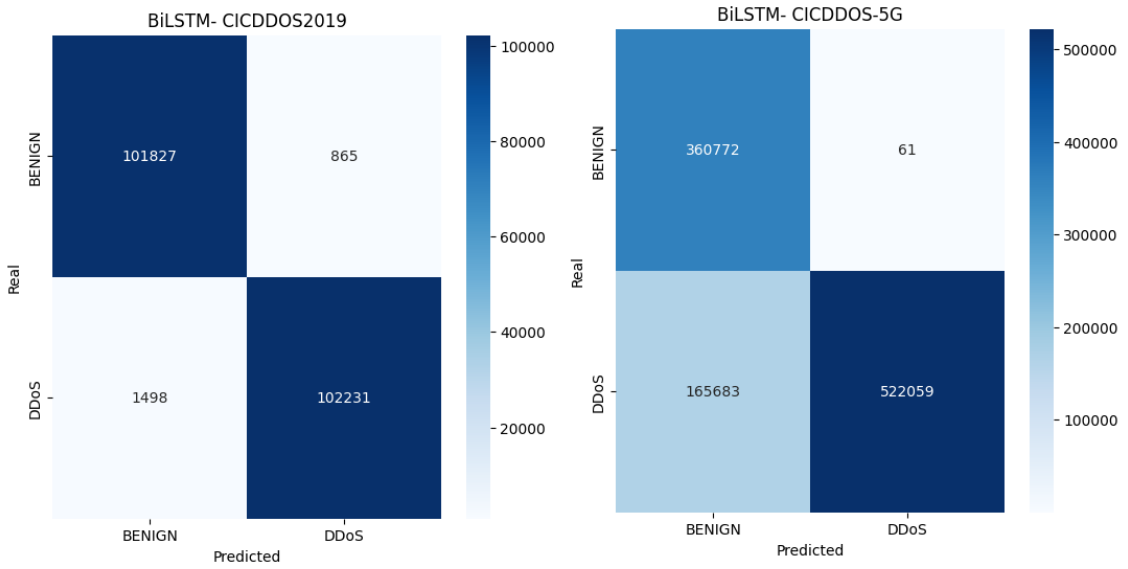


Figure 4.4: Confusion Matrix BiLSTM

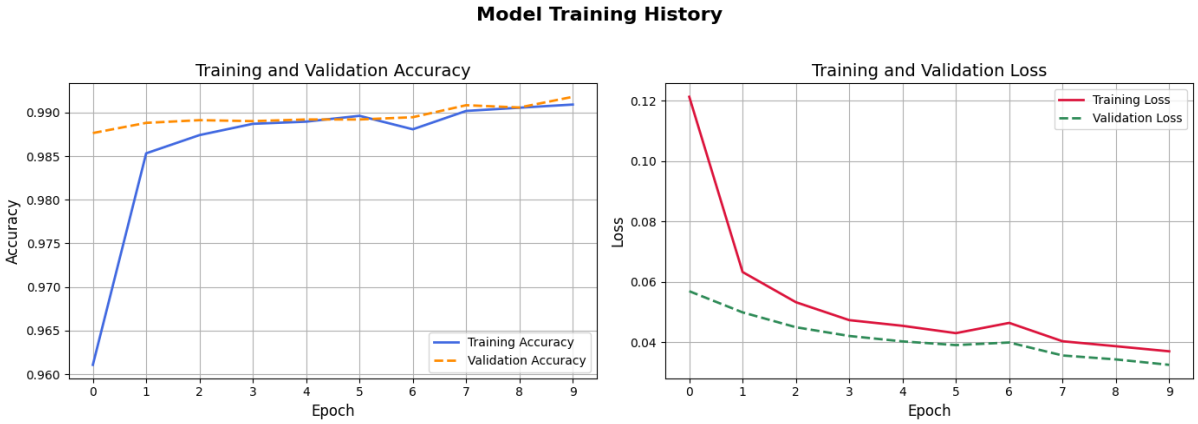


Figure 4.5: BiLSTM Training and Validation Accuracy and Loss Curves

4.2.3 BiLSTM Model Analysis

The BiLSTM model achieved a strong accuracy of 98.83% on the CICDDoS2019 dataset and 84.19% on the enhanced dataset CICDDOS-5G (Figure 4.4), demonstrating its ability to learn sequential patterns from network traffic data. The high performance on CICDDoS2019 confirms BiLSTM's strength in capturing temporal dependencies, as it processes data in both forward and backward directions. However, the drop in accuracy on the enhanced dataset suggests challenges in generalizing to more complex or varied data, as BiLSTM models can be sensitive to shifts in data distribution and variability.

The training and validation curves in Figure 4.5 illustrate the following key points:

- The BiLSTM model exhibits rapid convergence, achieving over **98.5%** accuracy by epoch 2, indicating efficient learning dynamics.
- The validation accuracy consistently surpasses the training accuracy, reaching up

to **99.2%**, which suggests a high degree of generalization to unseen data.

- Both training and validation loss curves exhibit a smooth and monotonic decrease, with validation loss remaining slightly lower, reflecting stable optimization and effective regularization.

- The absence of a significant gap between training and validation metrics indicates that the model is neither overfitting nor underfitting, maintaining strong predictive performance across the dataset.

4.2.4 Ensemble Model Analysis

To improve detection performance on the enhanced dataset, we developed an ensemble model that integrates the predictions of three complementary classifiers: BiLSTM, Random Forest, and CNN. This approach was designed to leverage the individual strengths of each model, BiLSTM's capability to capture temporal dependencies in sequential data, Random Forest's effectiveness in handling structured tabular features and delivering high interpretability, and BiLSTM's ability to extract local spatial patterns within flow level traffic data.

Each model was trained independently on the same preprocessed dataset. During inference, the final decision is made through a weighted voting scheme, which combines the prediction scores of each base model according to the following formula:

$$\text{Score}_{\text{ensemble}} = a \cdot \text{Score}_{\text{BiLSTM}} + b \cdot \text{Score}_{\text{RandomForest}} + c \cdot \text{Score}_{\text{CNN}} \tag{4.1}$$

where $a = 0.25$, $b = 0.5$, and $c = 0.25$. These weights were selected empirically based on the validation performance of each model, with Random Forest achieving the highest standalone accuracy and thus receiving the largest voting weight. The ensemble output score is then thresholded at 0.5 to classify network traffic as either benign(label 0) or DDoS(label 1).

The ensemble model achieved an overall accuracy of 84.18% on the enhanced dataset. According to the classification report, it attained a precision of 0.69 and recall of 1.00 for the benign class, and a precision of 1.00 and recall of 0.76 for the DDoS class. The macro average F1-score was 0.84, and the weighted average F1-score reached **0.85**, indicating a well-balanced and effective model performance. The confusion matrix is illustrated in fig4.6.

The ensemble architecture successfully mitigates the limitations of individual classifiers, offering improved generalization and robustness, especially in the context of dynamic and heterogeneous 5G network environments. Nevertheless, the relatively low precision for benign traffic (0.69) suggests a tendency to produce false positives, which could be a concern in real world deployments.

The strengths of the ensemble model include its ability to combine the complementary strengths of individual models, improving robustness and handling diverse traffic patterns more effectively. It achieves better generalization compared to single models and reduces the risk of overfitting to a specific dataset. However, the ensemble's precision for benign traffic is relatively low (0.69), meaning it tends to classify some benign traffic as DDoS, which could lead to false positives in a real world scenario.

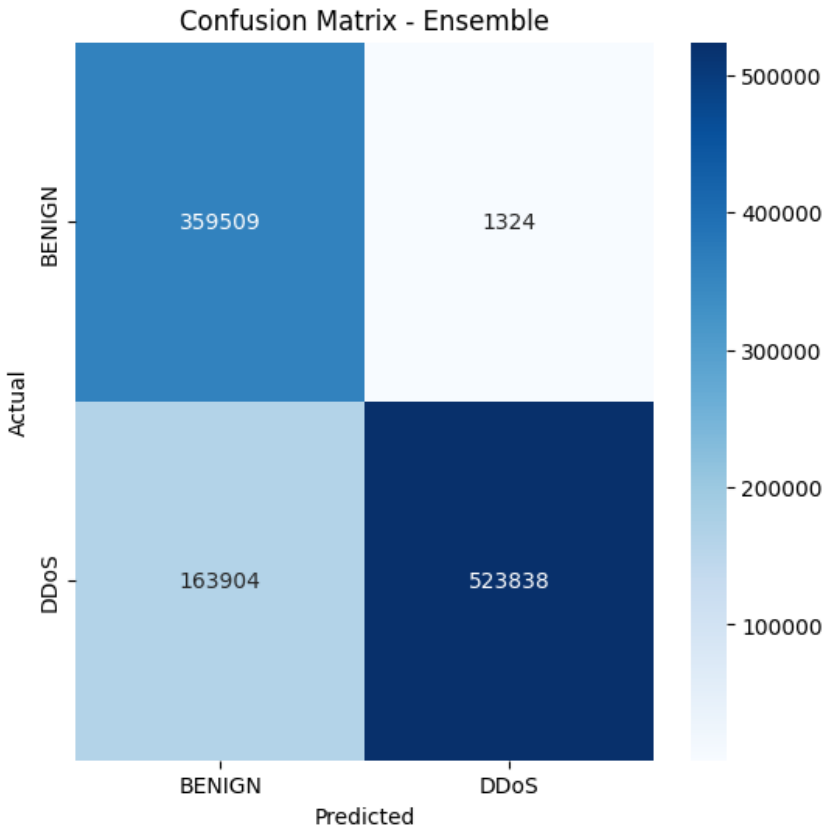


Figure 4.6: Ensemble's Confusion Matrix

4.3 Model Performance Comparison

This section presents a comparative analysis of the trained models Random Forest, CNN, BiLSTM, and an Ensemble approach. The evaluation is conducted using two datasets: the original CICDDoS2019 dataset and a realistically enhanced version, referred to as CICDDoS-5G, derived from traffic generated by the simulated 5G network.

The performance of each model is assessed using the following metrics (see table 4.2):

- **Accuracy**: Overall detection rate.
- **False Positive Rate (FPR)**: Ratio of benign traffic misclassified as attacks.
- **False Negative Rate (FNR)**: Ratio of attack traffic misclassified as benign.
- **Execution Time**: Time required to train and evaluate the model.
- **Precision**, **Recall**, and **F1-score**: Represented graphically in histograms.

Figures 4.7 and 4.8 illustrate the precision, recall, and F1-score of the models on both datasets.

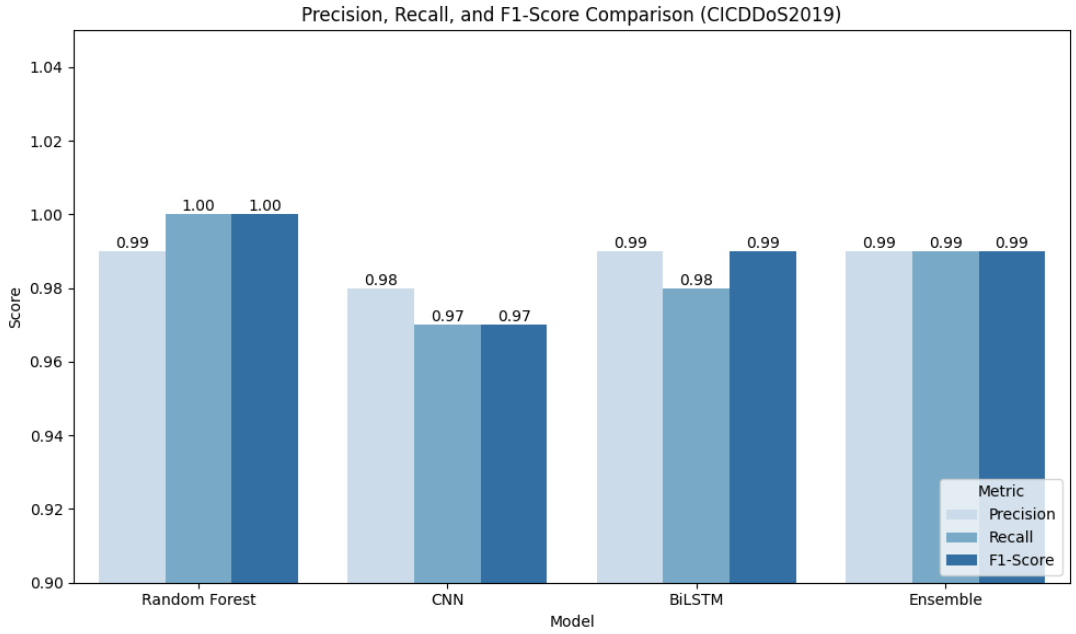


Figure 4.7: Precision, Recall, and F1-Score for Models on CICDDoS2019

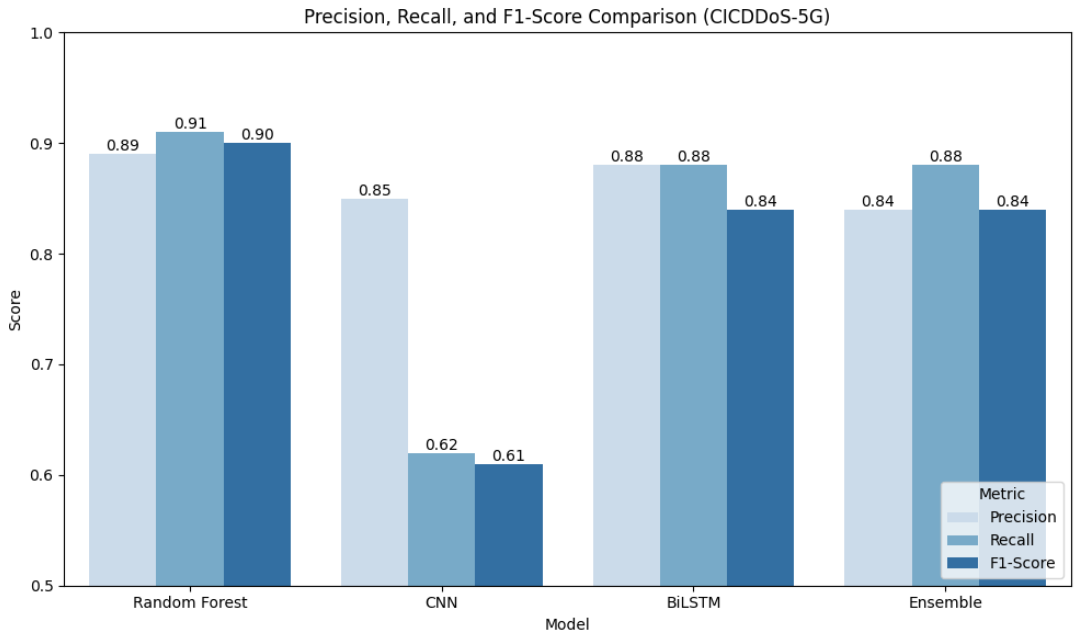


Figure 4.8: Precision, Recall, and F1-Score for Models on CICDDoS-5G

4.3.1 Interpretation and Discussion

- Random Forest consistently delivers the highest performance in terms of accuracy, F1-score, and low error rates. It also exhibits the fastest execution time, making it suitable for real time applications.

- CNN performs well on the original dataset but suffers a notable performance drop on the enhanced dataset CICDDoS-5G, indicating a limited ability to generalize temporal traffic variations.

- BiLSTM provides strong results on both datasets, benefiting from its ability to model sequential data. However, it demands higher computational resources and longer execution time.

- The Ensemble approach combines the strengths of individual models and achieves balanced performance but at the cost of increased complexity and slower execution.

Real Time Detection Performance

In high throughput 5G environments, Random Forest stands out due to its low latency and efficient processing. While CNN and BiLSTM are powerful in pattern recognition, their heavier computational requirements make them less suitable for ultra low latency scenarios unless hardware acceleration is employed.

Preprocessing Complexity

Random Forest requires minimal feature preprocessing. In contrast, CNN needs feature reshaping, and BiLSTM demands careful handling of temporal sequences, making them more complex to deploy.

Table 4.2: Performance Comparison of ML/DL Models for DDoS Detection in 5G Networks

Model	**Dataset**	**Accuracy (%)**	**FPR**	**FNR**	**Execution Time**
Random Forest	CICDDoS2019	99.90	0	0.001	Fast
	CICDDoS-5G	90.38	0.0741	0.1081	
CNN	CICDDoS2019	98.60	0.0428	0.0360	Moderate
	CICDDoS-5G	73.92	0.6433	0.1425	
BiLSTM	CICDDoS2019	98.83	0.0084	0.0134	Slow
	CICDDoS-5G	84.18	0.0001	0.2409	
Ensemble	CICDDoS2019	99.00	–	–	Slowest
	CICDDoS-5G	84.24	0.0037	0.2383	

Overall, the results highlight the trade offs between accuracy, execution time, and model complexity. Based on these considerations and the need for efficient real time detection in 5G networks, Random Forest was selected for this application due to its high accuracy, low false positive rate, and fast execution time.

4.4 Defense Framework Results

To validate our Defense framework, we designed a realistic test scenario where a compromised UE executes malicious traffic patterns. The system's response was evaluated across three key components: the external IDS for detection, the xApp for decision-making, and the gNB for proper mitigation execution.

4.4.1 Real time Detection Results

Our detection system was evaluated in a simulated 5G network scenario, where two UEs from the same network slice were deployed: one generating normal traffic through ICMP (ping), and the other simulating a DDoS attack using the Hping tool targeting the UPF, as shown in Figure 4.9.

```
linux@linux:~$ sudo ip netns exec ue1 bash
[sudo] password for linux:
root@linux:/home/linux# sudo ip route add 192.168.70.142 dev oaitun_ue1
root@linux:/home/linux# hping3 -S -d 100000 192.168.70.142 --flood
HPING 192.168.70.142 (oaitun_ue1 192.168.70.142): S set, 40 headers + 34464 data bytes
hping in flood mode, no replies will be shown
```

(a) DDoS attack traffic generated by UE1

```
linux@linux:~$ sudo ip netns exec ue3 bash
[sudo] password for linux:
root@linux:/home/linux# ping -I oaitun_ue1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.0.0.3 oaitun_ue1: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=108 time=114 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=620 ms
```

(b) Normal ICMP traffic generated by UE2

Figure 4.9: Traffic patterns from UEs connected to the same slice

The IDS consistently demonstrated strong detection capabilities. When the malicious UE (10.0.0.2) initiated the DDoS attack, the system accurately identified the anomalous behavior with a 100% true positive rate across all test runs, as shown in Figure 4.10. The detection process exhibited an average latency of 1.2 seconds from attack initiation, indicating near real time response capabilities. Additionally, the IDS effectively leveraged flow level features to establish a strong correlation between traffic characteristics and anomaly scores, confirming its robustness in distinguishing between benign and malicious flows. These results validate the system's ability to detect and respond to DDoS attacks in 5G environments.

```
linux@linux:~$ sudo docker exec -ti oai-upf-slice1 bash
[sudo] password for linux:
root@81513d3da96f:/app# python3 IDS.py
2025-05-31 20:01:52,149 - Connected to the server.
2025-05-31 20:05:54,334 - Predicting for UE IP: 10.0.0.3...
2025-05-31 20:05:54,729 - Normal: 4, DDoS: 0, UE: 10.0.0.3
2025-05-31 20:06:12,174 - Predicting for UE IP: 10.0.0.2...
2025-05-31 20:06:13,431 - Normal: 0, DDoS: 4, UE: 10.0.0.2
```

Figure 4.10: IDS Detection results.

```
Waiting for incoming connections...
Connection accepted from IDS 1
Connection accepted from IDS 2
Both IDS connected. Starting main loop...
[xApp]: CONTROL-REQUEST tx
[xApp]: CONTROL ACK rx
[xApp]: Successfully received CONTROL-ACK
RC initialization completed. Starting main loop...
Received message from IDS 1: sst:1,sd:1,normal:4,anomaly:0,ue_ip:10.0.0.3
Parsed values for IDS 1: SST: 1, SD: 1, Normal: 4, Anomaly: 0, IP: 10.0.0.3, RRC UE ID: 2
IDS 1: SST: 1, SD: 1, Normal: 4, Anomaly: 0
IDS 2: SST: 1, SD: 5, Normal: 0, Anomaly: 0
[xApp]: CONTROL-REQUEST tx
[xApp]: CONTROL ACK rx
[xApp]: Successfully received CONTROL-ACK
Received message from IDS 1: sst:1,sd:1,normal:0,anomaly:4,ue_ip:10.0.0.2
Parsed values for IDS 1: SST: 1, SD: 1, Normal: 0, Anomaly: 4, IP: 10.0.0.2, RRC UE ID: 1
IDS 1: SST: 1, SD: 1, Normal: 0, Anomaly: 4
IDS 2: SST: 1, SD: 5, Normal: 0, Anomaly: 0
IDS 1 (RAN UE ID: 1) has 100% anomaly. Triggering RRC release.
[xApp]: CONTROL-REQUEST tx
[xApp]: CONTROL ACK rx
[xApp]: Successfully received CONTROL-ACK
RRC Release triggered for UE 1
```

Figure 4.11: xApp Action Upon Attack Detection: RRC Release for Target UE.

```
[E2-AGENT]: CONTROL ACKNOWLEDGE tx
[TELNETSRV] Telnet client connected....
[NR_RRC]   [FRAME 00000][gNB][MOD 00][RNTI 1] Logical Channel DL-DCCH, Generate RRCRelease (bytes 3)
[E1AP]   releasing UE 1
[GTPU]   [101] Deleted all tunnels for ue id 1 (1 tunnels deleted)
[SDAP]   Successfully deleted SDAP entity for UE 1
[RRC]   UE 1: received bearer release complete
```

Figure 4.12: gNB logs confirming RRC Release execution for UE1

4.4.2 Real time Mitigation Results

The Figure 4.11 illustrate a successful mitigation process triggered by the xApp in response to detected anomalies. The xApp, after registering one E2 node and establishing connections with two IDS, began monitoring network traffic. IDS 1 reported significant anomalies, with four anomaly flows and zero normal activities from UE1 with IP address 10.0.0.2 while UE2 with IP address 10.0.0.3 has zero anomaly flows and four normal flows, IDS 2 showed no traffic. The xApp determined that IDS1's data indicate a 100% anomaly rate for the UE associated with RAN UE ID 1, prompting immediate action.

gNB reaction

Upon identifying the threat, the xApp initiated a control request to release the RRC connection for the affected UE. The gNB received the command and executed the RRC release procedure, as evidenced by the logs in figure 4.12. The gNB generated an RRC Release message, released the UE's resources, and deleted all associated GTP-U tunnels. The UE confirmed the release by sending a bearer release complete message, concluding the mitigation process. This sequence demonstrates the xApp's ability to autonomously detect anomalies and enforce countermeasures to isolate potentially compromised devices, thereby maintaining network integrity.

These results confirm that the proposed system not only detects threats rapidly but also reacts in a timely and coordinated manner to mitigate potential harm to the 5G

infrastructure.

4.5 Evaluation of the Defense Framework

To evaluate the effectiveness of the defense framework, we designed two distinct scenarios. The first scenario serves as a baseline in which no defense mechanisms are applied, leaving the network fully exposed to DDoS attacks. The second scenario integrates our proposed defense framework. By systematically comparing the performance of both scenarios across key metrics such as Round Trip Time and Throughput, we aim to demonstrate the obvious benefits of our solution in maintaining optimal network performance, even under attack conditions.

4.5.1 Round Trip Time

Round Trip Time (RTT) is a key metric for evaluating network performance, representing the delay between sending a packet and receiving its response. It is particularly critical in 5G networks, where ultra low latency is a fundamental requirement. To assess the effectiveness of the framework, we conducted two experiments within the 5G simulated network setup.

Interpretation

As shown in Figure 4.13, there is a clear performance gap between the two scenarios. In the first scenario, where no defense mechanisms were enabled, RTT increases significantly, with spikes reaching up to 4000 ms. This renders the network nearly unusable under attack conditions and highlights the severe impact of DDoS attacks on service availability.

In contrast, when the defense framework is activated, RTT remains stable and consistently below 100 ms even during the attack phase. This represents an improvement of over 97% in latency performance. The network remains responsive and maintains a good quality of service for legitimate users, showcasing the effectiveness of our defense system in preserving low latency requirements under adverse conditions.

Discussion

The two experiments were conducted as follows:

- **Without defense framework:** Two UEs were deployed, each belonging to a different network slice. UE1 from the first slice generated normal ICMP (ping) traffic, maintaining very low RTT initially. However, once UE2 from the second slice launched a flood based DDoS attack using the `hping` tool, the RTT for UE1 increased dramatically. This indicates a lack of slice isolation at the resource allocation level, as an attack targeting one slice severely affected the performance of another.

- **With defense framework:** In this scenario, the proposed mitigation system was enabled. Upon detecting the malicious traffic generated by UE2, the attacker was immediately disconnected from the network. As a result, UE1 continued to experience smooth and stable latency performance, as evidenced in Figure 4.13. This

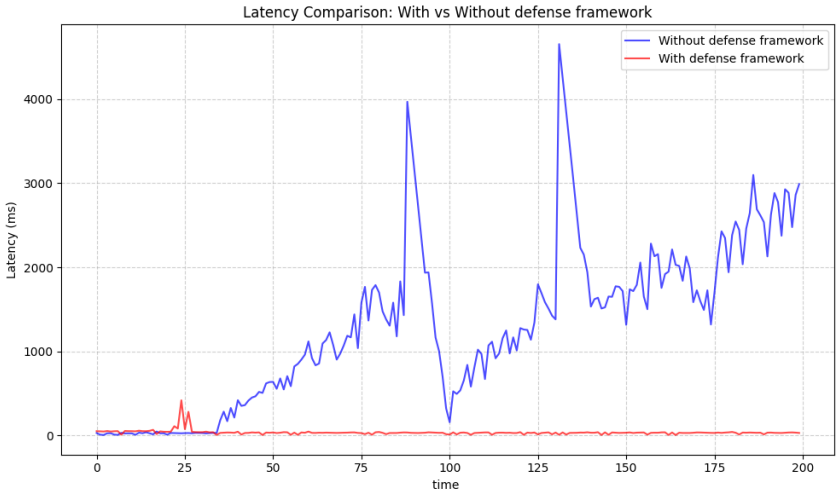


Figure 4.13: RTT comparison.

confirms the real time responsiveness and efficacy of our defense mechanism in mitigating DDoS attacks while ensuring continued quality of service.

These results emphasize the necessity of integrated defense mechanisms in modern 5G networks. Our framework not only detects and mitigates attacks effectively but also enforces strong isolation between network slices, ensuring a reliable and secure environment for all users.

4.5.2 Throughput

Throughput is one of the key measures of how well a 5G network performs. It tells us how much data is successfully sent from a user device(UE) to the network. For applications that need high speed connections like video streaming or critical real time services, maintaining fast and stable throughput is essential. To understand how well our defense system works under attack, we ran two test scenarios in our simulated 5G environment: one without any protection, and one with our mitigation framework turned on.

Interpretation

As illustrated in Figure 4.14, the impact of a DDoS attack on the uplink throughput of a normal UE is starkly different between the two scenarios.

In the first case, without any mitigation, the UE initially maintains a high throughput of approximately 24 Mbps. However, after the attack begins at the 18th second, there is a dramatic and sustained drop in throughput to around 11 Mbps. This clearly indicates the network's vulnerability to resource exhaustion caused by malicious traffic, which compromises performance for legitimate users.

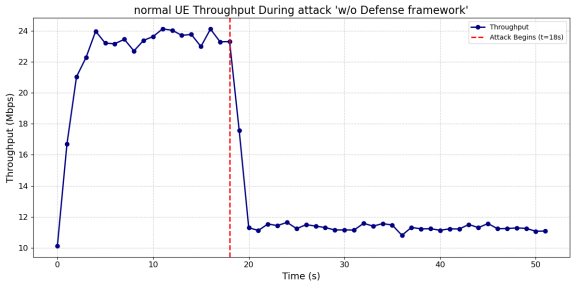
In contrast, when the defense mechanism is active, the throughput briefly dips after the attack started but quickly stabilizes in the range of 19–22 Mbps. This indicates that the framework successfully detects and suppresses the impact of the attack, maintaining network performance for non-malicious users.

Discussion

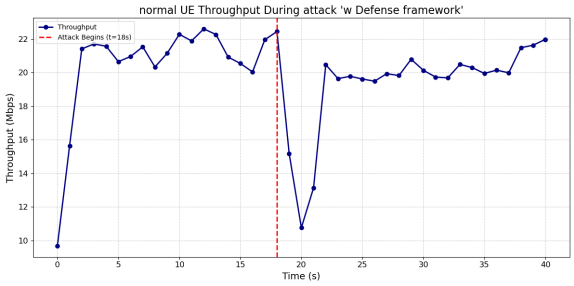
The experiment was structured as follows:

- **Without defense framework:** Two UEs were connected to the 5G core. UE1 transmitted traffic normally, while UE2 initiated a flood-based DDoS attack. The network failed to differentiate between normal and malicious flows, leading to a sharp drop in UE1's throughput due to resource contention caused by the attack.
- **With defense framework:** The defense mechanism monitored traffic patterns and identified the abnormal flow from UE2. Upon detection, the attacker was promptly removed from the network. As a result, UE1 maintained high throughput throughout the experiment, showing the resilience of the network under attack conditions.

These findings underscore the importance of intelligent, real time mitigation systems in 5G architectures. Our proposed framework not only identifies and neutralizes threats effectively but also ensures that legitimate users continue to experience high throughput, fulfilling one of the core promises of 5G networks.



(a) Normal UE throughput **without** defense framework



(b) Normal UE throughput **with** defense framework

Figure 4.14: Throughput comparison of normal UE during DDoS attack with and without defense mechanism

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis presents a meaningful advancement to the field of cybersecurity in 5G Networks through the design and evaluation of a complete machine learning based defense system capable of detecting and mitigating DDoS attacks in real time. Central to our approach was the creation of a realistic and modular 5G simulation environment built upon OpenAirInterface and enhanced with the FlexRIC controller to support O-RAN architectural principles. This testbed played a vital role in emulating complex, real world network behaviors and enabled the implementation of custom xApps within a near-RT RIC context.

Furthermore, our work integrates advanced machine learning models trained on an enhanced dataset, which combines the CICDDoS2019 benchmark dataset with real traffic generated from our simulated O-RAN based 5G network. This approach ensures high adaptability and performance under diverse network conditions. The result is a robust and scalable security solution that not only detects malicious behavior with high accuracy but also actively mitigates it by disconnecting compromised UEs through targeted RRC Release procedures. To preserve service quality, the system maintains balanced resource distribution across network slices following mitigation actions. This research demonstrates both the feasibility and the necessity of embedding intelligent security mechanisms directly into the fabric of next generation mobile networks.

5.1.1 Achievements of the Research

- **Simulation of a 5G Network Integrated with Open RAN Architecture:** A fully functional Simulated 5G Network was developed using OpenAirInterface (OAI) to simulate realistic 5G network scenarios. We extended this environment by integrating the FlexRIC near-RT RIC controller, thereby enabling support for the O-RAN architecture and laying the foundation for implementing intelligent xApps in a controlled setup.

- **Construction of the CICDDoS-5G Enhanced Dataset:**

To ensure model relevance to real world 5G environments, we constructed an enhanced dataset named **CICDDoS-5G**, which combines the CICDDoS2019 benchmark dataset with real traffic generated from our OAI-based O-RAN simulated network. This involved capturing both benign and malicious traffic in the simulated environment and carefully merging it with the existing traffic. The resulting dataset provides a richer and more representative training resource, enabling the machine learning models to generalize better and perform effectively in diverse and dynamic 5G scenarios.

- **Development and Training of Robust ML-Based Detection Models:**

  We built and trained machine learning models using the enhanced **CICDDoS-5G** dataset. The dataset was thoroughly preprocessed to extract relevant features and ensure model generalization. The developed models include Random Forest, Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory networks (BiLSTM), and an ensemble model that combines the strengths of all three, with Random Forest contributing the most due to its consistently high performance, achieving an accuracy of 90.38% across evaluation metrics.

- **Comprehensive Model Performance Evaluation:**

  The trained models were evaluated on both the original CICDDoS2019 and the enhanced CICDDoS-5G datasets. Key performance metrics such as accuracy, precision, recall, and F1-score were used to assess model effectiveness. Additionally, confusion matrices were generated to visualize classification results and analyze misclassification patterns. This dual evaluation confirmed the enhanced dataset's contribution to better generalization and higher performance, particularly in more realistic 5G traffic conditions.

- **Design of a Real Time Detection Framework:** A lightweight detection system was implemented using the Scapy library to capture live traffic, extract features on the fly, and classify incoming flows using our trained models. Instead of evaluating packets individually, the system processes groups of packets as a flow, enabling more accurate context aware predictions. The detection output is integrated into the O-RAN environment by forwarding alerts to the xApp running on the near-RT RIC, enabling coordinated mitigation.

- **Implementation of a Real Time Mitigation xApp:** We developed an xApp capable of listening to multiple Intrusion Detection Systems (IDS) simultaneously and taking real time mitigation actions. Based on the anomaly score of each UE, the xApp could issue RRC release commands to disconnect malicious UEs. Additionally, the xApp implemented an equal PRB reallocation strategy to maintain fairness and service continuity across slices. This ensured that the impact of an attack was isolated and did not degrade the experience of other legitimate UEs, whether on the same or different network slices.

- **Evaluation of the Defense Framework:** The effectiveness of the overall detection and mitigation frameworks were evaluated using key performance indicators such as Round Trip Time (RTT) and throughput. Measurements before, during, and after DDoS attack scenarios showed that the proposed system effectively reduced latency spikes and restored throughput levels after mitigation actions. This

confirms the framework's ability to maintain service quality and network responsiveness in the presence of attacks.

5.2 Future Work

To further enhance the effectiveness and applicability of our proposed defense system, several directions will be pursued in future work:

- **Multi class Classification:** Extend the current binary classification approach to a multi class classification framework. This will enable the system to accurately identify and differentiate between multiple types of DDoS attacks, thereby improving detection precision and response strategies.
- **Utilization of Real 5G Traffic Data:** Incorporate real world 5G network traffic data into the training and evaluation pipeline to improve model generalization, reliability, and relevance in practical deployment scenarios.
- **Dynamic PRB Allocation for Mitigation:** Investigate the implementation of dynamic PRB allocation strategies as a means to mitigate DDoS attacks. This involves adjusting resource allocation in real time to isolate or restrict malicious traffic while preserving service quality for legitimate users.
- **Dynamic Mapping of UE Identifiers for Precise Mitigation:** Enhance the mitigation process by implementing a dynamic mapping mechanism between UE IP addresses (identified by traffic analysis engines) and their corresponding `RAN_UE_ID` as known by the gNB. This mapping can be achieved by utilizing the O-RAN KPM (Key Performance Measurement) service model, which provides realt ime reporting of UE level performance metrics, including identifiers, thereby enabling precise correlation and targeted mitigation actions.
- **Adaptive and Self Learning Models:** Develop adaptive ML and DL models that dynamically adjust to changing network conditions and evolving attack patterns. Incorporating reinforcement learning techniques could enable these models to continuously learn from new data, improving detection accuracy and responsiveness over time.
- **Generalization to Diverse Network Threats:** Broaden the scope of the proposed detection framework by adapting its methodologies and models to detect a wider range of network threats, including intrusion attempts, malware propagation, and other sophisticated anomalies. Such an expansion is essential for achieving comprehensive and resilient security in increasingly complex and dynamic environments, particularly within 5G and future 6G networks.

These future directions aim to contribute to the development of a more intelligent, adaptive, and robust intrusion detection and mitigation system tailored for the dynamic environment of 5G networks.

Bibliography

[1] Cavli Wireless, "Architectural advancements in 5g technology," 2023, accessed: 2025-04-16. [Online]. Available: https://www.cavliwireless.com/blog/not-mini/architectural-advancements-in-5g-technology

[2] S. Park, B. Cho, D. Kim, and I. You, "Machine learning based signaling ddos detection system for 5g stand alone core network," *Applied Sciences*, vol. 12, no. 23, p. 12456, 2022, accessed: 2025-04-16. [Online]. Available: https://www.mdpi.com/2076-3417/12/23/12456

[3] O-RAN Alliance. (2022) O-ran alliance introduces 53 new specifications released since july 2022. Accessed: 2025-05-07. [Online]. Available: https://www.o-ran.org/blog/o-ran-alliance-introduces-53-new-specifications-released-since-july-2022

[4] OpenAirInterface, "Openairinterface: An open-source 5g wireless software platform," https://openairinterface.org/, 2025, accessed: 2025-05-09.

[5] MOSAIC5G, "FlexRIC: Open RAN near-RT RIC platform," https://gitlab.eurecom.fr/mosaic5g/flexric, 2025, accessed: 2025-05-09.

[6] A. Elbarbary, P. Bertin, L. Bertaux, Y. Meidan, and P. Owezarski, "Open ran security: Challenges and opportunities," *Journal of Network and Computer Applications*, vol. 210, p. 103563, 2023, accessed: Jun. 12, 2025. [Online]. Available: https://doi.org/10.1016/j.jnca.2022.103563

[7] 3GPP, "System architecture for the 5G system (release 15)," 3GPP, Technical Specification TS 23.501, 2018.

[8] Calsoft Inc., "5G service-based architecture (SBA) explained," 2023, accessed: 2025-04-16. [Online]. Available: https://www.calsoftinc.com/blogs/5g-service-based-architecture-sba.html

[9] Y. Imam-Fulani, N. Faruk, O. Sowande, A. Abdulkarim, E. Alozie, A. Usman, K. Adewole, A. Oloyede, H. Chiroma, S. Garba, A. Imoize, B. Baba, A. Musa, Y. Adediran, and L. Taura, "5g frequency standardization, technologies, channel models, and network deployment: Advances, challenges, and future directions," *Sustainability*, vol. 15, no. 6, p. 5173, 2023, accessed: 2025-04-16. [Online]. Available: https://www.mdpi.com/2071-1050/15/6/5173

[10] A. I. Grohmann, M. Seidel, S. A. W. Itting, R.-G. Cheng, M. Reisslein, and F. H. P. Fitzek, "Multi-ue 5g ran measurements: A gamut of architectural options," *IEEE Access*, vol. 13, 2025, accessed: 2025-04-16. [Online]. Available: https://ieeexplore.ieee.org/document/10816630

[11] K. Alam, M. A. Habibi, M. Tammen, D. Krummacker, W. Saad, M. Di Renzo, T. Melodia, X. Costa-Pérez, M. Debbah, A. Dutta, and H. D. Schotten, "A comprehensive overview and survey of o-ran: Exploring slicing-aware architecture, deployment options, and use cases," *Submitted to IEEE for possible publication*, 2024, accessed: 2025-05-07. [Online]. Available: https://arxiv.org/abs/2405.03555

[12] M. S. Khan, "Detection of dos and ddos attacks on 5g network slices using deep learning approach," Master of Science Thesis, University of Regina, Regina, Saskatchewan, 2023, accessed: 2025-04-16.

[13] Digi International, "What is 5g network architecture?" 2023, accessed: 2025-04-16. [Online]. Available: https://www.digi.com/blog/post/5g-network-architecture

[14] D. Bloom, R. Filkovsky, and B. Lifshitz, "The top 4 ddos attack vectors threatening 5g networks," https://www.allot.com/blog/top-ddos-attack-vectors-threatening-5g/, 2022, accessed: 2025-04-16.

[15] J. Zhou, M. Abolhasan, B. Javadi, and J. Lipman, "5gdad: A deep learning approach for ddos attack detection in 5g p4-based upf," *Computer Networks*, vol. 237, p. 110005, 2023, accessed: 2025-04-16.

[16] H. Wen, P. Porras, V. Yegneswaran, and Z. Lin, "A fine-grained telemetry stream for security services in 5g open radio access networks," in *Proceedings of the 2022 ACM Workshop on Emerging Topics in Wireless (EmergingWireless '22)*. Roma, Italy: ACM, December 2022, p. 6. [Online]. Available: https://doi.org/10.1145/3565474.3569070

[17] Cloudflare, "What is a ddos attack?" 2023, accessed: 2025-04-17. [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[18] Y. Huang, Y. Bai, J. Zhang, Y. Li, and Y. Feng, "Trend analysis and countermeasure research of ddos attack under 5g network," *IEEE Access*, vol. 9, pp. 137 587–137 600, 2021, accessed: 2025-04-16. [Online]. Available: https://ieeexplore.ieee.org/document/9560039

[19] Radware, "Dos vs. ddos attack: What is the difference?" 2023, accessed: 2025-04-17. [Online]. Available: https://fr.radware.com/cyberpedia/ddos-attacks/dos-vs-ddos-attack-what-is-the-difference/

[20] INSA TC, "Tout ce que vous avez toujours voulu savoir sur les attaques ddos," 2023, accessed: 2025-04-17. [Online]. Available: https://medium.com/insa-tc/tout-ce-que-vous-avez-toujours-voulu-savoir-sur-les-attaques-ddos-65beed2b96a

[21] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," in *ACM Computing Surveys*, vol. 39, no. 1, 2007, pp. 1–42.

[22] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03)*, 2003, pp. 303–314.

[23] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[24] M. Dimolianis, A. Pavlidis, and V. Maglaris, "Signature-based traffic classification and mitigation for ddos attacks using programmable network data planes," *IEEE Access*, vol. 9, pp. 113 061–113 075, 2021.

[25] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. M. del Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," pp. 876–889, 2020.

[26] IBM, "Machine learning," https://www.ibm.com/think/topics/machine-learning, 2025.

[27] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, online: https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.

[28] M. M. Noel, S. Bharadwaj, V. Muthiah-Nakarajan, P. Dutta, and G. B. D. Amali, "Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons," *Expert Systems with Applications*, vol. 266, 2025.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, online: http://www.deeplearningbook.org.

[30] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] B. S. Reddy, "Advancing ddos detection in 5g networks through machine learning and deep learning techniques," Master's thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 2024.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, 2017.

[33] C. Zeng, Z. Wang, and Z. Wang, "Image reconstruction of iot based on parallel cnn," in *Proc. 2020 Int. Conf. on Internet of Things (iThings)*, 2020, accessed: 2025-04-16.

[34] K. S. Varshitha, C. G. Kumari, M. Hasvitha, S. Fiza, A. K., and V. Rachapudi, "Natural language processing using convolutional neural network," in *Proc. 2023 7th Int. Conf. on Computing Methodologies and Communication (ICCMC)*, 2023, accessed: 2025-04-16.

[35] R. Alguliyev and R. Shikhaliyev, "Computer networks cybersecurity monitoring based on cnn-lstm model," in *Proc. 2024 IEEE 18th Int. Conf. on Application of Information and Communication Technologies (AICT)*, 2024, accessed: 2025-04-16.

[36] R. A. Bakar, F. Alhamed, P. Castoldi, A. Sgambelluri, J. J. V. Olmos, F. Cugini, and F. Paolucci, "5gdad: A deep learning approach for ddos attack detection in 5g p4-based upf," *2024 IEEE 25th Int. Conf. on High Performance Switching and Routing (HPSR)*, 2024.

[37] Z. Gao, "5g traffic prediction based on deep learning," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–5, 2022, online: https://www.hindawi.com/journals/cin/2022/3174530/.

[38] UpGrad Blog, "Basic cnn architecture," n.d., accessed: 2025-04-17. [Online]. Available: https://www.upgrad.com/blog/basic-cnn-architecture/

[39] M. M. Taye, "Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions," *Computers*, vol. 12, no. 6, p. 91, 2023.

[40] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[41] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[42] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[43] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, 2016, pp. 1–5.

[44] ResearchGate, "BAT: Deep Learning Methods on Network Intrusion Detection using NSL-KDD dataset - Scientific Figure," https://www.researchgate.net/figure/The-architecture-of-BLSTM-model_fig2_339174926, n.d., accessed: 2025-04-18.

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

[46] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[47] C. Stryker. (2024, October) What is a recurrent neural network (rnn)? Accessed: 2025-05-11. [Online]. Available: https://www.ibm.com/think/topics/recurrent-neural-networks

[48] IBM Corporation, "Loss function – ibm cloud learn hub," https://www.ibm.com/think/topics/loss-function, 2024, accessed: May 2025.

[49] IBM. (2020) Overfitting in machine learning. Accessed: 2025-05-11. [Online]. Available: https://www.ibm.com/think/topics/overfitting

[50] ——. (2020) Regularization in machine learning. Accessed: 2025-05-11. [Online]. Available: https://www.ibm.com/think/topics/regularization

[51] N. Team, "Performance metrics in machine learning: Complete guide," https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide, 2022, accessed: 2025-04-18.

[52] S. Raschka, "Confusion matrix — mlxtend," https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/, 2024, accessed: 2025-04-18.

[53] B. Bousalem, M. A. Sakka, V. Silva, W. Jaafar, A. Ben Letaifa, and R. Langar, "DDoS Attacks Mitigation in 5G-V2X Networks: A Reinforcement Learning-Based Approach," in *2023 19th International Conference on Network and Service Management (CNSM)*, ser. 2023 19th International Conference on Network and Service Management (CNSM). Niagara Falls, Canada: IEEE, Oct. 2023, accessed: 2025-04-18. [Online]. Available: https://hal.science/hal-04492996

[54] S. Sheikhi and P. Kostakos, "Ddos attack detection using unsupervised federated learning for 5g networks and beyond," University of Oulu, Faculty of Information Technology and Electrical Engineering, Oulu, Finland, Master's Thesis, 2024, accessed: 2025-04-18.

[55] A. Saini and A. Arora, "Detection of ddos attacks using machine learning algorithms," in *7th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2020, pp. 16 978–16 983, accessed: 2025-04-18.

[56] B. S. Reddy, "Advancing ddos detection in 5g networks through machine learning and deep learning techniques," Master's thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 2024.

[57] P. C. A. S. J. J. V. O. F. C. R. Abu Bakar, F. Alhamed and F. Paolucci, "5gdad: A deep learning approach for ddos attack detection in 5g p4-based upf," in *Proc. 2024 IEEE 25th Int. Conf. on High Performance Switching and Routing (HPSR)*. Pisa, Italy: IEEE, Jun. 2024, pp. 1–6.

[58] A.-A. Maiga, E. Ataro, and S. Githinji, "Xgboost and deep learning based-federated learning for ddos attack detection in 5g core network vnfs," in *2024 6th International Conference on Computer Communication and the Internet (ICCCI)*. Tokyo, Japan: IEEE, Jun. 2024, pp. 1–6.

[59] S. K. M. Sheibani and I. Awan, "Ddos attack detection and mitigation in software-defined networking-based 5g mobile networks with multiple controllers," in *2022 9th International Conference on Future Internet of Things and Cloud (FiCloud)*. Rome, Italy: IEEE, 2022, pp. 32–39.

[60] B. Bousalem, V. F. Silva, R. Langar, and S. Cherrier, "Ddos attacks detection and mitigation in 5g and beyond networks: A deep learning-based approach," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Rio de Janeiro, Brazil, Dec. 2022, pp. 1259–1264.

[61] H. Wen, P. Porras, V. Yegneswaran, A. Gehani, and Z. Lin, "5g-spector: An o-ran compliant layer-3 cellular attack detection service," in *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS'24)*. San Diego, CA: Internet Society, February 2024. [Online]. Available: https://dx.doi.org/10.14722/ndss.2024.24527

[62] M. Awad, A. A. Hamid, Y. Ranganathan, N. Choubik, R. Langar, and W. Jaafar, "xapps for ddos attacks detection and mitigation in 5g-v2x o-ran networks," in

Proceedings of the 2024 7th Conference on Cloud and Internet of Things (CIoT). Paris, France: IEEE, June 2024, pp. 97–104, accessed: 2025-06-12. [Online]. Available: https://doi.org/10.1109/CIoT63799.2024.10757133

[63] OpenAirInterface, "Nr sa tutorial - oai 5g core," 2025, accessed: 2025-05-09. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/NR_SA_Tutorial_OAI_CN5G.md

[64] Docker, Inc., "What is docker?" https://docs.docker.com/get-started/docker-overview/, 2025, accessed: 2025-05-09.

[65] ——, "Docker compose overview," https://docs.docker.com/compose/, 2025, accessed: 2025-05-09.

[66] M. Elsayed, N.-A. Le-Khac, S. Dev, and A. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Jun. 2020, pp. 367–372.

[67] Google. (2023) Tensorflow: An end-to-end open-source machine learning platform. Accessed: 2025-05-11. [Online]. Available: https://www.tensorflow.org

[68] Keras Team. (2023) Keras: Deep learning for humans. Accessed: 2025-05-11. [Online]. Available: https://keras.io

[69] The Pandas Development Team. (2023) Pandas: Python data analysis library. Accessed: 2025-05-11. [Online]. Available: https://pandas.pydata.org

[70] NumPy Developers. (2023) Numpy: The fundamental package for scientific computing with python. Accessed: 2025-05-11. [Online]. Available: https://numpy.org

[71] Scikit-learn Developers. (2023) Scikit-learn: Machine learning in python. Accessed: 2025-05-11. [Online]. Available: https://scikit-learn.org

[72] Matplotlib Development Team. (2023) Matplotlib: Visualization with python. Accessed: 2025-05-11. [Online]. Available: https://matplotlib.org

Appendix A

Software Tools

A.1 Preprocessing and Training Tools

- **Python:** Python version 3.10 served as the foundational basis for our project development. It is a high-level, interpreted programming language widely recognized for its readability, versatility, and robust open-source community. Python supports multiple programming paradigms—including object-oriented, and functional programming—making it an ideal choice for scientific computing and machine learning. Its extensive ecosystem of libraries and strong integration capabilities facilitate the seamless development of end-to-end data analysis pipelines. [**?**].

- **TensorFlow:** TensorFlow is a scalable open-source framework developed by Google for large-scale machine learning and deep learning tasks. In this work, TensorFlow was essential for designing and training deep learning models such as CNN and BiLSTM. Its support for GPU acceleration significantly reduced training time, while its flexible computational graph architecture allowed for efficient experimentation and model tuning [67].

- **Keras:** Keras, a high-level API built on top of TensorFlow, played a key role in rapidly developing and fine-tuning deep learning models. Its intuitive syntax and modular design simplified the construction of complex neural networks, accelerating prototyping and experimentation with architectures suitable for classifying DDoS traffic patterns [68].

- **Pandas:** Pandas was used extensively for preprocessing the CICDDoS2019 dataset. Its powerful DataFrame structures facilitated tasks such as handling missing values, merging data, transforming categorical variables, and summarizing network traffic statistics. These preprocessing steps were critical for cleaning and preparing the dataset prior to model training [69].

- **NumPy:** NumPy provided the numerical backbone for many preprocessing operations. Its efficient manipulation of multi-dimensional arrays enabled fast computation of statistical features and transformations applied to large-scale traffic data. NumPy also served as a fundamental dependency for Pandas and Scikit-learn, supporting matrix operations and numerical stability during training [70].

- **Scikit-Learn:** Scikit-learn was vital for implementing the machine learning pipeline, particularly for the Random Forest model. It also offered a rich set of utilities for

feature scaling, encoding, dimensionality reduction, and model evaluation. These capabilities ensured consistent preprocessing and robust validation of model performance on DDoS detection tasks [71].

- **Matplotlib:** Matplotlib was used to visualize data distributions, model performance, and evaluation metrics such as accuracy, precision, recall, and F1-score. These visualizations were instrumental in interpreting model behavior and validating the effectiveness of the detection system in both training and real-time inference scenarios [72].

- **CICFlowMeter v3:** CICFlowMeter is an open-source tool developed by the Canadian Institute for Cybersecurity to convert raw network traffic captured in PCAP format into structured bidirectional flow-based CSV files. In this work, CICFlowMeter v3 was used to transform the captured PCAP files from the 5G simulated environment (using OpenAirInterface and FlexRIC) into flow-level features compatible with ML and DL models for DDoS detection.

- **Scapy:** Scapy is a powerful Python-based packet manipulation tool. In our setup, Scapy was used to capture live traffic and extract essential packet-level information in real time. This capability was critical for enabling on-the-fly feature extraction and feeding the real-time detection system integrated within our xApp framework.