

الجمهورية الجزائرية الديمقراطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعليم العالي و البحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC
RESEARCH
جامعة عبد الحفيظ بالصوف بميلة
UNIVERSITY OF ABDELHAFID BOUSSOUF MILA



كلية الرياضيات و الاعلام الآلي
INSTITUTE OF MATHEMATICS AND COMPUTER SCIENCE
قسم الإعلام الآلي
DEPARTMENT OF COMPUTER SCIENCE
MASTER THESIS

Field : Mathematics and Computer Science

Option : Computer Science

Specialization : Artificial Intelligence and its Applications

Submitted by : Boukheche Khawla & Bouchair Mouna

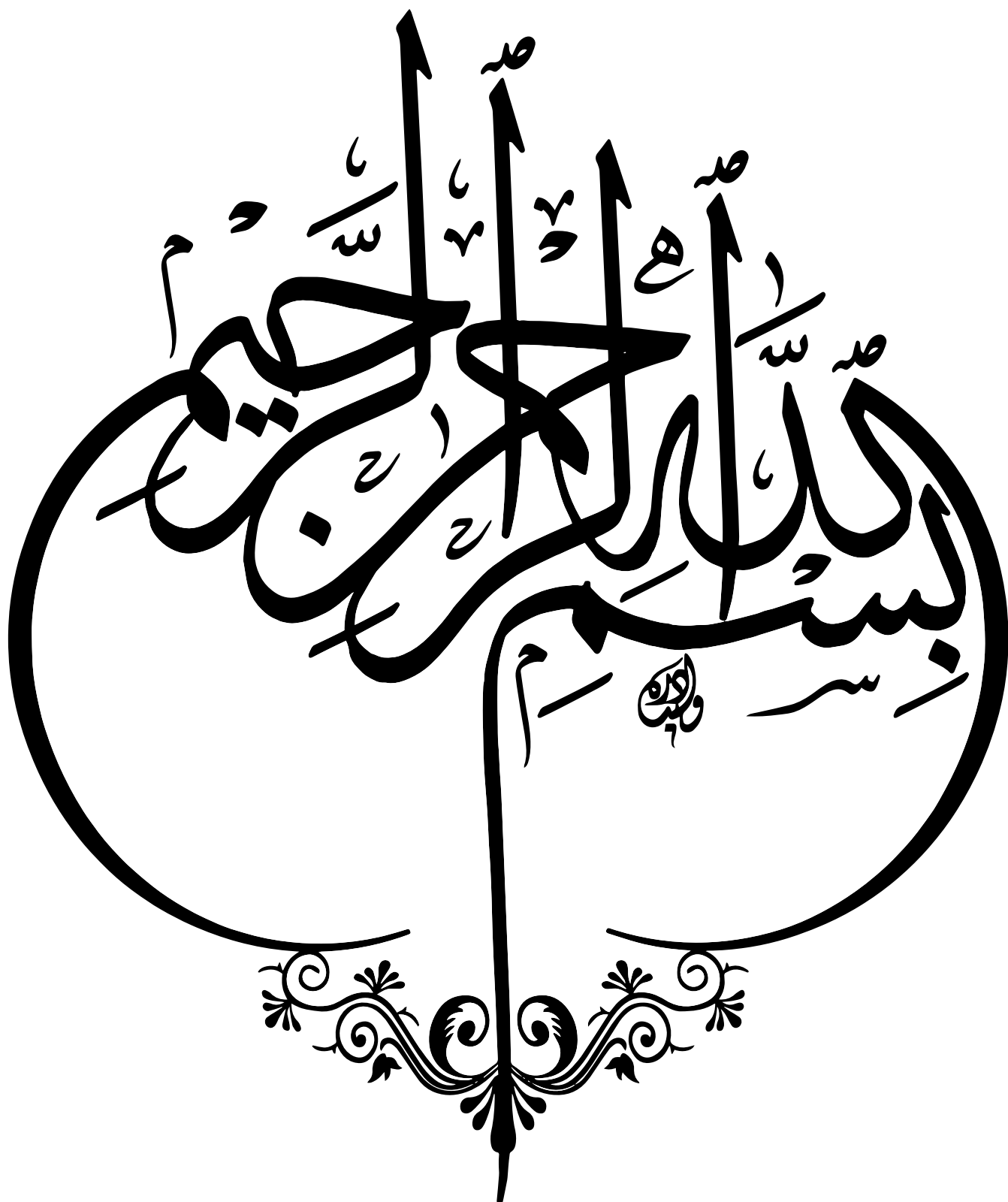
Theme

**Intelligent Intrusion Detection in IoT Networks Using
Data Mining and Machine Learning Techniques**

Jury members :

Mr. Abderrahim Dib	MAA	(University of Mila)	President
Mr. Abdelaziz Alliouche	MAA	(University of Mila)	Examiner
Mr. Ali Lalouci	MCB	(University of Mila)	Advisor

Academic Year 2025/2026



إهداء

الحمد لله أولاً وآخراً، ظاهراً وباطناً، عند البدء وعند الختام، وآخر دعواهم أن الحمد لله رب العالمين.

بعد رحلة مليئة بالتحديات والتجارب، ها أنا اليوم أضع ثمرة جهدي بين أيديكم، بكل فخر وامتنان. لم تكن الطريق سهلة، لكنها كانت مليئة بالإصرار والأمل والدعم الذي لا ينسى.

إلى أبي الغالي:

إلى سندي في الحياة، إلى من علمني معنى القوة والعمل والاجتهاد، لك مني كل الحب والاحترام والتقدير.

إلى أمي الحبيبة:

إلى منبع الحنان والدعاء، إلى قلب لا يعرف إلا العطاء، إلى من كانت سبب نجاحي بعد الله، لك مني يا أمي كل الحب الذي لا ينتهي، وكل الشكر الذي لا يكفي، وكل الامتنان الذي يعجز عن وصفه الكلام.

إلى إخوتي الأعزاء:

إلى رفقاء الدرب وسندي الدائم، الذين كانوا دائماً بجانبني في كل الظروف، حفظكم الله لي.

إلى أطفال البيت (دينا، جنى، بهاء، يوسف، تيم):

إلى زهور حياتي وبهجتها، إلى مصدر الفرح والابتسامة في البيت، لكم مني كل الحب والدعاء بمستقبل جميل ومشرق.

إلى أصدقائي وزملائي:

إلى من شاركوني لحظات الدراسة والتعب والفرح، وكانوا خير عون وسند في الطريق.

إلى نفسي:

إلى تلك الروح التي صبرت واجتهدت ولم تستسلم رغم كل الصعوبات، أقول لك: واصلي، فالقادم أجمل بإذن الله.

وفي الختام، أسأل الله أن يجعل هذا العمل خالصاً لوجهه الكريم، وأن يكتب له القبول، وأن ينفع به، وأن يوفقني لما فيه الخير والسداد في مسيرتي القادمة

منى

إهداء

بسم الله الرحمن الرحيم

الحمد لله الذي بنعمته تتم الصالحات، والحمد لله الذي وهبني من الصبر زاداً، ومن العزيمة نوراً لأبلغ هذا المسعى، وأتمّ مذكرتي هذه، فله الشناء أولاً وأخيراً.

إلى والدي العزيز:

إلى من علمني أن الخطوات الواثقة تبدأ بالعمل الدؤوب، إلى سندي الأول ومصدر قوتي الذي أفنى سنين عمره ليضيء طريقي، أهدي إليك هذا النجاح، دمت لي فخراً وعزاً أستند إليه ما حييت.

إلى أمي الغالية:

إلى نبع الحنان وملاذي الآمن، إلى من رافقتني دعواتها الصادقة في كل خطوة، وسهرت لأرتاح، وبذلت من روحها ووقتها لأبلغ هذه المرتبة، التي لولا صلواتها وتضحياتها العظيمة لما أدركت هذا الإنجاز، أطال الله في عمرك وأسعد قلبك كما أسعدت أيامي.

إلى إخوتي وأخواتي:

إلى من قاسمتهم تفاصيل الحياة، وكانوا لي العُضد والسند في لحظات الشدة قبل الرخاء، رفاق الدرب وشركاء الحلم، شكراً لقلوبكم التي كانت دائماً بالجوار.

إلى عائلتي وأقاربي:

إلى كل روح طيبة ساندتني بكلمة، أو نصيحة، أو دعوة صادقة في ظهر الغيب، وإلى كل من ترك بصمة خير في مسيرتي الأكاديمية، أهديكم ثمرة هذا الجهد.

إلى نفسي:

إلى نفسي التي جاهدت وصبرت، وتخطت لحظات الشك والتعب وعناء السهر، أهدي هذا الجهد المتواضع تقديراً لرحلة من الاجتهاد واليقين لم تكن لتكتمل لولا الإصرار.

أسأل الله العليّ القدير أن يبارك في هذا العلم، وأن يجعله نافعاً وخطوةً مباركةً نحو مستقبلٍ بفيض بالنجاح والعطاء.

خولة

Acknowledgments

First and foremost, words cannot adequately express our gratitude to **Allah**. All praise is due to Allah, the Lord of the Worlds.

We extend our deepest gratitude towards our supervisor, Mr. Ali Lalouci, for his continuous guidance, invaluable advice, and unwavering support throughout the development of this thesis. His expertise and encouragement were instrumental in completing our research.

We would also like to express our sincere appreciation to the honorable jury members for accepting to review and evaluate our work.

A special thanks goes to the teachers and faculty members of the Artificial Intelligence and Applications (I2A) track at the University Center Abdelhafid Boussouf of Mila. The knowledge and academic foundation you provided us throughout our Master's journey have been truly invaluable.

Finally, we are profoundly thankful to our families, friends, and everyone who has supported us, directly or indirectly, with motivation and prayers to bring this project to completion.

Khawla & Mouna, May 2026

Abstract

The rapid expansion of Internet of Things (IoT) technologies has introduced major cybersecurity challenges due to the increasing number of connected and resource-constrained devices vulnerable to various cyberattacks. In this context, Intrusion Detection Systems (IDS) play a fundamental role in protecting IoT environments against malicious activities and unauthorized access.

This thesis aims to design and develop an efficient IDS framework based on machine learning and deep learning techniques for IoT security. A comprehensive theoretical study was first conducted on IoT concepts, architectures, communication protocols, and security challenges, along with an overview of intrusion detection systems and the most widely used datasets in this field.

In the practical part, the CICIoT2023 dataset was utilized and processed through several preprocessing stages, including data cleaning, attack label grouping, encoding, class imbalance handling, data normalization, and feature selection. Multiple machine learning and deep learning models were then trained and evaluated using standard performance metrics such as accuracy, precision, recall, and F1-score. In addition, computational efficiency metrics including training time, inference speed, and model size were analyzed to evaluate the suitability of each model for IoT environments.

The experimental results demonstrated that tree-based machine learning models, particularly XGBoost and LightGBM, achieved superior classification performance while maintaining low computational cost compared to deep learning models. Furthermore, explainability techniques such as SHAP (SHapley Additive exPlanations) were integrated to improve model interpretability and identify the most influential network traffic features in the decision-making process.

In conclusion, this study highlights the effectiveness of machine learning approaches in improving intrusion detection systems for IoT security while emphasizing the importance of balancing detection performance, computational efficiency, and model interpretability in modern cybersecurity applications.

Keywords: Internet of Things (IoT), Intrusion Detection System (IDS), Cybersecurity, Machine Learning, Deep Learning, XGBoost, SHAP, CICIoT2023, Explainable Artificial Intelligence (XAI), Class Imbalance.

Résumé

L’expansion rapide des technologies de l’Internet des objets (IoT) a entraîné une augmentation considérable du nombre d’appareils connectés, créant ainsi d’importants défis de sécurité en raison de leur vulnérabilité face aux cyberattaques. Dans ce contexte, les systèmes de détection d’intrusion (IDS) jouent un rôle essentiel dans la protection des environnements IoT contre les activités malveillantes et les accès non autorisés.

Ce mémoire vise à concevoir et développer un système IDS efficace basé sur les techniques d’apprentissage automatique et d’apprentissage profond pour la sécurité des environnements IoT. Une étude théorique approfondie a d’abord été réalisée sur les concepts de l’IoT, ses architectures, ses protocoles de communication et ses défis de sécurité, ainsi qu’un aperçu des systèmes IDS et des jeux de données les plus utilisés dans ce domaine.

Dans la partie pratique, le jeu de données CICIoT2023 a été utilisé et traité à travers plusieurs étapes de prétraitement, le nettoyage des données, le regroupement des attaques, le codage, la gestion du déséquilibre des classes, la normalisation des données et la sélection des caractéristiques. Plusieurs modèles d’apprentissage automatique et d’apprentissage profond ont ensuite été entraînés et évalués à l’aide de métriques standards telles que l’exactitude, la précision, le rappel et la mesure F1. De plus, des métriques d’efficacité computationnelle telles que le temps d’entraînement, la vitesse d’inférence et la taille des modèles ont été analysées afin d’évaluer leur adéquation aux environnements IoT.

Les résultats expérimentaux ont montré que les modèles d’apprentissage automatique basés sur les arbres, en particulier XGBoost et LightGBM, offrent d’excellentes performances de classification tout en conservant un faible coût computationnel par rapport aux modèles d’apprentissage profond. En outre, les techniques d’explicabilité telles que SHAP (SHapley Additive exPlanations) ont été intégrées afin d’améliorer l’interprétabilité des modèles et d’identifier les caractéristiques réseau les plus influentes dans le processus de décision.

En conclusion, cette étude met en évidence l’efficacité des approches d’apprentissage automatique pour améliorer les systèmes IDS destinés aux environnements IoT, tout en soulignant l’importance d’équilibrer les performances de détection, l’efficacité computationnelle et l’interprétabilité des modèles dans les applications modernes de cybersécurité.

Mots Clés : Internet des objets (IoT), Système de détection d’intrusion (IDS), Cybersécurité, Apprentissage automatique, Apprentissage profond, XGBoost, SHAP, CICIoT2023, Intelligence artificielle explicable (XAI), Déséquilibre des classes.

ملخص

أدى التوسع السريع في تقنيات إنترنت الأشياء (IoT) إلى ظهور تحديات أمنية كبيرة نتيجة الزيادة الهائلة في عدد الأجهزة المتصلة والمعرضة لمختلف الهجمات السيبرانية. وفي هذا السياق، تلعب أنظمة كشف التسلل (IDS) دوراً أساسياً في حماية بيئات إنترنت الأشياء من الأنشطة الضارة ومحاولات الاختراق.

تهدف هذه الدراسة إلى تصميم وتطوير نظام فعال لكشف التسلل يعتمد على تقنيات تعلم الآلة والتعلم العميق من أجل تعزيز أمن بيئات إنترنت الأشياء. وقد تم في البداية إجراء دراسة نظرية شاملة حول مفاهيم إنترنت الأشياء، وبنيتها، وبروتوكولات الاتصال الخاصة به، إضافة إلى التحديات الأمنية المرتبطة به، مع تقديم نظرة عامة حول أنظمة كشف التسلل ومجموعات البيانات الأكثر استخداماً في هذا المجال.

في الجانب التطبيقي، تم استخدام مجموعة بيانات (CICIoT2023) ومعالجتها عبر عدة مراحل من المعالجة المسبقة، شملت تنظيف البيانات، والترميز، والتطبيع، واختيار الخصائص، ومعالجة اختلال توازن الفئات باستخدام تقنيات هجينة لإعادة أخذ العينات. بعد ذلك، تم تدريب وتقييم عدة نماذج من تعلم الآلة والتعلم العميق باستخدام مقاييس الأداء القياسية مثل الدقة، والاسترجاع، ومعامل (F1). كما تم تحليل مؤشرات الكفاءة الحسابية، مثل زمن التدريب، وسرعة الاستدلال، وحجم النموذج، لتقييم مدى ملائمة هذه النماذج لبيئات إنترنت الأشياء.

أظهرت النتائج التجريبية أن نماذج تعلم الآلة القائمة على الأشجار، وخاصة (XGBoost) و (LightGBM)، حققت أداءً مرتفعاً في التصنيف مع الحفاظ على تكلفة حسابية منخفضة مقارنة بنماذج التعلم العميق. بالإضافة إلى ذلك، تم دمج تقنيات التفسير مثل (SHAP) لتحسين قابلية تفسير النماذج وتحديد الخصائص الأكثر تأثيراً في عملية اتخاذ القرار. وفي الختام، تؤكد هذه الدراسة فعالية تقنيات تعلم الآلة في تحسين أنظمة كشف التسلل الخاصة بإنترنت الأشياء، مع إبراز أهمية تحقيق التوازن بين دقة الكشف، والكفاءة الحسابية، وقابلية تفسير النماذج في تطبيقات الأمن السيبراني الحديثة.

الكلمات المفتاحية: إنترنت الأشياء (IoT)، نظام كشف التسلل (IDS)، الأمن السيبراني، تعلم الآلة، التعلم العميق، XGBoost، SHAP، مجموعة بيانات CICIoT2023، الذكاء الاصطناعي القابل للتفسير XAI، اختلال توازن البيانات.

Contents

Dedications	i
Acknowledgments	iii
Abstract	iv
Contents	vii
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Context	2
1.2 Problem Statement and Objective	3
1.3 Organization of the Thesis	4
2 Foundations of IoT Cybersecurity and IDS	5
2.1 Introduction	6
2.2 Fundamentals of IOT	6
2.2.1 Definition of IOT	6
2.2.2 Characteristics of IOT	6
2.2.3 IOT Architecture	7
2.3 IOT Security Challenges	8
2.3.1 IoT Security and Privacy Issues	8
2.3.2 Recommended Solutions	9
2.4 Cybersecurity and Intrusion Detection Systems (IDS)	10
2.4.1 Cybersecurity	11
2.4.2 Intrusion Detection Systems (IDS)	16
2.5 Intrusion Detection Datasets	20
2.5.1 Datasets	21
2.5.2 Comparison between datasets	26
2.6 Conclusion	26

3	DM and ML Techniques for IoT Protection	28
3.1	Introduction	29
3.2	Data Mining	29
3.2.1	Definition of Data Mining	29
3.2.2	Data Mining Process	29
3.2.3	Data Mining Tasks	31
3.2.4	Role of Data Mining in Cybersecurity	32
3.3	Machine Learning	32
3.3.1	Definition of Machine Learning	32
3.3.2	Types of Machine Learning	32
3.3.3	Machine Learning Algorithms	33
3.3.4	Machine Learning Challenges	33
3.4	Deep Learning	34
3.4.1	Definition of Deep Learning	34
3.4.2	Deep Learning Models	34
3.4.3	Deep Learning Challenges	35
3.5	Related Works	36
3.6	Conclusion	37
4	Experimental Design and Performance Evaluation	39
4.1	Introduction	41
4.2	Dataset Presentation	41
4.2.1	CICIoT2023 Dataset Attributes and Feature Description	42
4.3	Execution Environment	43
4.4	Data Preprocessing	44
4.4.1	Data cleaning	44
4.4.2	Attack label grouping	45
4.4.3	Encoding of Categorical Labels	45
4.4.4	Handling Imbalanced Data	46
4.4.5	Data Normalization	46
4.4.6	Feature Selection	47
4.5	Model Application and Evaluation	48
4.5.1	Performance Evaluation Metrics	49
4.5.2	Machine Learning Models	50
4.5.3	Deep Learning Models	54
4.6	Global Model Comparison	59
4.6.1	Comparison Between Machine Learning and Deep Learning Models	59
4.6.2	Performance and Computational Efficiency Analysis	60
4.7	Model Interpretability Using SHAP	62

4.7.1	SHAP Results for Binary Classification	63
4.7.2	SHAP Results for Multi-Class Classification	65
4.7.3	Discussion of SHAP Findings	66
4.8	Conclusion	68
5	General Conclusion	69
5.1	General Conclusion	69
5.1.1	Summary of Work	69
5.1.2	Key Findings	70
5.1.3	Concluding Remarks	70
	Bibliography	71

List of Figures

2.1	Four-Layer Architecture of IoT Systems	7
2.2	Overview of IoT Security Challenges	8
2.3	The CIA Triad in Cybersecurity	11
2.4	General Classification of IDS	17
2.5	General Architecture of IDS	19
2.6	Number of Instances by Traffic Types in the KDD Cup 99 Dataset.	21
2.7	Number of Instances by Traffic Types in the NSL-KDD Dataset.	22
2.8	Number of Instances by Traffic Types in the UNSW-NB15 Dataset.	23
2.9	Number of Instances by Traffic Types in the CICIDS2017 Dataset.	23
2.10	Number of Instances by Traffic Types in the BoT-IoT Dataset.	24
2.11	Number of Instances by Traffic Types in the IoT-23 Dataset.	25
2.12	Number of rows for each scenario.	26
3.1	Knowledge Discovery Databases (KDD) Process Model	30
3.2	CRISP-DM Process Model	30
4.1	Selected Features (Multi-class)	48
4.2	Selected Features (Binary)	48
4.3	Binary Confusion Matrix (Random forest)	51
4.4	Multi-class Confusion Matrix (Random Forest)	51
4.5	Binary Confusion Matrix (XGBoost)	53
4.6	Multi-class Confusion Matrix (XGBoost)	53
4.7	Binary Confusion Matrix (LightGBM)	54
4.8	Multi-class Confusion Matrix (LightGBM)	54
4.9	Binary Confusion Matrix (1D-CNN)	56
4.10	Multi-class Confusion Matrix (1D-CNN)	56
4.11	Binary Confusion Matrix (DNN)	57
4.12	Multi-class Confusion Matrix (DNN)	57
4.13	Binary Confusion Matrix (LSTM)	59
4.14	Multi-class Confusion Matrix (LSTM)	59
4.15	SHAP Feature Importance for Binary Classification	63

4.16 SHAP Summary Plot (Feature Impact Distribution) for Binary Classification	64
4.17 SHAP Global Feature Importance (Multi-Class Classification)	65
4.18 SHAP Summary Plot for Multi-Class Classification	66

List of Tables

2.1	Comparison of Intrusion Detection Datasets	26
3.1	Data Mining Process Models Comparison	31
4.1	Comparison of Original and Stratified Versions of the CICIoT2023 Dataset	42
4.2	Description and Categorization of the CICIoT2023 Dataset Features	43
4.3	Random Forest Hyperparameters	50
4.4	Overall Performance: Binary vs. Multi-class Random Forest	50
4.5	Multi-class Classification Report per Attack Type	51
4.6	XGBoost Hyperparameters	52
4.7	Overall Performance: Binary vs. Multi-class XGBoost	52
4.8	Multi-class Classification Report per Attack Type (XGBoost)	52
4.9	LightGBM Hyperparameters	53
4.10	Overall Performance: Binary vs. Multi-class LightGBM	53
4.11	Multi-class Classification Report per Attack Type (LightGBM)	54
4.12	1D-CNN Model Configuration	55
4.13	Overall Performance: Binary vs. Multi-class 1D-CNN	55
4.14	Multi-class Classification Report per Attack Type (1D-CNN)	55
4.15	DNN Hyperparameters	56
4.16	Overall Performance: Binary vs. Multi-class DNN	57
4.17	Multi-class Classification Report per Attack Type (DNN)	57
4.18	LSTM Hyperparameters	58
4.19	Overall Performance: Binary vs. Multi-class LSTM	58
4.20	Multi-class Classification Report per Attack Type (LSTM)	58
4.21	Comprehensive Comparison of Models (Performance and Efficiency)	61

Chapter 1

Introduction

Contents

1.1	Context	2
1.2	Problem Statement and Objective	3
1.3	Organization of the Thesis	4

1.1 Context

The rapid expansion of the [Internet Of Things \(IOT\)](#) has fundamentally transformed the digital landscape, connecting billions of devices across critical sectors such as health-care, transportation, energy, and manufacturing. From smart home appliances and wearable health monitors to industrial sensors and autonomous vehicles, these connected systems have ushered in an era of unprecedented automation, efficiency, and data-driven decision-making. However, this massive proliferation has also introduced significant security challenges. Each connected device represents a potential entry point for malicious actors, and the inherent heterogeneity of [IOT](#) protocols including MQTT, CoAP, and 6LoWPAN combined with the limited computational resources of these systems, creates a uniquely complex and expanded attack surface.

In response to these challenges, [Intrusion Detection System \(IDS\)](#) have become essential components of modern network security architectures. These systems are designed to monitor network traffic, analyze system activities, and detect anomalous or malicious behavior in real time, enabling administrators to respond before significant damage occurs. Traditional signature-based [IDS](#) rely on databases of known attack patterns; while effective against established threats, they are inherently incapable of detecting novel or zero-day attacks. Anomaly-based [IDS](#), which learn models of normal behavior and flag deviations, offer a more adaptive alternative. Nevertheless, their effectiveness depends heavily on the quality, representativeness, and balance of the data used for training – a particularly acute challenge in the [IOT](#) domain.

To enhance the performance of anomaly-based [IDS](#), researchers have increasingly turned to [Machine Learning \(ML\)](#) and [Deep Learning \(DL\)](#) techniques. These approaches excel at identifying subtle, non-linear patterns in large volumes of data and have demonstrated impressive detection capabilities. Among [ML](#) techniques, ensemble methods such as [Random Forest \(RF\)](#), [eXtreme Gradient Boosting \(XGBoost\)](#), and [Light Gradient Boosting Machine \(LightGBM\)](#) have gained prominence due to their robustness and high accuracy. Within [DL](#), [Convolutional Neural Network \(CNN\)](#) – particularly one-dimensional variants – have shown promise in capturing local patterns in network traffic data. However, the practical deployment of such models on resource-constrained [IOT](#) gateways remains an open problem. Most existing studies focus exclusively on maximizing detection accuracy, neglecting critical factors such as model size, inference latency, and energy consumption – parameters that directly determine whether a model can operate effectively at the network edge.

The availability of appropriate datasets is another fundamental challenge. Early bench-

mark datasets such as NSL-KDD and UNSW-NB15, while valuable for their time, do not reflect the traffic patterns, protocols, or attack types encountered in modern **IOT** environments. Recent efforts have introduced dedicated **IOT** security datasets that better capture contemporary threats. Among these, the CICIoT2023 dataset, developed by the Canadian Institute for Cybersecurity, stands out as one of the most comprehensive and realistic benchmarks available today. It contains real traffic from 105 physical **IOT** devices and includes 33 distinct attack types across seven categories, with over 97% of samples representing malicious activity – a realistic and challenging class imbalance scenario.

Class imbalance itself remains one of the most significant obstacles to effective **IDS** performance. When benign traffic vastly outnumbers malicious instances, models become biased toward the majority class, leading to poor detection rates for rare but potentially devastating attacks. To address this issue, oversampling techniques such as **Synthetic Minority Over-sampling Technique (SMOTE)** have been widely adopted to generate synthetic samples for minority classes, thereby improving class distribution and enhancing model generalization. Yet the impact of **SMOTE** on both classification performance and computational efficiency in **IOT IDS** contexts remains underexplored.

Beyond accuracy and efficiency, interpretability has emerged as a critical requirement for practical **IDS** deployment. Network administrators are understandably reluctant to act on alerts generated by black-box models whose decision-making processes they cannot understand or trust. The field of **Explainable Artificial Intelligence (XAI)** addresses this gap by providing human-interpretable justifications for model predictions. Among **XAI** techniques, **SHapley Additive exPlanations (SHAP)** has gained particular attention for its ability to attribute predictions to individual features in a theoretically sound and visually intuitive manner. Integrating such interpretability into **IOT IDS** would not only increase trust but also assist security analysts in understanding the nature and origin of detected attacks.

1.2 Problem Statement and Objective

This thesis addresses a gap that runs through much of the existing **IOT IDS** literature: the near-exclusive focus on detection accuracy, to the detriment of computational efficiency and model interpretability two properties that are equally critical for practical deployment on resource-constrained edge devices.

The main objectives are:

- Preprocess the CICIoT2023 dataset through a structured pipeline that handles class imbalance using a hybrid resampling strategy combining under-sampling with

SMOTE, and reduces input dimensionality through automated, importance-guided feature selection.

- Implement and compare six classification models drawn from two distinct paradigms: three ML models: Random Forest, XGBoost, and LightGBM — and three DL models: a 1D-CNN, a DNN, and an LSTM network, so as to investigate whether deep architectures offer a meaningful advantage over ensemble-based methods on structured IOT traffic data.
- Evaluate all models under both binary and multi-class classification scenarios using standard performance metrics (precision, recall, and F1-score), complemented by computational efficiency indicators (model size, training time, and inference latency) that directly reflect the constraints of real IOT gateway deployment.
- Apply SHAP to explain the predictions of the best-performing model, identifying the network traffic features that most influence the detection decision and connecting those features to known attack behaviors.
- Derive practical recommendations for deploying lightweight, interpretable IDS solutions on constrained IOT gateways, where memory footprint, processing budget, and response latency are all binding factors.

1.3 Organization of the Thesis

This thesis is structured into five chapters, each addressing a specific aspect of the research:

- Chapter 1: Introduction — Provides the background, security challenges, problem statement, and research objectives.
- Chapter 2: Foundations of IOT Cybersecurity and IDS — Covers IOT fundamentals, cybersecurity principles, IDS, and reviews existing datasets, with emphasis on CICIoT2023.
- Chapter 3: Data Mining and Machine Learning Techniques for IOT Protection — Discusses data mining, ML, and DL techniques, and reviews related works on anomaly-based IDS.
- Chapter 4: Experimental Design and Performance Evaluation — Details the dataset, execution environment, preprocessing pipeline, model implementation, and comparative evaluation including efficiency metrics and SHAP analysis.
- Chapter 5: Conclusion — Summarizes findings.

Chapter 2

Foundations of IOT Cybersecurity and IDS

Contents

2.1	Introduction	6
2.2	Fundamentals of IOT	6
2.2.1	Definition of IOT	6
2.2.2	Characteristics of IOT	6
2.2.3	IOT Architecture	7
2.3	IOT Security Challenges	8
2.3.1	IoT Security and Privacy Issues	8
2.3.2	Recommended Solutions	9
2.4	Cybersecurity and Intrusion Detection Systems (IDS)	10
2.4.1	Cybersecurity	11
2.4.2	Intrusion Detection Systems (IDS)	16
2.5	Intrusion Detection Datasets	20
2.5.1	Datasets	21
2.5.2	Comparison between datasets	26
2.6	Conclusion	26

2.1 Introduction

The rapid growth of (IoT) has enabled billions of connected devices across various domains, but it also introduces major cybersecurity challenges due to device diversity and large-scale connectivity. To address these risks, security mechanisms such as (IDS) are used to monitor network traffic and detect malicious activities.

Developing effective IDS solutions requires realistic datasets; however, many existing datasets do not accurately represent modern IoT environments. Recent datasets like CICIoT2023 provide more realistic traffic and diverse attack scenarios. This chapter presents the basics of IoT and cybersecurity, explains IDS concepts, and reviews IoT security datasets with a focus on CICIoT2023.

2.2 Fundamentals of IoT

2.2.1 Definition of IoT

The IoT can be described as a global and dynamic network infrastructure capable of self-configuration through the use of standardized and interoperable communication protocols. It connects both physical and virtual entities, referred to as “things,” which possess unique identities, physical characteristics, and virtual representations. These entities are equipped with intelligent interfaces that enable seamless integration into information networks and facilitate continuous data exchange related to users and their surrounding environments [1].

2.2.2 Characteristics of IoT

IoT systems exhibit several key characteristics that distinguish them from traditional networks [1]. These include:

1. Dynamic and Self-Adaptive:

IoT systems can respond and adapt to changing conditions in real time. Devices adjust their behavior based on operational status, user requirements, or environmental data. For example, in a surveillance system, cameras can automatically switch between normal and infrared modes depending on lighting conditions. When motion is detected, they may increase resolution and alert nearby devices, demonstrating adaptive behavior.

2. Self-Configuring:

IoT systems are capable of automatically configuring themselves without significant human intervention. Large groups of devices can establish connections, integrate into networks, and update their software autonomously, ensuring scalability and efficiency.

3. Interoperable Communication Protocols:

IOT devices rely on standardized communication protocols that enable interoperability. This allows heterogeneous devices and systems to communicate and exchange data efficiently across different platforms.

4.Unique Identity:

Each **IOT** device possesses a unique identifier, such as an IP address or URI, which enables proper identification and communication within the network. This uniqueness facilitates device management and continuous monitoring.

2.2.3 **IOT** Architecture

The architecture of the Iot is commonly organized into four layers, as illustrated in Figure [2]. .

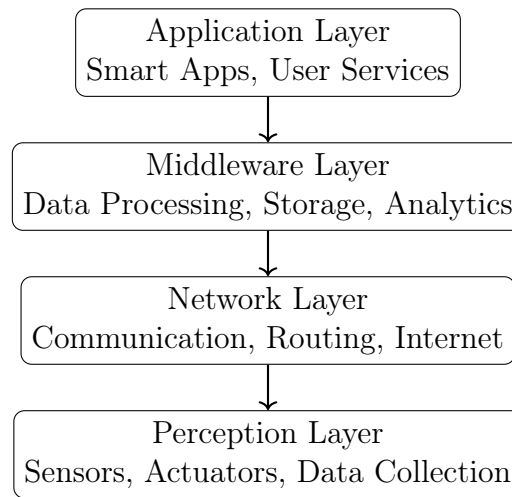


Figure 2.1: Four-Layer Architecture of IoT Systems

1.Perception Layer

The Perception Layer represents the physical environment of the **IOT** system. It consists of sensors, actuators, and embedded devices that collect data from the surroundings, such as temperature, motion, and device status. This layer is responsible for data acquisition and initial sensing, providing the raw information required for higher-level processing .

2.Network Layer

The Network Layer ensures the transmission of data between devices and system components. It includes communication technologies such as wired networks (e.g., LAN) and wireless networks (e.g., Wi-Fi, cellular, and IoT-specific protocols). This layer enables reliable and secure data transfer from the Perception Layer to the upper layers and supports connectivity across the entire system .

3.Middleware Layer

The Middleware Layer handles data processing, storage, and management. It acts as an intermediary between the hardware and application layers. In this layer, raw data is

filtered, analyzed, and stored. It also supports decision-making processes by transforming unstructured data into meaningful information and providing services such as device management and data analytics.

4. Application Layer

The Application Layer delivers services to end users based on the processed data. It includes various applications such as smart healthcare, smart cities, industrial monitoring, and intelligent transportation systems. This layer provides user interfaces and visualization tools that allow users to monitor system performance, analyze results, and make informed decisions .

2.3 IOT Security Challenges



Figure 2.2: Overview of IoT Security Challenges

2.3.1 IoT Security and Privacy Issues

IoT systems face numerous security and privacy challenges due to their distributed nature and heterogeneous components. These challenges can be analyzed across different layers of the IoT architecture [3]:

1. Perception Layer Security

This layer includes physical devices and sensors, making it vulnerable to several attacks:

- **Sybil Attack:** Malicious nodes create multiple fake identities to disrupt network operations and consume resources.
- **Jamming Attack:** Attackers interfere with wireless communication by emitting radio signals, causing network disruption.
- **Sleep Deprivation Attack:** Devices are forced to remain active, leading to excessive energy consumption and reduced lifetime.

2. Network Layer Security

This layer handles data transmission and routing, making it susceptible to:

- **Sinkhole Attack:** A malicious node attracts network traffic and manipulates or drops packets.

- Wormhole Attack: Attackers create a tunnel between distant nodes to alter routing paths.

3. Transport Layer Security

This layer ensures end-to-end communication and is exposed to threats such as:

- Session Hijacking: Attackers take control of an active communication session.
- Message Manipulation: Data packets can be altered during transmission, affecting integrity.
- Secure communication requires encryption and strong authentication mechanisms.

4. Application Layer Security

At this level, security issues mainly affect software and user interfaces:

- Unsafe Interfaces: Vulnerable APIs and web interfaces may expose sensitive data.
- Weak Middleware Protection: Poorly secured middleware can lead to unauthorized access and data leakage.
- Insecure Software/Firmware: Outdated or poorly secured code can introduce vulnerabilities such as injection attacks or unauthorized system access.

2.3.2 Recommended Solutions

IoT security threats target multiple components across different layers, including firmware, network resources, physical devices, software, and applications. Users interact with these components through various interfaces, which can themselves become points of vulnerability. Therefore, security mechanisms must be properly designed and enforced to prevent system compromise [3].

1. Low-level Security Solutions

- Sybil Attacks: A malicious node may impersonate multiple identities using fake MAC addresses, leading to resource exhaustion and network disruption. Detection techniques include:
 - Signal Intensity Analysis: Identifies inconsistencies in signal strength.
 - Channel Prediction: Uses channel characteristics to detect abnormal behavior.
- Jamming Attacks: These attacks disrupt wireless communication. Countermeasures include:
 - Monitoring signal strength and packet delivery ratio.

- Using error-correcting codes and encryption.
- Channel surfing and spatial retreat techniques.
- Sleep Deprivation Attacks: Cluster-based architectures reduce energy consumption by organizing nodes into sectors. Multi-layer IDS are deployed, where higher-level nodes provide advanced monitoring while lower-level nodes handle basic detection.

2. Intermediate-level Security Solutions

- Secure Device Communication: Encryption techniques such as Elliptic Curve Cryptography (ECC) ensure secure data transmission, authentication, and key management.
- Authentication Mechanisms: Protocols like VeRA use hash functions and digital signatures to prevent attacks such as rank manipulation and unauthorized access.
- Cloud-based Privacy: Secure frameworks ensure data protection during transmission and storage, preventing data leakage and unauthorized modifications.

3. High-level Security Solutions

- Middleware Security: Middleware platforms ensure secure communication and data integrity:
 - VIRTUS: Uses TLS, authentication, and encryption mechanisms.
 - Otsopack: Enables secure communication between heterogeneous IoT systems.
- Access Control and AAA: Authorization, authentication, and accounting mechanisms are used to control access and monitor system activity.

2.4 Cybersecurity and Intrusion Detection Systems (IDS)

As a result of a number of reasons, such as the increasing number of information systems being accessible via the Internet, the fast pace at which communications technology is developing and the sheer volume of data being transmitted across networks, new risks have arisen regarding unauthorised access to and manipulation of data, whether this is unintentional or intentional. Thus, it is now essential to implement a strong security policy around these systems.

One of the main defenses in place for protecting networks from attacks and for providing constant solution monitoring is through the use of IDS.

2.4.1 Cybersecurity

2.4.1.1 Definition

Cybersecurity refers to all the methods and ways that protect information systems and the data that are stored, sent, or processed by them. The protection provided by cybersecurity has three main objectives: maintaining the integrity, availability, and confidentiality (C.I.A) of resources that the information systems are using or storing. [4]

The CIA triad is a name used to describe the three main concepts of information security that make up the basis of information security.

2.4.1.2 CIA triad

The CIA triad is a core principle in information security and acts as a guide to security system design, making up the foundational components of security within the context of cybersecurity. The three principles within this model include Confidentiality, Integrity, and Availability and each principle has its own overarching objective. [4]

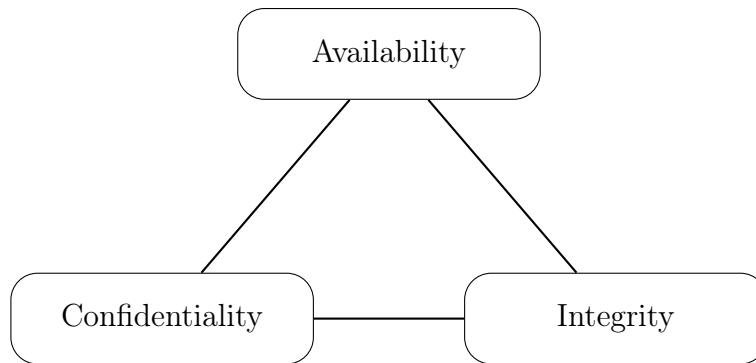


Figure 2.3: The CIA Triad in Cybersecurity

1. Confidentiality: ensures that private information is accessible only to authorized entities (people, organizations, or systems) by employing mechanisms such as encryption, authentication, and access control policies to prevent unauthorized data disclosure.

2. Integrity : protects the accuracy, completeness, and authenticity of data by preventing unauthorized or accidental modification, addition, or deletion during storage, transmission, or processing.

3. Availability : ensures that authorized users can reliably access information systems and data when needed, despite potential disruptions such as hardware failures, cyberattacks, or network congestion.

2.4.1.3 Cyber Attacks

A cyber attack is an intentional act that seeks to use weaknesses in a network of computers, software, or devices, to breach security and access confidential data without authorization;

or steal confidential data (i.e., trade secrets or intellectual property); or disrupt the normal operation of an information system.

2.4.1.4 Cyberattack Lifecycle

The cyberattack lifecycle is a system that describes all the phases of a common cyber-attack. The model was created by Lockheed Martin and is now used by most of the cybersecurity community. [5, 6]

Understanding the stages of the cyberattack lifecycle enables organizations to develop proactive defenses, detect breaches in progress, and implement incident response plans to minimize impact and enable rapid recovery.

1. **Reconnaissance:** the first stage of the cyberattack lifecycle, involves collecting information about a target system or organization—such as network ranges, IP addresses, DNS records, employee emails, OS versions, and publicly available organizational data. Attackers use passive techniques (e.g., mining public or social media data) or active techniques (e.g., port and vulnerability scans), as well as social engineering and phishing. The quality of reconnaissance directly determines the success of subsequent attack stages.

2. **Weaponization:** is the stage where attackers prepare the means to exploit their target by creating or modifying malicious tools—such as malware payloads, exploit kits, backdoors, or ransomware components—tailored to vulnerabilities identified during reconnaissance. This involves combining an exploit with a payload to deliver the attack, often before traditional security measures can detect it.

3. **Delivery:** is the stage where attackers transmit the weaponized payload to an unsuspecting victim using methods such as phishing emails (with malicious attachments or links), drive-by downloads, infected USB devices, or network-based attacks. From a defensive perspective, this phase is critical because security products like firewalls, email filters, IDS, and IPS aim to detect and block attacks at this point.

4. **Exploitation:** occurs when the attacker successfully runs a malicious payload on the target machine by leveraging vulnerabilities—such as insecure applications, malicious macros in documents, or stolen credentials. This grants initial system access. IDS can help identify abnormal activity or exploitation attempts during this phase.

5. **Installation:** is the stage where an attacker, having gained initial access, deploys additional malicious components—such as backdoors, new user accounts, altered system configurations, or remote access tools—to establish persistence. This ensures continued access even if the original vulnerability is patched, allowing re-entry without repeating the full attack process.

6. **Command and Control (C2):** is the phase where an attacker establishes a communication link between a compromised system and an external control server, enabling remote commands, additional malware downloads, lateral network movement, and data extraction. Since C2 traffic can mimic legitimate network activity, detecting unusual

outbound traffic patterns is essential for identification.

7. **Actions on Objectives:** is the final phase of the cyberattack lifecycle, where the attacker executes their ultimate goals—such as stealing sensitive information, disrupting services, committing financial fraud, espionage, or altering/destroying data. Successful execution can compromise all three pillars of the CIA triad: Confidentiality, Integrity, and Availability, with potentially disastrous consequences for the organization.

2.4.1.5 Classification of CyberAttacks

The classification of Cyber Attacks can be done in various ways based on several criteria including attacks target and purpose, the impact they have or how serious they could become, and their intended effects on systems [4]. Clear classifications will help develop effective measures for understanding cyber threats and developing effective methods to detect those threats.

1. **Classification Based on Impact:** fall into two categories. Passive attacks aim to acquire sensitive information without modifying system resources or disrupting normal operations (e.g., monitoring communications), making them harder to detect. Active attacks involve modification, interruption, or destruction of system resources (e.g., altering or deleting data), potentially compromising confidentiality, integrity, and availability, but often leaving observable anomalies.

2. **Classification Based on Target:** focus on network infrastructure, software applications, or internal organizational entities. Network attacks target communication channels and infrastructure to intercept, disrupt, or manipulate data in transit. Application attacks exploit software vulnerabilities (e.g., web services, databases, APIs) to manipulate logic or gain unauthorized data access. Malware attacks use malicious software to compromise, destroy, or control systems. Insider attacks originate from authorized users abusing their access rights, whether intentionally or unintentionally, posing a serious security threat.

2.4.1.6 Examples of Cyber Attacks:

The following examples illustrate common cyber attacks targeting networks, applications, and devices. Each demonstrates different methods attackers use to compromise confidentiality, integrity, or availability [4, 7].

1. **IP Spoofing:** An IP spoofing attack is a network-level attack where an attacker forges the source IP address of packets to impersonate a trusted device, bypassing IP-based authentication. It is commonly used as a basis for larger attacks such as DDoS and Man-in-the-Middle (MITM).
2. **Denial of Service (DoS):** makes a system or network unavailable to legitimate users by overwhelming it with traffic or exploiting software flaws, consuming bandwidth,

memory, or CPU until performance degrades or stops. A Distributed Denial of Service (DDoS) attack uses multiple compromised systems to launch the attack.

3. **Man-in-the-Middle (MITM):** occurs when an attacker secretly intercepts communications between two parties, enabling eavesdropping, message modification, or data injection. By exploiting weak encryption or unprotected connections, attackers can steal sensitive information such as authentication credentials or financial data.
4. **SQL Injection:** attacks exploit programming weaknesses in database-driven applications by inserting unauthorized SQL code into user input fields, allowing access to or modification of confidential data and potentially full application control. It is a common and damaging web application vulnerability.
5. **Malware Attacks:** use malicious software to compromise, damage, or control information systems, delivered via downloaded files, email attachments, or compromised websites. Common types include viruses (attach to legitimate programs), worms (self-replicate on networks), Trojans (disguised as legitimate software), and ransomware (encrypts data and demands payment).
6. **IoT botnet attacks:** involve compromising thousands of IoT devices (e.g., routers) to form a remotely controlled botnet capable of launching large-scale attacks. The Mirai botnet exemplifies this threat, exploiting weak authentication protocols on vulnerable IoT devices to create highly scalable distributed attacks.

2.4.1.7 Security Techniques and Protection Mechanisms

The increasing sophistication of cyber attacks highlights the importance of deploying comprehensive security techniques and protection mechanisms. These methods aim to prevent, detect, and mitigate threats across different layers of information systems, from networks and applications to IoT devices. Security mechanisms can be classified into preventive, detective, and corrective measures, each serving specific purposes in maintaining the confidentiality, integrity, and availability of systems.

1. **Preventive Mechanisms:** Preventive mechanisms are designed to stop attacks before they can compromise a system. They reduce the attack surface by enforcing security policies and controlling access. Examples of preventive mechanisms:
 - **Firewalls:** Prevent unauthorized access to a network by controlling movement over the Internet through the use of predetermined guidelines established by network administrators.
 - **Encryption:** A means of ensuring that confidential information is protected from eavesdropping when it is transmitted from point A to point B and maintained in a secure location.

- Authentication and Access Control: Prevents unauthorized users and devices from accessing system resources.
- Secure Software Development: Incorporates security practices when developing or coding software to provide the lowest possible number of vulnerabilities to exploitation.

While Preemptive Functions may provide a user with an initial level of protection due to their ability to reduce the possibility of an intrusion, sophisticated types of attacks have developed that utilize methods that can evade the preemptive functions.

2. Detective Mechanisms: The purpose of detective mechanisms is to discover attacks or breaches currently happening or that have happened by observing system behavior, traffic and logs in order to find signs of irregularities or potentially malicious activity. Examples of Detective Mechanisms:

- Intrusion Detection Systems (IDS): are used to continuously observe network or host activity for indication of an attack.
- Log Analysis and SIEM (Security Information and Event Management): systems collect and analyze log information from multiple sources to identify suspicious activity.
- Network Monitoring Tools: allow for the detection of abnormal network activity such as/port scans or abnormal volume of traffic.

Detective mechanisms are critical to detecting attacks that were successful in bypassing prevention mechanisms and for providing a means of responding quickly.

3. Corrective Mechanisms: The goal of corrective mechanisms is to restore system integrity and recover from an attack. Corrective mechanisms complement the preventative and detective measures by mitigating the extent of the impacts of an attack and reducing the amount of time the system is down, and they may also assist with further strengthening the network against future attacks. Examples of Corrective Mechanisms:

- Backup and Recovery Systems: Restore data and system configurations to a known good state after a system has been compromised.
- Patch Management: Implement updates to fix security weaknesses in software or hardware.
- Incident Response Plans: Create structured, predefined methods for responding to and minimising the effects of cyber-attacks.

Corrective mechanisms create a resilient environment that enables business to continue and mitigate the effects of confirmed defeat.

4. Security in IoT Context: The security methods used to secure the IoT must be modified because of the constraints and differences of the devices connected to the network. The resource-limited devices that are connected to the IoT generally, have low computational power, weak authentication methods, and many different communication protocols. Therefore, the standard IT security measures used on systems are not sufficient.

The following are some of the primary metrics that can be used to measure security in IoT systems:

- Use of lightweight encryption and authentication methods that are applicable for devices with low resources.
- Use of secure firmware updates to avoid hacking or the installation of malicious software.
- Network separation and network monitoring create methods to separate compromised devices and to identify anomalies in the devices connected to the network.
- Integration with IDS allows real-time detection of threats throughout the IoT ecosystem.

2.4.2 Intrusion Detection Systems (IDS)

2.4.2.1 Definition of IDS

An IDS is a security tool that constantly watches system activity to catch unauthorized access. It focuses on three things: spotting illegal attempts to reach data or networks, finding misuse of organizational resources, and detecting unusual behavior that might threaten information security. [8]

IDSs don't block attacks but alert administrators and log attack details for a fast, damage-limiting response. They come in several types based on operation level: network-based (NIDS), host-based (HIDS), and specialized versions for IoT and industrial systems.

2.4.2.2 Types of IDS

Modern cybersecurity architectures nearly always include an IDS. Due to diverse network environments and threat models, IDSs are classified by various characteristics (architecture, decision-making method, distribution, response, and detection approach) to understand their operation, implementation, and threat coverage. Academic literature provides the most widely used classifications. [9]

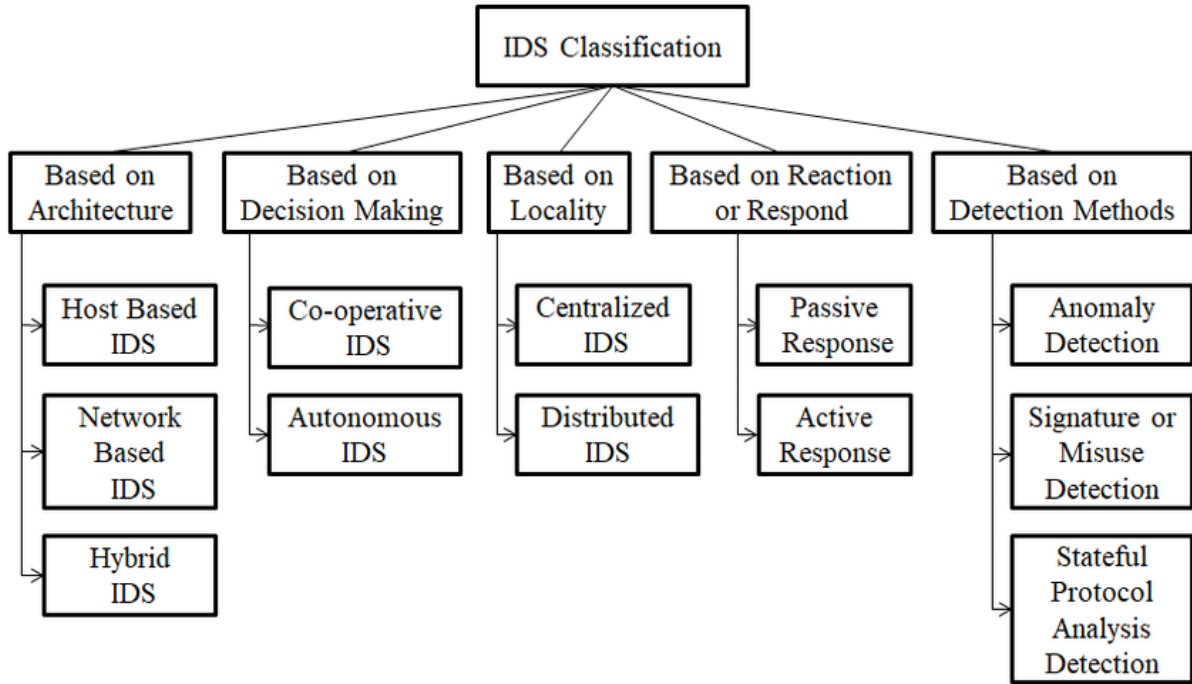


Figure 2.4: General Classification of IDS

1. Classification Based on Architecture: This classification defines how an IDS can be either physically or logically located in the monitored area.
 - Host-Based IDS (HIDS): is installed on individual devices (servers, PCs, IoT nodes) and monitors internal activity: logs, file integrity, user behavior, and process execution. It detects insider threats and unauthorized changes on that host, but cannot see activity on other devices.
 - Network-Based IDS (NIDS): monitors traffic over network communication links, analyzing packets, protocol behavior, and traffic patterns to detect suspicious activity. It protects multiple hosts at once and works well for perimeter defense but is less effective on encrypted traffic.
 - Hybrid IDS: combines host-based and network-based methods for comprehensive monitoring, improving detection rates and reducing blind spots in complex environments.
2. Classification Based on Decision-Making: This classification of IDS is focused on the method of creating decisions for intrusion detection.
 - Autonomous IDS: performs local data analysis and makes independent decisions without any external collaboration.
 - Cooperative IDS: shares information across multiple nodes to make collaborative intrusion detection decisions, improving detection capability. This method

is especially useful in distributed systems like IoT networks, where devices share alerts and threat information.

3. **Classification Based on Locality:** This category refers to where the analysis and processing of security data take place.

- **Centralized IDS:** sends data from multiple sources to a single location for analysis, enabling easier management and coordination. However, it introduces a single point of failure and scalability issues in large infrastructures.
- **Distributed IDS:** performs analysis on each individual node as part of a unified detection process, increasing overall security by involving all nodes. It is well-suited for IoT and cloud environments.

4. **Classification Based on Response Strategy:** IDS can be further divided based on their responses after detecting suspicious activity:

- **Passive IDS :** generates logs and alerts but does not take direct corrective action, requiring human intervention to resolve threats.
- **Active IDS :** automatically responds to threats by blocking traffic, terminating sessions, or modifying firewall rules. When combined with prevention measures, it functions as an Intrusion Prevention System (IPS).

5. **Classification Based on Detection Methodology:** The detection method represents a central point of intelligence for the IDS and is listed as one of the top criteria for classification when assessing an IDS.

- **Signature-Based Detection:** matches observed activity against a database of known attack patterns. It is highly accurate for known attacks but cannot detect zero-day (never-before-seen) attacks.
- **Anomaly-Based Detection:** defines normal system behavior and flags deviations as suspicious. It can detect unknown (zero-day) attacks but tends to have a higher false positive rate than signature-based detection.
- **Stateful Protocol Analysis:** analysis compares network traffic against predetermined protocol standards (e.g., expected states and outcomes), flagging discrepancies that may indicate protocol-targeted attacks.

2.4.2.3 Intrusion Detection System Architecture

IDS architectural design defines the layout and interaction of components for data collection, processing, analysis, and alert generation. It ensures effective monitoring, accurate detection, and responsive action to security events. [10]

While implementations of IDS solutions may differ from each other, they generally follow a common modular structure and will contain a number of essential components.

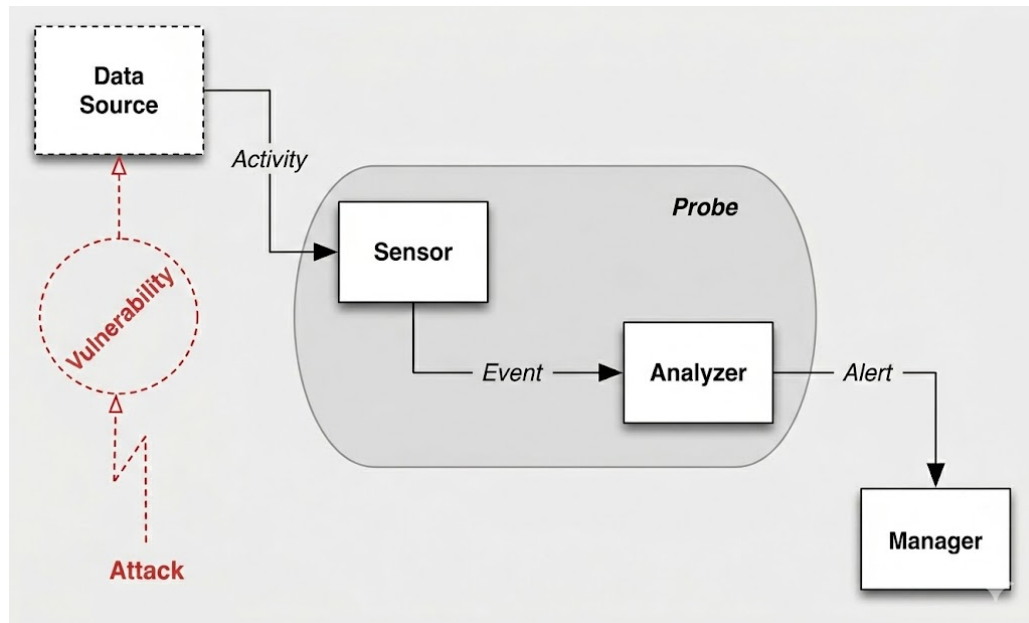


Figure 2.5: General Architecture of IDS

1. Vulnerability: is a weakness in software, hardware, configuration, or network design that attackers exploit to gain unauthorized access, leak confidential data, or disrupt services.
2. Data Source: is the origin of monitored information (e.g., network traffic, system logs, OS activities, user activity) that may contain evidence of malicious activity.
3. Sensor: collects and monitors activity from a data source, then forwards the captured data to an analyzer for evaluation. [10]
4. Analyzer: processes collected events using detection techniques (e.g., signature-based) to determine if behavior is normal or indicates an intrusion. [10]
5. Manager: serves as the IDS central control, receiving alerts, logging events, providing graphical event displays, and enabling administrator control over incident responses and security policies. [10]

2.4.2.4 IDS in IOT Environments

IoT environments challenge traditional IDS due to device heterogeneity and resource constraints (processing, memory, energy). Conventional IDS may be unsuitable, so IoT solutions must be lightweight or distributed.

Furthermore, IoT environments are dynamic and scalable, with growing traffic and attack surfaces, making real-time intrusion detection complex. Modern IoT IDS solutions increasingly use machine learning and intelligent methods for efficient, effective, and scalable security monitoring.

2.4.2.5 Challenges of IDS in IOT

There are several key challenges encountered when deploying an Intrusion Detection System (IDS) in IoT space due to the features of IoT systems. [11]

- **Resource Limitations:** Most IoT devices face CPU, memory, storage, and battery limitations, making it challenging to run complex detection algorithms for identifying sophisticated attacks.
- **Heterogeneity (Device Variety):** IoT device heterogeneity (various protocols and operating systems) complicates developing a consistent, compatible IDS across all devices.
- **Scalability:** Scalability in IoT networks involves monitoring real-time traffic from hundreds or thousands of devices without performance loss.
- **High Data Volume:** High data volume from IoT sensors creates massive network traffic, requiring optimized detection systems for effective processing and analysis.
- **Dynamic Topology:** Frequent device movement in IoT networks causes dynamic topology and fluctuating device counts, requiring IDS to operate effectively under such conditions.
- **High False Positives:** Random traffic patterns across many IoT devices and protocols make it hard to distinguish normal from malicious behavior, leading to high false positive rates that reduce IDS trustworthiness.

2.5 Intrusion Detection Datasets

Datasets are essential to the process of developing, training, and evaluating the performance of IDS. Datasets consist of records of network traffic collected over a period of time that are tagged either as normal or malicious. These datasets are utilized to determine how effective, accurate, and reliable an IDS is, particularly with respect to IDS implementations using machine learning or deep learning techniques.

Thus, it is critical to select an appropriate dataset, as an IDS model's performance is greatly influenced by the quality, variety, and realism of the data contained within a particular dataset.

An effective IDS dataset must:

- Contain labeled traffic (benign and attack).
- Include multiple attack categories.
- Reflect realistic network behavior.
- Provide relevant statistical and flow-based features.
- Avoid excessive redundancy and bias.

Selecting datasets is even more important for IoT implementations than for traditional IDS implementations because they have fewer resources available, there are many different devices available, and the attack vectors keep changing.

2.5.1 Datasets

- KDD Cup 99: The KDD Cup 1999 dataset (KDD99) consists of about 4.9 million instances of simulated network connection records. There are 41 features that describe each of the records. KDD99 categorizes network attacks into four types: Denial of Service (DoS), Probing, Remote to Local (R2L), and User to Root (U2R). Even though the KDD Cup 1999 dataset is important for IDS research, it has a significant amount of redundancy in the number of records, the data is highly imbalanced, and the dataset reflects network traffic patterns from many years ago—making it unsuitable for the most recent IoT. [12]

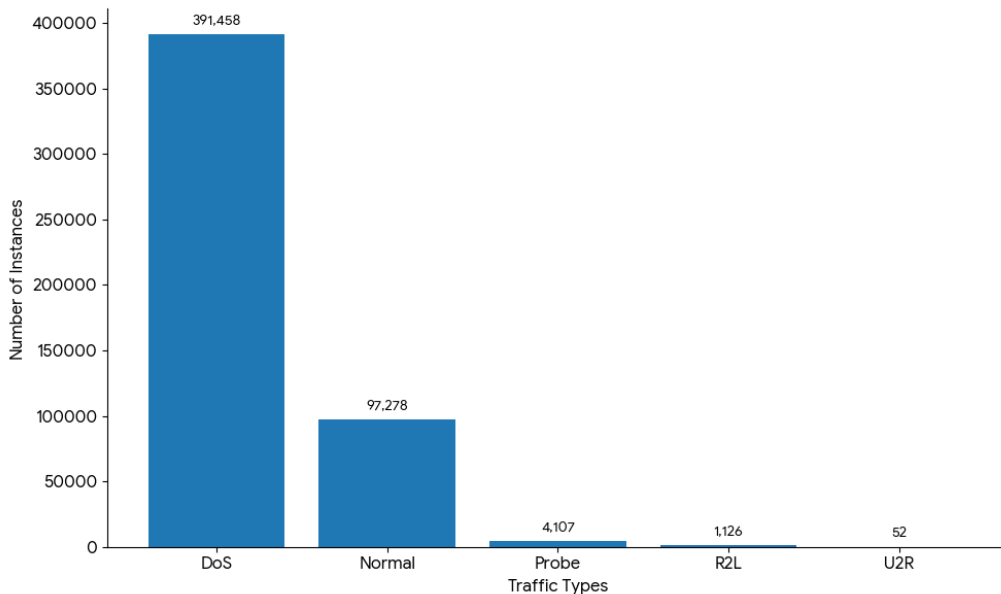


Figure 2.6: Number of Instances by Traffic Types in the KDD Cup 99 Dataset.

- NSL-KDD dataset: The NSL-KDD dataset was designed to resolve the inherent problems of redundant and saturated records found in the original KDD Cup 99

benchmark [12]. While it retains the exact same feature structure consisting of 41 flow-based attributes, it provides a much more balanced and cleaned distribution of samples to support the fair evaluation of classification algorithms without majority class bias.

Although there have been substantial improvements made to the NSL-KDD framework, its high similarity to traditional network traffic severely limits its modern usability. As emphasized in the literature [11, 13], because it represents a static, non-IoT network environment, it is fundamentally incapable of evaluating contemporary security risks associated with modern IoT communication protocols (such as MQTT and CoAP) and the rapidly changing nature of edge device attack strategies.

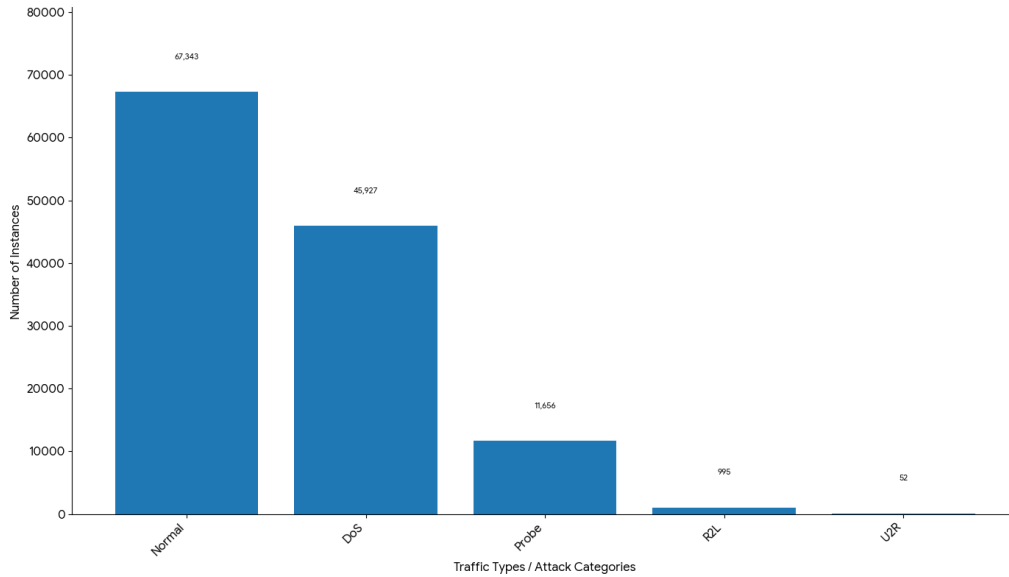


Figure 2.7: Number of Instances by Traffic Types in the NSL-KDD Dataset.

- UNSW-NB15: The UNSW-NB15 data set consists of 49 flow-based attributes as well as nine types of attack including Exploit, Fuzzer, and Reconnaissance [14]. Compared to KDD datasets, the UNSW-NB15 dataset has updated attack patterns; this dataset provides information about current cyberattacks. However, the UNSW-NB15 Dataset focuses on business networks and not IoT environments [11].

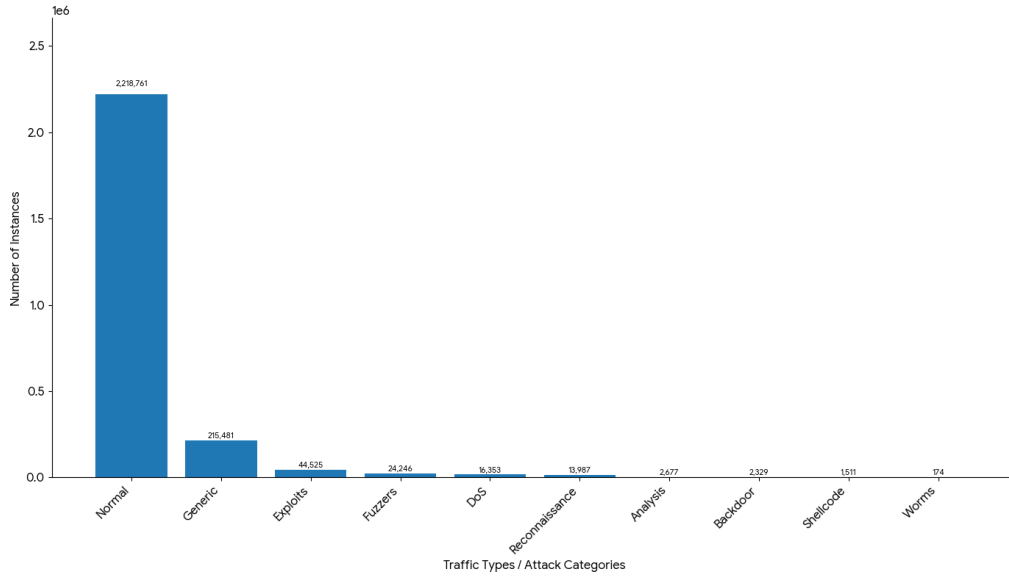


Figure 2.8: Number of Instances by Traffic Types in the UNSW-NB15 Dataset.

- **CICIDS2017:** The dataset for CICIDS2017 encompasses realistic data that can be gathered over a period of time. The dataset also has a wide variety of different attacks, from DDoS attacks to brute-force authentication attempts to botnet and web-based attacks. It can be described as having approximately 80+ distinct statistical features we have extracted from the data [15].

CICIDS2017 is a very well-known data set in machine learning studies and research but is not designed to specifically represent the way that IoT devices would behave or interact with other devices [11].

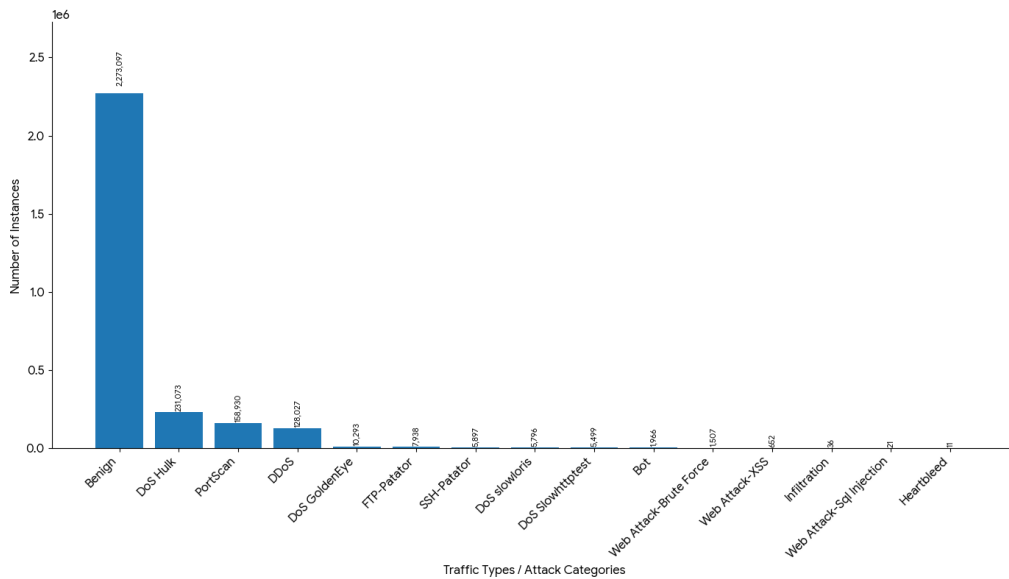


Figure 2.9: Number of Instances by Traffic Types in the CICIDS2017 Dataset.

- **Bot-IoT:** The Bot-IoT dataset is a simulation of IoT network environments and

mainly looks at attacks caused by botnets. Also included with this dataset are different types of attacks (DDoS, DoS, and data exfiltration) within an IoT network environment [13].

The Bot-IoT dataset is a widely adopted research resource for IoT security-related research; however, there are significant issues surrounding class imbalance and the dominance of attacks when looking at the Bot-IoT dataset [13].

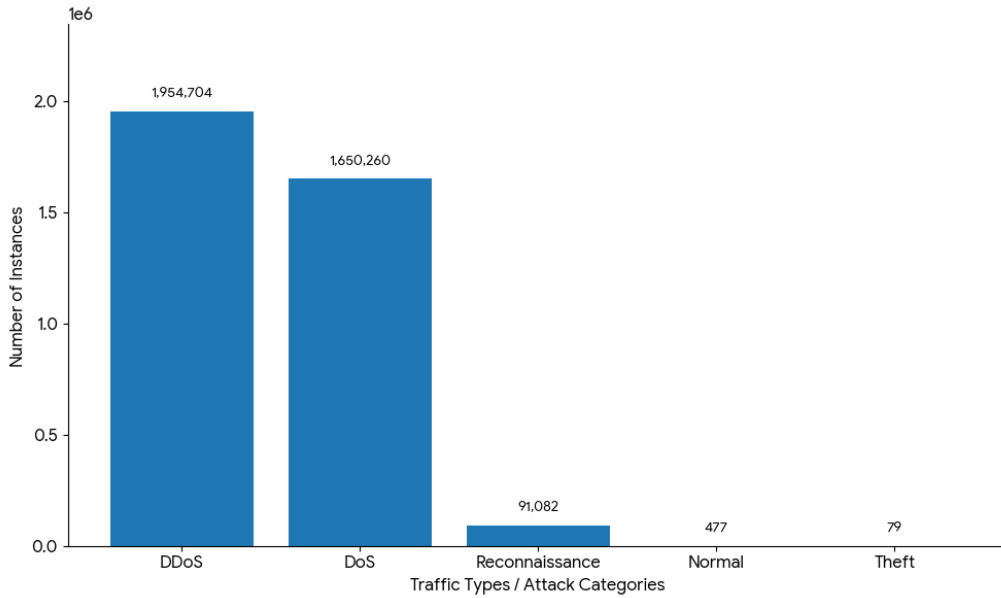


Figure 2.10: Number of Instances by Traffic Types in the BoT-IoT Dataset.

- IoT-23: IoT-23's dataset includes actual malware traffic that has been captured from infected IoT devices [16]. It shows a realistic portrayal of botnet behavior and how criminals communicate with those botnets.

IoT-23 only has a small range of attacks (i.e., botnet attacks) compared to broader IoT datasets, but they can still be used to train machine learning models [16].

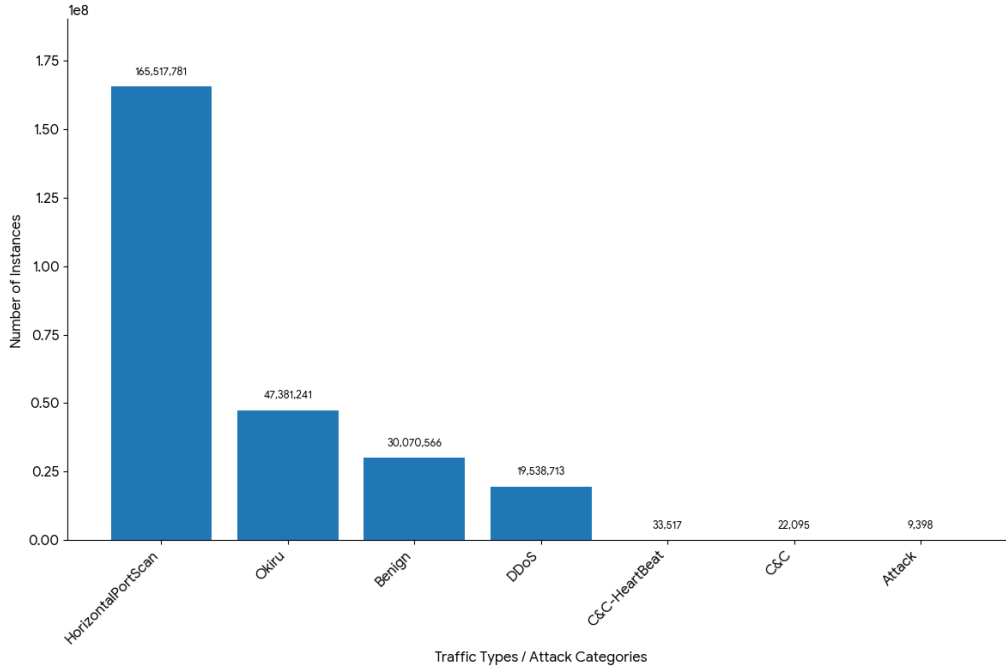


Figure 2.11: Number of Instances by Traffic Types in the IoT-23 Dataset.

- CICIoT2023: The CICIoT2023 collection was designed to create a new, extensive database that could be used as a benchmark for intrusion detection on IoT networks. To accomplish this task, it develops simulated environments representative of the wide variety of attack types currently being used against IoT networks, such as DDoS, Mirai-based, spoofing, reconnaissance and brute-force. [17] The CICIoT2023 Dataset is different from prior datasets for the following reasons:
 - Contains IoT-specific environments.
 - Contains modern botnet-driven threats.
 - Contains structured flow features that would work well with both Machine Learning (ML) and Deep Learning (DL) algorithms.
 - Contains an improved balance of normal traffic to attack traffic

These factors contribute to a more representative dataset of current IoT intrusion detection effort, and therefore, support its use in this research effort.

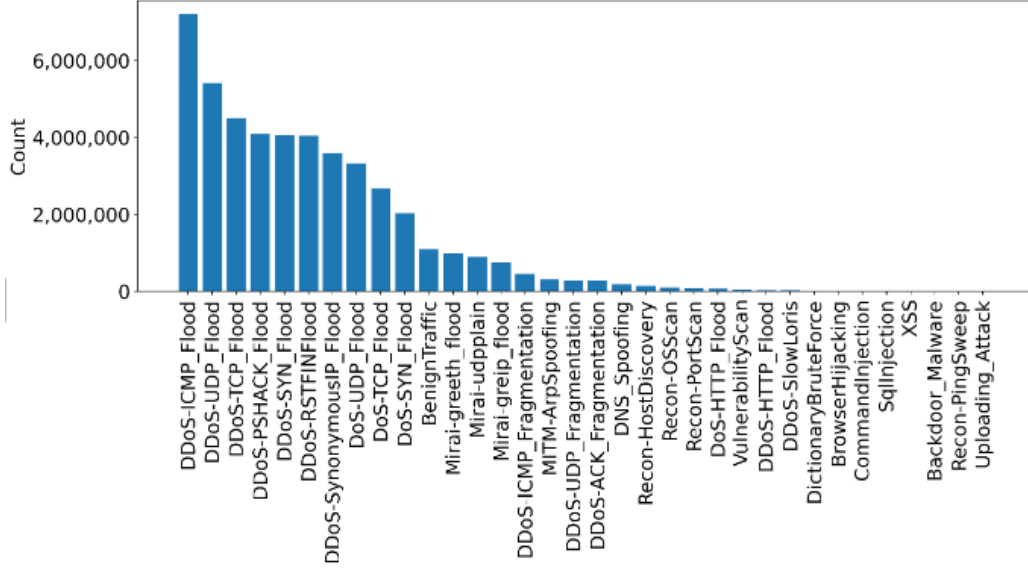


Figure 2.12: Number of rows for each scenario.

2.5.2 Comparison between datasets

compares intrusion detection datasets, showing a shift from traditional (KDD99, NSL-KDD) to IoT-focused datasets (Bot-IoT, IoT-23, CICIoT2023). Early datasets are outdated and lack IoT support, while newer ones address IoT environments but face issues like imbalance, limited diversity, or preprocessing needs.

Table 2.1: Comparison of Intrusion Detection Datasets

Dataset	Year	Environment	IoT	Features	Attack Types	Strengths	Limitations
KDD Cup 99	1999	Traditional	No	41	DoS, Probe, R2L, U2R	Benchmark dataset	Outdated, redundant
NSL-KDD	2009	Traditional	No	41	DoS, Probe, R2L, U2R	Reduced redundancy	Still outdated
UNSW-NB15	2015	Enterprise	No	49	9 categories	Modern attacks	Not IoT
CICIDS2017	2017	Enterprise	No	80+	DDoS, Botnet, Web	Realistic traffic	Large size
Bot-IoT	2018	IoT	Yes	Flow-based	DDoS, DoS	IoT-focused	Imbalanced
IoT-23	2020	IoT	Yes	PCAP	Malware, Botnet	Real data	Limited diversity
CICIoT2023	2023	IoT	Yes	Flow-based	Multiple attacks	Balanced, modern	Needs preprocessing

2.6 Conclusion

This chapter presented the fundamental concepts of IDS in IoT environments, including their architectures, detection methods, and security challenges such as heterogeneity,

scalability, and resource constraints. It also reviewed and compared traditional and IoT-specific intrusion detection datasets.

The analysis showed that traditional datasets are not well-suited for modern IoT scenarios, while recent datasets offer improved realism and diversity. Among them, CIIoT2023 was selected for this study due to its relevance to IoT environments and compatibility with machine learning approaches.

Overall, this chapter provided the theoretical basis for developing an intelligent IoT IDS framework. The next chapter will focus on the proposed methodology and implementation.

Chapter 3

Data Mining and Machine Learning Techniques for IOT Protection

Contents

3.1	Introduction	29
3.2	Data Mining	29
3.2.1	Definition of Data Mining	29
3.2.2	Data Mining Process	29
3.2.3	Data Mining Tasks	31
3.2.4	Role of Data Mining in Cybersecurity	32
3.3	Machine Learning	32
3.3.1	Definition of Machine Learning	32
3.3.2	Types of Machine Learning	32
3.3.3	Machine Learning Algorithms	33
3.3.4	Machine Learning Challenges	33
3.4	Deep Learning	34
3.4.1	Definition of Deep Learning	34
3.4.2	Deep Learning Models	34
3.4.3	Deep Learning Challenges	35
3.5	Related Works	36
3.6	Conclusion	37

3.1 Introduction

The rapid growth of IoT offers benefits like automation, cost savings, and real-time data access. However, the increasing prevalence of IoT also introduces major security risks. Because most IoT devices have limited resources and operate in dynamic environments, traditional security methods are ineffective against sophisticated cyberattacks.

To address IoT security challenges, new intelligent data analytics techniques such as data mining, machine learning, and deep learning are being used. These methods help organizations analyze network traffic patterns and automatically detect signs of cyberattacks. As a result, they enable advanced IDS that can identify both known and novel threats.

This chapter reviews data-driven methods for enhancing IoT security. It covers data mining concepts for security analysis, describes machine learning and deep learning techniques used in IDS, and concludes with a summary of prior research on anomaly-based IDS specifically for IoT environments.

3.2 Data Mining

3.2.1 Definition of Data Mining

Data mining is the process of analyzing large datasets to extract meaningful patterns and relationships. It is a key phase in the Knowledge Discovery in Databases (KDD) process, turning raw data into valuable information for decision-making and prediction. By combining machine learning, statistics, database systems, and AI, data mining automatically identifies hidden structures and trends in complex data. [18]

According to Han, Kamber, and Pei, data mining is an automated process that extracts implicit patterns and knowledge from large datasets like databases and data warehouses. It identifies correlations, trends, clusters, and anomalies, offering valuable insights for fields such as business intelligence, healthcare, and cybersecurity. [18]

3.2.2 Data Mining Process

Several data mining process models exist, including KDD, CRISP-DM, and SEMMA. While all aim to turn raw data into actionable knowledge through sequential steps, they differ in structure and focus. [19]

1. The KDD process: The KDD model is an early framework for data mining, involving steps like domain understanding, data selection, preprocessing, transformation, mining, and interpretation. It emphasizes explicitly discovering valuable patterns from large datasets.

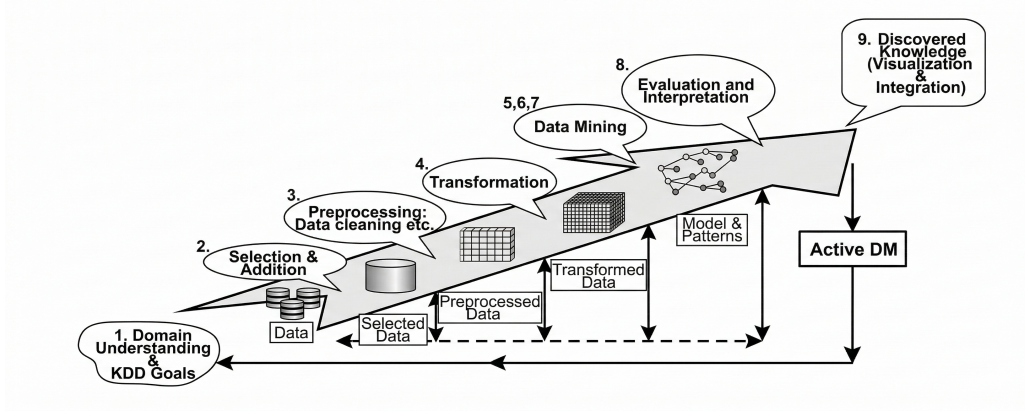


Figure 3.1: Knowledge Discovery Databases (KDD) Process Model

2. CRISP-DM process: Cross-Industry Standard Process for Data Mining is a structured methodology with six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. It focuses on project context, objectives, and real-world integration, making it highly practical for implementations.



Figure 3.2: CRISP-DM Process Model

3. SEMMA process: developed by SAS Institute, consists of five stages: sampling, exploration, modification, modeling, and assessment. While closely tied to SAS software tools, its core steps follow a progression similar to KDD and CRISP-DM.

KDD, CRISP-DM, and SEMMA share stages like data preparation, modeling, and evaluation. However, CRISP-DM uniquely includes business understanding and deployment

phases, making it better for complex real-world applications. KDD is more theoretical, while SEMMA is tool-oriented and less general.

Table 3.1: Data Mining Process Models Comparison

Process Models	KDD	CRISP-DM	SEMMA
No. of Steps	9	6	5
Name of Steps	Developing and Understanding	Business Understanding	———
	Creating a Target Data Set	Data Understanding	Sample
	Data Cleaning		Explore
	Data Transformation	Data Preparation	Modify
	Data Mining Tasks	Modeling	Model
	Mining Algorithms		
	Employing Algorithms		
	Interpreting Patterns	Evaluation	Assessment
	Using Knowledge	Deployment	———

3.2.3 Data Mining Tasks

Data mining tasks extract knowledge from large datasets to aid decision-making. Common tasks include classification, clustering, association rule mining, and anomaly detection, with the choice depending on the desired insights.

- **Classification:** is a supervised learning task that assigns data instances to predefined categories using labeled datasets. It is widely used in applications like intrusion detection to distinguish between normal and malicious network traffic.
- **Clustering:** is an unsupervised task that groups similar data items into clusters without predefined labels, aiming to maximize intra-cluster similarity and inter-cluster dissimilarity. It is used to reveal hidden patterns and unknown structures in large datasets.
- **Association Rule Mining:** identifies relationships between variables in large datasets, showing how certain items imply the presence of others. A common application is market basket analysis for understanding customer purchasing behavior.
- **Anomaly Detection:** identifies patterns that deviate from normal behavior, which may indicate errors, fraud, or cyberattacks. It is especially important in cybersecurity and IoT for detecting unknown attacks and suspicious activities.

3.2.4 Role of Data Mining in Cybersecurity

Data mining enhances cybersecurity by analyzing network data to uncover hidden patterns and detect anomalies. Techniques like classification, clustering, and anomaly detection help identify threats automatically. In IoT environments with massive data from connected devices, data mining enables efficient monitoring. Due to growing attack complexity, advanced methods like machine learning are needed to improve accuracy and adaptability.

3.3 Machine Learning

3.3.1 Definition of Machine Learning

Machine learning (ML) enables systems to learn patterns from data without explicit programming, using historical data to build predictive models. It is widely applied in cybersecurity and IoT, where it analyzes network traffic, detects unusual behavior, and identifies intrusions (including both known and unknown attacks) by learning normal and malicious activity patterns. [20]

3.3.2 Types of Machine Learning

Machine learning methods are divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning. Each category has unique methodologies for data analysis, prediction, and cybersecurity applications. [21]

1. **Supervised Learning:** uses labeled datasets to train models that learn input-output relationships, enabling predictions on new data. It is commonly applied to classification and regression tasks, including IDS that distinguish normal from malicious network traffic.
2. **Unsupervised Learning:** works with unlabeled data to uncover hidden patterns or structures using techniques like clustering and association rule mining. In cybersecurity, it helps detect anomalies in network traffic that may indicate potential threats.
3. **Reinforcement Learning:** involves an agent that learns through interaction with an environment, receiving rewards or penalties for its actions. Through trial and error, the agent gradually learns the optimal strategy to achieve a goal, improving its decision-making over time.

3.3.3 Machine Learning Algorithms

IDS rely on machine learning algorithms to automatically analyze network traffic and system behavior, classifying activity as normal or malicious. By learning from past event logs, ML models improve IDS accuracy in detecting cyberattacks, especially in complex environments like IoT networks. [22]

1. Decision Tree: is a supervised learning algorithm that uses a tree structure (nodes as features, branches as decision rules). It is interpretable and computationally efficient, making it suitable for real-time intrusion detection, where it classifies network traffic as normal or malicious. [18]
2. Random Forest: is an ensemble method combining multiple decision trees trained on random data samples, with predictions made by majority vote. It is robust to complex datasets and offers high classification performance, making it widely used in IDS. [23]
3. Support Vector Machine (SVM): is a supervised learning model that separates data using an optimal hyperplane, effective for high-dimensional datasets. It is widely used in IDS to distinguish normal from malicious network traffic. [18]
4. K-Nearest Neighbors (KNN): is a simple classifier that assigns new data points to the class most common among their closest neighbors. Although it can be resource-intensive on large datasets, KNN is effective in IDS for identifying abnormal behavior by comparing activity to known patterns. [18]
5. Naive Bayes: is a fast, probability-based classifier that assumes feature independence. It is computationally efficient on large datasets and is used in IDS to assess network packets for potential attacks at low cost. [18]
6. XGBoost: is a gradient boosting algorithm that builds decision trees sequentially, where each new tree focuses on correcting the errors of the previous ones. Compared to Random Forest, it is generally more effective in capturing complex patterns in structured data. [24]
7. LightGBM: is a gradient boosting algorithm that builds decision trees using a leaf-wise growth strategy, allowing it to learn complex patterns efficiently. It offers faster training and lower memory consumption than many traditional boosting methods, making it well-suited for large-scale intrusion detection tasks [25].

3.3.4 Machine Learning Challenges

While machine learning is effective for building IDS, applying it in cybersecurity presents challenges that affect every stage of the process, from data collection to model deployment.

The challenges of ML used in IDS are summarized below:

- Availability of cybersecurity datasets:

Datasets are crucial for training ML models, but many existing ones are outdated and fail to reflect modern cyberattack complexity, limiting model generalization in real-world environments.

- Data quality issues:

The quality of cybersecurity data affects model performance. Issues like inconsistency, noise, or imbalance (e.g., many normal events but few attacks) can bias learning. Proper preprocessing, cleaning, and balancing are essential to improve detection accuracy.

- Feature engineering complexity:

High dimensionality in network traffic data complicates feature selection and extraction. While techniques like PCA and LDA can reduce dimensionality, identifying the most relevant features remains difficult and directly impacts ML model performance.

- Adaptability to evolving threats:

Cyberattacks constantly evolve, making it challenging for static machine learning models to maintain high performance. Models must be continuously updated and retrained to detect new or unknown attacks; otherwise, their reliability in dynamic environments is significantly reduced.

3.4 Deep Learning

3.4.1 Definition of Deep Learning

Deep learning, a subfield of ML, uses multi-layered neural networks to automatically learn hierarchical features from large datasets, making it effective for high-dimensional data like network traffic. In cybersecurity and IoT, it enhances intrusion detection by processing large data volumes, identifying key features, and detecting new or unseen attacks, providing robust protection in complex environments.

3.4.2 Deep Learning Models

Deep learning architectures use multi-layered neural networks to automatically recognize patterns without manual feature extraction. Their ability to handle complex, high-dimensional, dynamic data like IoT traffic makes them effective for intrusion detection. Common models for IDS include ANN, CNN, RNN, LSTM, and Autoencoders.

1. Artificial Neural Networks (ANN): inspired by the human brain, consist of interconnected neurons that learn patterns through layered structures. In IDS, ANNs can classify network traffic and identify potential attacks by analyzing relationships between input features and target outcomes.
2. Convolutional Neural Networks (CNN): are deep learning models that automatically extract features from numeric data. Initially used for image processing, CNNs can also analyze structured network traffic, making them effective for detecting patterns and anomalies in large volumes of IoT data.
3. Recurrent Neural Networks (RNN): are designed for sequential data, such as time-series network traffic, and retain memory of past inputs. This long-term memory allows RNNs to detect patterns over time and identify attacks in IDS.
4. Long Short-Term Memory (LSTM): a type of RNN, overcomes short-term memory limitations using special memory cells. In IDS, LSTMs capture long-term network dependencies, improving the detection of complex and evolving attacks.
5. Autoencoders (AE): are unsupervised deep learning models that compress input data into a lower-dimensional representation and then reconstruct it. Large reconstruction errors can indicate abnormal or malicious behavior, making AEs useful for anomaly detection in cybersecurity.

3.4.3 Deep Learning Challenges

Deep learning has proven to be an effective technology for IDS, but its real-world application presents a number of different challenges, especially when looking at IoT or resource-constrained environments. Some of the key challenges of using deep learning in these situations include:

- Requirement for large-scale datasets:

Deep learning models perform better with large amounts of labeled data. In cybersecurity, however, privacy concerns and the limited availability of labeled attack data make it challenging to build effective deep learning models.

- High computational complexity:

Deep learning models require substantial computational power, memory, and storage—often using GPUs—to train effectively. This makes deploying them on IoT devices, which typically have limited resources, a significant challenge.

- Lack of interpretability:

Deep learning models are often thought of as "black boxes", making it difficult to understand how they come to decisions. This lack of transparency can lead to less trust in the system, especially for important cyber security applications.

- Handling imbalanced data:

Like machine learning, deep learning models are affected by class imbalance in cybersecurity datasets, making it difficult to detect rare but critical attacks. Techniques such as data augmentation or resampling are often needed to address these challenges.

3.5 Related Works

Recent studies show that artificial intelligence (AI) can enhance traditional IDS in IoT networks. With the growing number of connected devices and increasingly complex attacks, traditional systems often fall short, prompting organizations to adopt smarter, data-driven security solutions.

- Almaraz-Rivera et al. developed an intrusion detection system using the Bot-IoT dataset, addressing the class imbalance problem by selecting representative traffic flows for each attack category. They evaluated multiple models, including RF, DT, LSTM, MLP, GRU, RNN, and SVM, for both binary and multiclass classification. Their results showed very high performance, with accuracy exceeding 99%, where Decision Tree and MLP achieved the best results while maintaining real-time processing capabilities. [26]
- Alzaharani et al. investigated the use of several machine learning algorithms for detecting DDoS attacks using the CICDDoS2019 dataset. By applying feature selection techniques, they improved model performance, with Decision Tree and Random Forest achieving the highest accuracy of around 99%, while Decision Tree demonstrated better efficiency in terms of computation time. [27]
- Kumar et al. proposed a hybrid approach combining multiple machine learning algorithms, including KNN, RF, DT, and NB, alongside artificial neural networks. Using the Bot-IoT dataset and a custom IoT testbed, their results showed that hybrid models and KNN achieved high accuracy, highlighting the effectiveness of combining techniques for improved detection. [28]
- Patil et al. introduced a real-time intrusion detection system based on distributed machine learning using Spark Streaming and Kafka. Their system was evaluated on the CICDDoS2019 dataset, where Random Forest achieved the best performance among the tested models. [29]

- Jia et al. proposed FlowGuard, an intelligent edge-based defense mechanism that combines LSTM and CNN models for detecting and classifying DDoS attacks. Using the CICDDoS2019 dataset, the system achieved high detection accuracy, demonstrating the effectiveness of hybrid deep learning models in IoT environments. [30]
- Seifousadati et al. focused on feature selection and preprocessing to improve DDoS detection using machine learning techniques. Their study showed that boosting algorithms such as AdaBoost and XGBoost achieved excellent performance, reaching very high detection accuracy. [31]
- Becerra-Suarez et al. evaluated multiple machine learning and deep learning models using optimized feature selection and hyperparameter tuning. Their results indicated that Random Forest and XGBoost provided the best performance, achieving near-perfect accuracy in detecting DDoS attacks. [32]

The most recent research has shown the increasing significance of new data sets like CICIoT2023.

- Neto et al. introduced this dataset to simulate realistic large-scale IoT attacks and evaluated several machine learning models. Their results showed that models perform well on frequent attacks but struggle with less common ones, highlighting the challenge of class imbalance. [17]
- Abbas et al. explored deep learning approaches, including DNN, CNN, and RNN, for intrusion detection using the CICIoT2023 dataset. Their results showed that RNN achieved the best performance among the tested models. [33]
- Khan et al. evaluated several machine learning models for IoT-based anomaly detection, showing that Random Forest provided the best performance after applying preprocessing and feature selection techniques. [34]

Despite improvements, current IoT intrusion detection systems still face limitations such as data imbalance, high computational cost, and poor adaptability to evolving threats. This highlights the need for more configurable, scalable, and smarter IDS specifically designed for IoT networks.

3.6 Conclusion

This chapter provides an overview of the use of data mining, machine learning and deep learning techniques for protecting IoT with an emphasis on IDSs. It discusses the fundamental concepts and processes of data mining first, followed by a thorough examination

of the types, models, and challenges of machine learning approaches. Deep learning techniques are also discussed, focusing on their ability to handle complex and large-scale IoT data, yet also their limitations.

Additionally, recent related work was reviewed in order to demonstrate how intelligent techniques can successfully identify cyberattacks, while also identifying ongoing challenges such as imbalance in the data, computational complexity of the model, and adaptability to changing threats. Overall, this chapter provides the basis for developing efficient and robust IDS models designed specifically for IoT environments, which will be discussed further in subsequent chapters.

Chapter 4

Experimental Design and Performance Evaluation

Contents

4.1	Introduction	41
4.2	Dataset Presentation	41
4.2.1	CICIoT2023 Dataset Attributes and Feature Description	42
4.3	Execution Environment	43
4.4	Data Preprocessing	44
4.4.1	Data cleaning	44
4.4.2	Attack label grouping	45
4.4.3	Encoding of Categorical Labels	45
4.4.4	Handling Imbalanced Data	46
4.4.5	Data Normalization	46
4.4.6	Feature Selection	47
4.5	Model Application and Evaluation	48
4.5.1	Performance Evaluation Metrics	49
4.5.2	Machine Learning Models	50
4.5.3	Deep Learning Models	54
4.6	Global Model Comparison	59
4.6.1	Comparison Between Machine Learning and Deep Learning Models	59
4.6.2	Performance and Computational Efficiency Analysis	60
4.7	Model Interpretability Using SHAP	62
4.7.1	SHAP Results for Binary Classification	63
4.7.2	SHAP Results for Multi-Class Classification	65

4.7.3	Discussion of SHAP Findings	66
4.8	Conclusion	68

4.1 Introduction

This chapter presents the implementation and evaluation of the proposed IOT intrusion detection framework using ML and DL techniques.

The performance of such approaches depends heavily on data quality and preprocessing. Therefore, a preparation pipeline was developed using the CICIoT2023 dataset, including data cleaning, attack grouping, hybrid under-sampling and SMOTE resampling, feature scaling, and feature selection.

Although Chapter 03 reviewed a broader range of algorithms, the experimental study focuses on six models: Random Forest, XGBoost, LightGBM, 1D-CNN, DNN, and LSTM. These models were selected based on their strong performance in the literature, their suitability for large-scale supervised classification, and their relevance to structured IOT traffic data. Other algorithms (Decision Trees, SVM, KNN, Naive Bayes, RNN, and Autoencoders) were not retained due to limitations related to computational cost, model assumptions, learning paradigm, or redundancy with the selected approaches.

Two classification scenarios are considered: binary classification (Benign vs. Attack) and multi-class classification. The selected models are evaluated using accuracy, precision, recall, and F1-score, together with computational efficiency metrics such as training time, inference latency, and model size.

This chapter presents the complete experimental workflow and discusses the key design choices made to develop an accurate, efficient, and interpretable IDS for modern IOT environments.

4.2 Dataset Presentation

The *CICIoT2023* dataset, developed by the Canadian Institute for Cybersecurity, is used as the main data source in this study for IoT intrusion detection. It contains network traffic generated from real IoT devices, including both benign and malicious activities grouped into seven attack categories: DDoS, DoS, Mirai, reconnaissance, spoofing, brute force, and web-based attacks. Each record is represented using 46 network flow features.

Due to its large size and high class imbalance, a reduced and stratified version of the dataset obtained from Kaggle is used in this work. This version significantly reduces storage and computational requirements while preserving the essential characteristics of the original dataset, including features and attack categories. It also provides predefined training, validation, and test splits, which facilitates the experimental setup. However, class imbalance is still partially present and requires additional preprocessing steps.

Table 4.1 summarizes the main differences between the original CICIoT2023 dataset and the stratified version adopted in this study.

Table 4.1: Comparison of Original and Stratified Versions of the CICIoT2023 Dataset

Attribute	Original Dataset	Stratified Dataset
Total Instances	~47 million	~7.8 million (Train / Validation / Test)
Storage Size	~14 GB	~2 GB
Features per Record	46 flow-based features	46–47 features (including label)
Attack Categories	7 categories	7 categories
Attack Types	~33 attack types	~33 attack types
Class Distribution	Highly imbalanced	More manageable but still imbalanced
Data Splitting	Not predefined	Predefined (Train / Validation / Test)
Redundancy	High	Reduced

4.2.1 CICIoT2023 Dataset Attributes and Feature Description

To provide a clear understanding of the network traffic characteristics analyzed in this study, the 46 flow-based features of the CICIoT2023 dataset are categorized and described. These features capture various behavioral patterns of IoT network traffic, including packet dimensions, temporal attributes, protocol indicators, and statistical metrics. Table 4.2 outlines the structural composition of these attributes along with their operational meanings.

Table 4.2: Description and Categorization of the CICIOT2023 Dataset Features

Feature Group	Example Column Names	Description and Operational Meaning
Protocol & Framework Identifiers	ProtocolType, Datarate	Identifies the network protocol layer (e.g., TCP, UDP, ICMP) and the transmission rate utilized by the IoT devices during connection.
Packet Length Statistics	Min, Max, Avg, Std, Variance	Statistical measurements calculated over the lengths of transmitted packets to detect anomalies in payload sizes (common in volumetric attacks like DDoS).
Inter-Arrival Time (IAT)	IAT_Min, IAT_Max, IAT_Avg, IAT_Std	Measures the time intervals between consecutive packet arrivals. Malicious automated traffic (bots) often exhibits distinct deterministic time patterns compared to human or regular IoT traffic.
TCP Flag Counters	FIN_Flag_Count, SYN_Flag_Count, RST_Flag_Count, ACK_Flag_Count	Tracks the frequency of specific control bits in the TCP header. Sudden spikes in flags like SYN or RST strongly indicate scanning activities or flooding attacks.
Traffic Rates & Flow Control	Rate, Srate, Drate	Captures packet transmission speeds from source to destination, helping identify sudden traffic bursts or brute-force speed intervals.
Network Flow Geometry	Magnitude, Radius, Covariance, Weight	Advanced statistical and geometric abstractions introduced by the CIC framework to model multi-dimensional traffic flow behaviors and variances over a temporal window.
Target Output	label	The ground-truth classification target containing either discrete malicious/benign flags (Binary) or the specific 33 attack classifications (Multi-class).

4.3 Execution Environment

In this work, we conducted all experiments using Google Colaboratory (Colab), a cloud-based Jupyter notebook environment that provides access to computational resources such as CPUs, GPUs, and RAM. This choice allowed us to run large-scale experiments efficiently without being limited by local hardware constraints. It also ensured easier reproducibility and flexibility during the experimentation process.

For initial data exploration and basic preprocessing, we used a local machine (Toshiba TECRA Z40-B laptop) equipped with an Intel Core i5-5300U processor, 8 GB of RAM, and an SSD for storage, running Windows 11. However, due to the size of the dataset

and the computational requirements of the models, the local machine was not sufficient for full training and evaluation.

Therefore, we relied primarily on Google Colab to perform all major tasks, including data preprocessing, model training, and evaluation. The experiments were executed using Colab’s standard runtime environment, which typically provides an Intel Xeon processor, around 12 GB of RAM, and access to a GPU (NVIDIA Tesla T4) when available. This environment offered a good balance between performance and accessibility, enabling us to handle the dataset efficiently and train models within a reasonable time.

4.4 Data Preprocessing

In our work, the main preprocessing steps are summarized as follows:

- Data cleaning
- Attack label grouping
- Encoding of categorical labels
- Handling class imbalance using hybrid resampling
- Data normalization
- Feature selection

All preprocessing steps were applied only to the training data when required (e.g., resampling), while validation and test sets were kept unchanged to ensure a fair evaluation.

4.4.1 Data cleaning

The data cleaning process was carried out in a structured manner to ensure the quality and consistency of the dataset before applying machine learning models. Since the CICIoT2023 dataset contains a large number of records, even small inconsistencies can negatively impact the model performance.

We started by loading the training, validation, and test datasets provided in the Kaggle version. During this step, we verified the consistency of column names and ensured that the label column was correctly formatted across all files.

Next, we handled invalid values by replacing infinite values with missing values (NaN), followed by removing all rows containing missing data. This approach was chosen to avoid introducing noise that could result from imputation techniques.

We also removed duplicate records to eliminate redundancy in the dataset. This step helps prevent the model from learning repeated patterns that could bias the training process.

In addition, we standardized the label format by converting all labels to uppercase and removing unnecessary spaces. This ensures consistency when grouping attack types in later steps.

This cleaning process significantly improved the quality of the dataset and ensured that the data was consistent, reliable, and ready for the subsequent preprocessing steps.

4.4.2 Attack label grouping

The original CICIoT2023 dataset contains a large number of attack labels representing different attack types. However, using all these labels directly can increase the complexity of the classification task and negatively affect model performance.

To address this, we grouped the original labels into a smaller number of meaningful categories based on their characteristics. Specifically, the attacks were grouped into seven main classes: DDoS, DoS, Mirai, Spoofing, Reconnaissance, Web-based, and Brute Force, in addition to the Benign class.

This grouping was implemented using a rule-based function that maps each original label to its corresponding category. Labels that did not match any of these categories were assigned to an “Other” class and subsequently removed from the dataset.

This step reduces complexity while preserving the essential differences between attack types, making the classification task more manageable for the models.

4.4.3 Encoding of Categorical Labels

Since machine learning and deep learning algorithms perform mathematical operations during training and optimization, the categorical target labels must be represented numerically. Therefore, the target attribute, represented by the *label* column, was converted into numerical values using the *LabelEncoder* technique.

Each attack category was assigned a unique integer value. To ensure consistency and avoid discrepancies during prediction and evaluation, the same label mapping was used for the training, validation, and test sets. To prevent data leakage, the encoder was fitted only on the training set, while the validation and test sets were transformed using the mapping learned from the training data.

In addition to the multi-class setting, a binary classification scenario was also considered. In this case, all attack categories were grouped into a single class labeled as 1 (Attack), whereas normal traffic was assigned the label 0 (Benign). This enabled the evaluation of the proposed models in both binary and multi-class classification tasks.

4.4.4 Handling Imbalanced Data

During our analysis of the dataset, we noticed a clear imbalance between classes. Some attack types contain a very large number of samples, while others are much less represented. This imbalance can affect the learning process, as models tend to favor majority classes and may fail to correctly detect minority attacks.

To address this issue, we applied a hybrid resampling strategy on the training data. This approach combines under-sampling and over-sampling techniques. Under-sampling reduces the number of samples in majority classes, while over-sampling increases the size of minority classes. In our case, we used SMOTE, which generates new synthetic samples based on existing data rather than simply duplicating them [35].

For the multi-class scenario, we adopted a moderate strategy. Classes with a large number of samples were reduced to a maximum of 100,000 instances using random under-sampling. At the same time, smaller classes were increased to 10,000 instances using SMOTE. This approach avoids forcing all classes to have the same size, which helps preserve the natural distribution of the data.

We initially experimented with more aggressive balancing strategies; however, they led to overfitting in some cases. Therefore, a moderate approach was adopted to maintain a balance between class representation and data realism.

For the binary classification scenario (Benign vs. Attack), we applied only under-sampling. In this case, reducing the majority class was sufficient to obtain a balanced dataset without introducing synthetic data.

Finally, it is important to note that all resampling operations were applied only to the training data. The validation and test sets were kept unchanged to ensure a fair and realistic evaluation.

Overall, this approach allowed us to improve class balance while maintaining meaningful data patterns for model training.

4.4.5 Data Normalization

After balancing the dataset, we applied feature scaling to ensure that all input variables are on a comparable scale. The attributes concerned with this procedure encompass all 46 numerical network flow features (such as packet length statistics, traffic rates, and temporal attributes), while the pre-encoded target `label` column was strictly excluded from this scaling process. This step is critical because many machine learning and deep learning algorithms are highly sensitive to the absolute magnitude of feature values, which can heavily affect both training stability and optimization performance.

For the multi-class classification scenario, we implemented the `RobustScaler`. This method scales the 46 numerical features using the median and the interquartile range (IQR). This specific approach was chosen because network traffic datasets often contain

extreme variance and bursts of traffic; the `RobustScaler` ensures that outliers—which are highly common in massive malicious traffic streams—do not distort the feature representation.

For the binary classification scenario, we used the `MinMaxScaler` to normalize the 46 numerical attributes into a fixed bound between 0 and 1. This linear transformation helps improve convergence rates during model optimization and ensures a uniform baseline range across all independent variables.

The scaling configurations were fitted exclusively on the training data partitioning of these 46 features and then used to transform the validation and test sets. This methodology strictly prevents any form of data leakage and ensures a mathematically rigorous and fair evaluation of the models.

Overall, normalization significantly enhanced the computational stability and convergence efficiency of the learning process across the independent feature space.

4.4.6 Feature Selection

After normalization, we applied feature selection to reduce the number of input variables and keep only the most relevant ones for the learning process. This step helps improve model performance, reduce computational cost, and avoid unnecessary complexity.

We used a tree-based approach based on the Random Forest algorithm to estimate feature importance. This method assigns an importance score to each feature based on its contribution to the classification task.

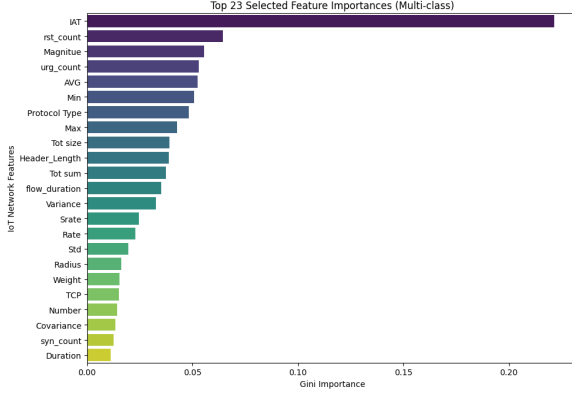
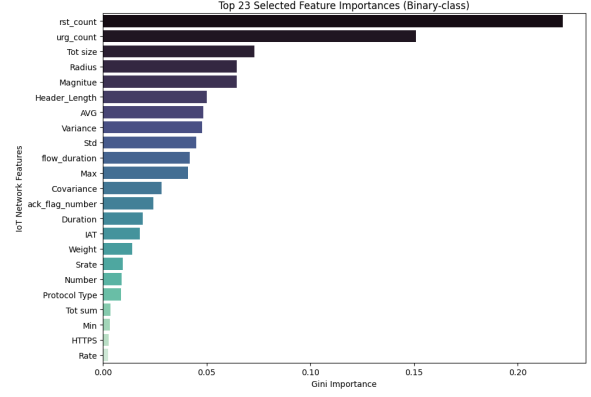
For both multi-class and binary scenarios, we trained a Random Forest model on the training data and used the `SelectFromModel` technique to select the most important features. Instead of using a very strict threshold, we adopted a relaxed strategy by setting the threshold to the median importance value.

We observed that using stricter thresholds significantly reduced the number of selected features, which negatively affected model performance. Therefore, a more flexible threshold was preferred in order to retain enough informative features while still removing less useful ones.

This choice helps avoid losing potentially important information, especially since network traffic data can contain subtle patterns that are useful for distinguishing between different types of attacks.

The same feature selection process was applied consistently to the training, validation, and test sets to ensure compatibility during model training and evaluation.

In addition, feature importance scores were analyzed and visualized to better understand which features contribute the most to the classification task.

**Figure 4.1:** Selected Features (Multi-class)**Figure 4.2:** Selected Features (Binary)

Following the feature selection procedure, exactly 23 features were retained for both the binary and multi-class classification scenarios (as illustrated in Figure 4.1 and Figure 4.2).

Overall, this step allowed us to reduce the dimensionality of the dataset while preserving the most informative features for both classification scenarios.

4.5 Model Application and Evaluation

After completing the data preprocessing phase, we move to the next step, which is the implementation and evaluation of different machine learning and deep learning models.

In this work, we consider six models in total: three from machine learning and three from deep learning. For the machine learning part, we use Random Forest (RF), XGBoost, and LightGBM, as these models are known to perform well on structured data such as network traffic. For the deep learning part, we implement a Deep Neural Network (DNN), a 1D Convolutional Neural Network (1D-CNN), and a Long Short-Term Memory (LSTM) model to capture more complex patterns in the data.

To evaluate the performance of these models, we consider two classification scenarios: binary classification and multi-class classification. In the multi-class case, the models are trained to distinguish between benign traffic and seven attack categories, namely DDoS, DoS, Mirai, Spoofing, Reconnaissance, Web-based, and Brute Force. In the binary case, all attack types are grouped into a single class (Attack), and the models are trained to distinguish between benign and malicious traffic.

This approach allows us to evaluate how well each model performs in both simple and more detailed classification tasks. It also helps us understand their ability to generalize to unseen IoT network traffic.

Finally, by comparing machine learning models with deep learning models under the same conditions, we aim to identify which approach is more suitable for intrusion detection in IoT environments, especially in scenarios where computational resources are limited.

4.5.1 Performance Evaluation Metrics

To evaluate the performance of the implemented models and compare their effectiveness, we use several evaluation metrics derived from the confusion matrix. These metrics provide a clear and detailed understanding of how well each model performs on the test data.

4.5.1.1 Confusion Matrix

The confusion matrix is a common tool used to evaluate classification models. It summarizes the predictions of a model by comparing them with the actual labels, allowing us to identify both correct and incorrect classifications.

For the binary classification case (Benign vs. Attack), the confusion matrix consists of four main components:

- TP (True Positives): Malicious instances correctly classified as attacks.
- TN (True Negatives): Benign instances correctly classified as normal traffic.
- FP (False Positives): Benign instances incorrectly classified as attacks (false alarms).
- FN (False Negatives): Malicious instances incorrectly classified as benign (missed attacks).

For the multi-class classification scenario, the confusion matrix is extended to multiple classes, where each row represents the actual class and each column represents the predicted class. This allows us to analyze how well the model distinguishes between different attack categories.

4.5.1.2 Evaluation Metrics

Based on the confusion matrix, we compute the following metrics:

- Accuracy: The proportion of correctly classified instances among all samples.

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

- Precision: Measures how many of the predicted attacks are actually malicious.

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

- Recall (Sensitivity): Measures the ability of the model to correctly detect attack instances.

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

- F1-Score: The harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

For the multi-class case, these metrics are computed for each class and then averaged using macro or weighted averaging to provide an overall evaluation of model performance.

In addition to accuracy, we focus particularly on precision and recall, as they are more informative in intrusion detection where false alarms and missed attacks have different impacts.

These metrics are computed on the test set to evaluate how well the models generalize to unseen data.

4.5.2 Machine Learning Models

4.5.2.1 Random Forest

In this work, we applied Random Forest for both binary and multi-class classification. The model was trained using the preprocessed dataset described earlier. The main hyperparameters used in our implementation are summarized in Table below.

Table 4.3: Random Forest Hyperparameters

Parameter	Binary	Multi-class
n_estimators	100	100
max_depth	30	35
max_features	-	sqrt
min_samples_split	5	5
min_samples_leaf	2	-
random_state	42	42

The performance of the Random Forest model is evaluated using the metrics defined earlier, namely accuracy, precision, recall, and F1-score. These metrics are computed on the test set for both binary and multi-class classification tasks in order to assess the model’s ability to generalize to unseen data. The obtained results are summarized in the table below.

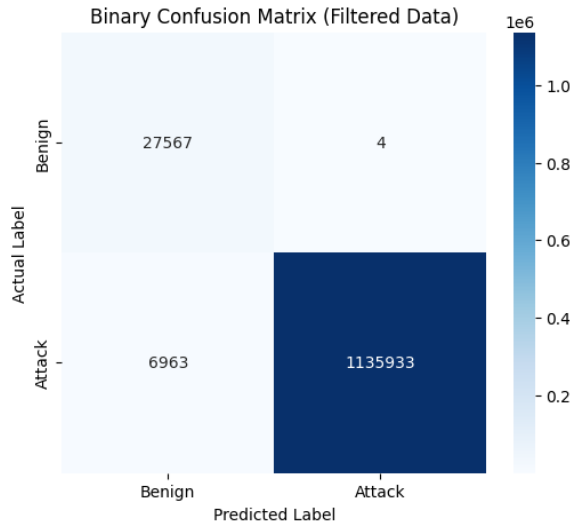
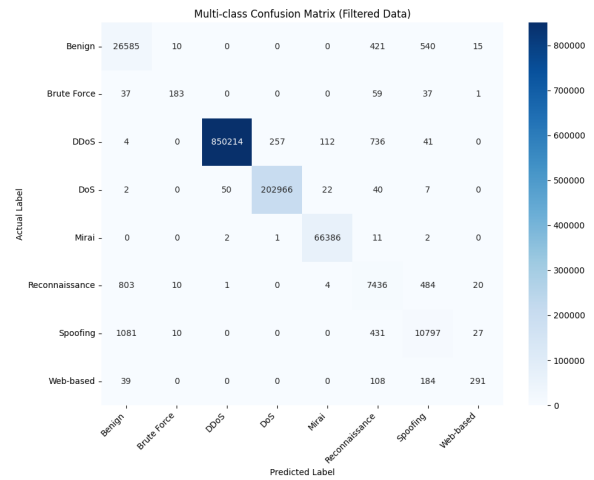
Table 4.4: Overall Performance: Binary vs. Multi-class Random Forest

Metric	Binary (Benign vs Attack)	Multi-class (weighted avg)
Accuracy	99.40%	99.52%
Precision	99.52%	99.52%
Recall	99.40%	99.52%
F1-score	99.43%	99.51%

Table 4.5: Multi-class Classification Report per Attack Type

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	93.11	96.42	94.74	27,571
Brute Force	85.91	57.72	69.05	317
DDoS	99.99	99.86	99.92	851,364
DoS	99.87	99.94	99.90	203,087
Mirai	99.79	99.97	99.88	66,402
Reconnaissance	80.45	84.90	82.62	8,758
Spoofing	89.29	87.45	88.36	12,346
Web-based	82.20	46.78	59.63	622

The confusion matrices obtained for both binary and multi-class classification tasks are presented in Figure 4.3 and Figure 4.4. These matrices provide a detailed representation of the model's predictions by comparing the actual and predicted labels. While the binary confusion matrix highlights the model's ability to distinguish between benign and attack traffic, the multi-class confusion matrix offers deeper insight into its performance across different attack categories.

**Figure 4.3:** Binary Confusion Matrix (Random forest)**Figure 4.4:** Multi-class Confusion Matrix (Random Forest)

4.5.2.2 XGBoost

As in RF, we applied XGBoost to both binary and multi-class classification tasks using the same preprocessed dataset. The main hyperparameters used are summarized in Table 4.6.

Table 4.6: XGBoost Hyperparameters

Parameter	Binary	Multi-class
n_estimators	100	100
max_depth	10	10
learning_rate	0.1	0.1
subsample	1.0	1.0
colsample_bytree	1.0	1.0
random_state	42	42

The performance of the model is evaluated using accuracy, precision, recall, and F1-score on the test set. The overall results are presented in Table 4.7.

Table 4.7: Overall Performance: Binary vs. Multi-class XGBoost

Metric	Binary (Benign vs Attack)	Multi-class
Accuracy	99.38%	99.57%
Precision	99.51%	99.57%
Recall	99.38%	99.57%
F1-score	99.41%	99.57%

Table 4.8: Multi-class Classification Report per Attack Type (XGBoost)

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	92.69	96.35	94.48	27571
Brute Force	81.44	56.78	66.91	317
DDoS	99.99	99.95	99.97	851364
DoS	99.90	99.96	99.93	203087
Mirai	99.92	99.99	99.96	66402
Reconnaissance	86.11	85.17	85.64	8758
Spoofing	90.25	86.45	88.31	12346
Web-based	68.53	49.05	57.17	622

The confusion matrices for both binary and multi-class classification are shown in Figure 4.5 and Figure 4.6, providing a clear view of the model’s prediction behavior across different classes.

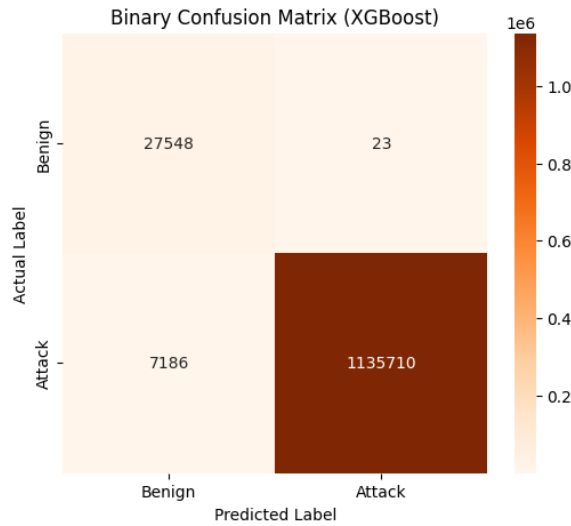


Figure 4.5: Binary Confusion Matrix (XGBoost)

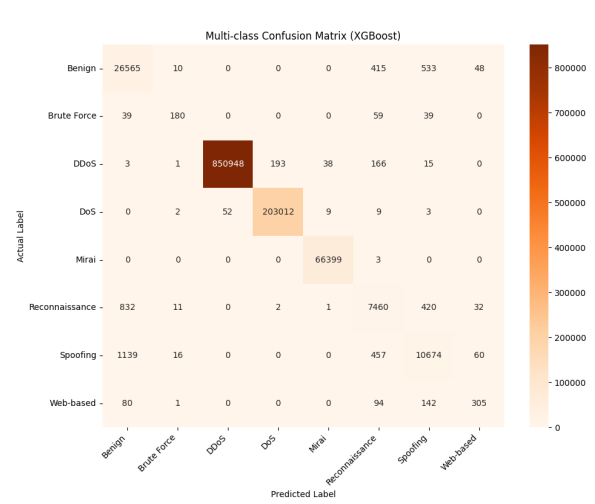


Figure 4.6: Multi-class Confusion Matrix (XGBoost)

4.5.2.3 LightGBM

LightGBM is a gradient boosting algorithm designed for high efficiency and fast training. It uses a leaf-wise tree growth strategy, which allows it to handle large-scale data and complex patterns more effectively compared to traditional boosting methods. The main hyperparameters used are summarized in Table 4.9.

Table 4.9: LightGBM Hyperparameters

Parameter	Binary	Multi-class
n_estimators	100	100
learning_rate	0.1	0.1
max_depth	-	-
num_leaves	default	default
random_state	42	42

The model performance is evaluated using accuracy, precision, recall, and F1-score on the test set. The overall results are presented in Table 4.10.

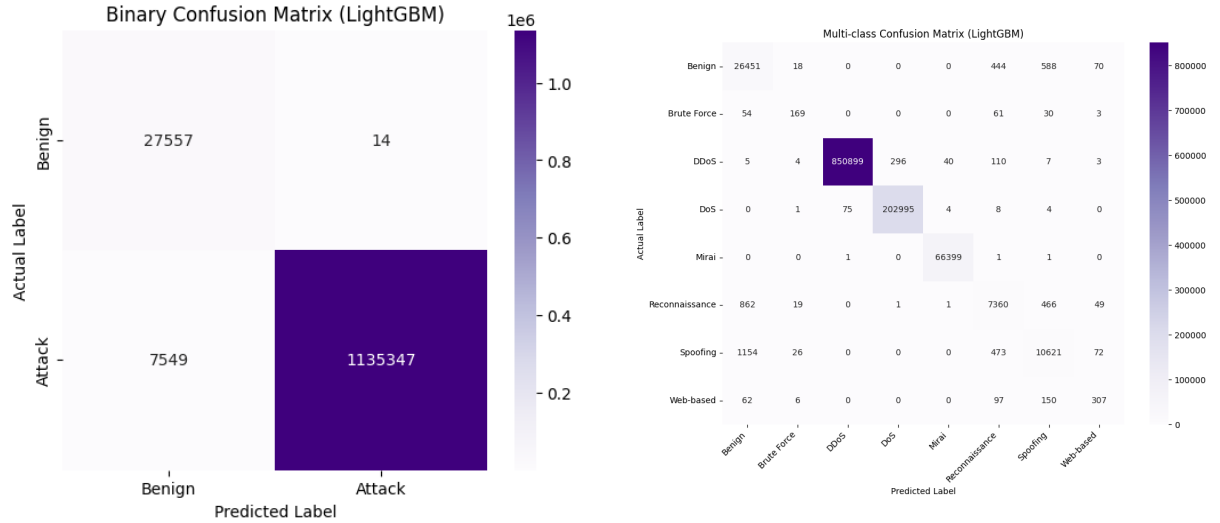
Table 4.10: Overall Performance: Binary vs. Multi-class LightGBM

Metric	Binary (Benign vs Attack)	Multi-class
Accuracy	99.35%	99.55%
Precision	99.49%	99.54%
Recall	99.35%	99.55%
F1-score	99.39%	99.54%

Table 4.11: Multi-class Classification Report per Attack Type (LightGBM)

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	92.52	95.93	94.20	27571
Brute Force	69.54	53.31	60.35	317
DDoS	99.99	99.94	99.96	851364
DoS	99.85	99.95	99.90	203087
Mirai	99.93	99.99	99.96	66402
Reconnaissance	86.04	84.03	85.02	8758
Spoofing	89.50	86.02	87.73	12346
Web-based	60.91	49.35	54.52	622

The confusion matrices for both binary and multi-class classification are shown in Figure 4.7 and Figure 4.8, providing a clear view of the model’s predictions across different traffic categories.

**Figure 4.7:** Binary Confusion Matrix (LightGBM)**Figure 4.8:** Multi-class Confusion Matrix (LightGBM)

4.5.3 Deep Learning Models

4.5.3.1 1D-CNN

To further explore deep learning approaches, we implemented a 1D Convolutional Neural Network (1D-CNN), which is well-suited for handling structured sequential data such as network traffic features. Unlike traditional models, CNNs can automatically extract local patterns from the input data, which may improve the detection of complex attack behaviors.

In our work, we applied a relatively shallow 1D-CNN architecture for both binary and multi-class classification. The model consists of convolutional layers followed by pooling and fully connected layers, allowing it to progressively learn higher-level feature represen-

tations. The same general architecture is used for both tasks, with slight adjustments in the output layer depending on the classification type.

The main hyperparameters and architectural choices are summarized in the table below.

Table 4.12: 1D-CNN Model Configuration

Parameter	Binary	Multi-class
Conv Layers	2	2
Filters	32, 64	32, 64
Kernel Size	3	3
Pooling	MaxPooling	MaxPooling
Dense Layers	64 units	128 units
Dropout	0.1	0.1
Optimizer	Adam	Adam
Loss Function	Binary Crossentropy	Sparse Categorical Crossentropy
Batch Size	256	256
Epochs	up to 20 (Early Stopping)	up to 20 (Early Stopping)

The performance of the 1D-CNN model is evaluated using the same metrics defined earlier, namely accuracy, precision, recall, and F1-score. These metrics are computed on the test set for both binary and multi-class classification in order to assess the model's generalization capability.

The obtained results are presented in the following table.

Table 4.13: Overall Performance: Binary vs. Multi-class 1D-CNN

Metric	Binary (Benign vs Attack)	Multi-class (weighted avg)
Accuracy	98.94%	99.04%
Precision	99.26%	99.06%
Recall	98.94%	99.04%
F1-score	99.04%	99.03%

Table 4.14: Multi-class Classification Report per Attack Type (1D-CNN)

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	83.69	93.24	88.20	27,571
Brute Force	13.21	31.54	18.62	317
DDoS	99.98	99.84	99.91	851,364
DoS	99.74	99.91	99.83	203,087
Mirai	99.97	99.92	99.95	66,402
Reconnaissance	68.36	63.47	65.82	8,758
Spoofing	81.99	68.32	74.53	12,346
Web-based	20.59	18.81	19.66	622

The confusion matrices for both classification scenarios are presented in Figure 4.9

and Figure 4.10. These matrices provide a more detailed view of the model’s predictions and highlight where misclassifications occur.

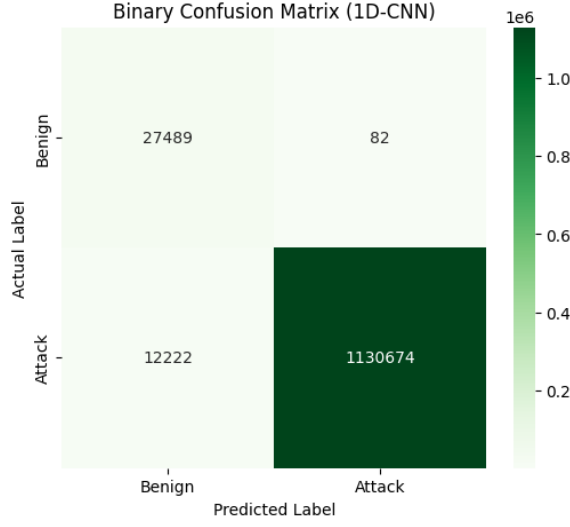


Figure 4.9: Binary Confusion Matrix (1D-CNN)

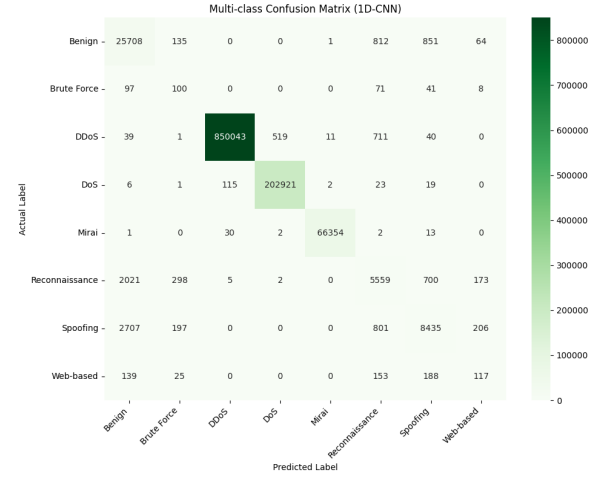


Figure 4.10: Multi-class Confusion Matrix (1D-CNN)

4.5.3.2 Deep Neural Network (DNN)

In addition to convolution-based models, we also implemented a Deep Neural Network (DNN), also referred to as a Multilayer Perceptron (MLP), to evaluate its ability to model complex relationships in structured IoT traffic data. Unlike CNNs, the DNN operates directly on flattened feature vectors, making it a suitable baseline for comparison with other deep learning approaches.

In our implementation, we designed a relatively simple architecture composed of fully connected layers combined with Batch Normalization and Dropout. These components help stabilize training and reduce overfitting, especially given the size and imbalance of the dataset. The model was trained using the Adam optimizer along with early stopping to avoid unnecessary training once convergence is reached. The main hyperparameters used are summarized in Table 4.15.

Table 4.15: DNN Hyperparameters

Parameter	Binary	Multi-class
Input Layer	Dense(64)	Dense(128)
Hidden Layers	Dense(32)	Dense(64)
Activation	ReLU	ReLU
Dropout	0.1	0.1
Batch Size	256	256
Epochs	20 (EarlyStopping)	20 (EarlyStopping)
Optimizer	Adam	Adam
Output Layer	Sigmoid	Softmax

The performance of the DNN model is evaluated using the same metrics defined earlier. The results obtained on the test set for both binary and multi-class classification are summarized in Table 4.16.

Table 4.16: Overall Performance: Binary vs. Multi-class DNN

Metric	Binary (Benign vs Attack)	Multi-class (weighted avg)
Accuracy	98.95%	81.15%
Precision	99.26%	89.68%
Recall	98.95%	81.15%
F1-score	99.04%	82.78%

To better understand the model's behavior across different attack types, a detailed multi-class classification report is presented in Table 4.17.

Table 4.17: Multi-class Classification Report per Attack Type (DNN)

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	79.45	89.15	84.02	27,571
Brute Force	5.46	2.20	3.14	317
DDoS	99.95	75.85	86.25	851,364
DoS	49.91	99.13	66.39	203,087
Mirai	92.65	98.73	95.59	66,402
Reconnaissance	65.25	51.11	57.32	8,758
Spoofing	65.91	65.85	65.88	12,346
Web-based	15.38	0.32	0.63	622

The confusion matrices obtained for both binary and multi-class classification tasks are presented in Figure 4.11 and Figure 4.12. These matrices provide a detailed visualization of the model's predictions by comparing the actual and predicted labels.

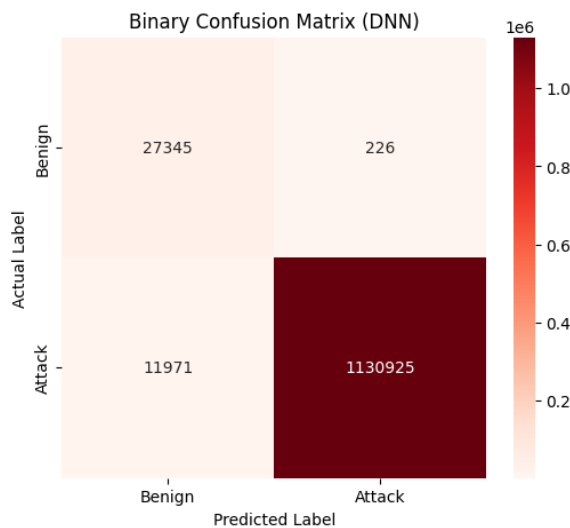


Figure 4.11: Binary Confusion Matrix (DNN)

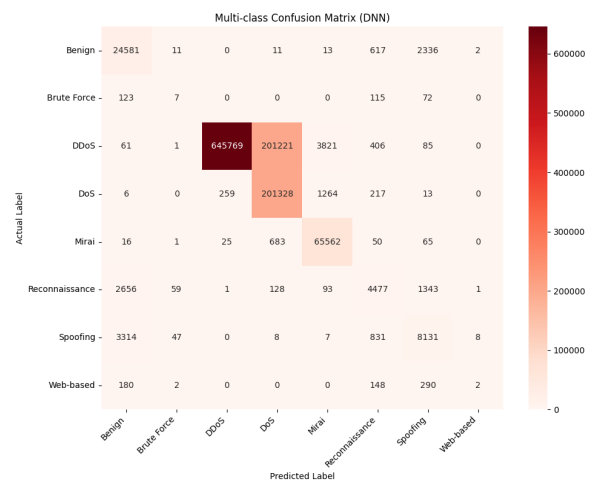


Figure 4.12: Multi-class Confusion Matrix (DNN)

4.5.3.3 Long Short-Term Memory (LSTM)

We apply LSTM for both binary and multi-class classification tasks using the preprocessed dataset. The main configuration of the model is summarized in the table below.

Table 4.18: LSTM Hyperparameters

Parameter	Binary	Multi-class
LSTM units	64	64
Dense layer	32	32
Dropout	0.1	0.1
Batch size	256	256
Epochs	20	20
Optimizer	Adam	Adam
Loss function	Binary crossentropy	Sparse categorical crossentropy

The performance of the LSTM model is evaluated using accuracy, precision, recall, and F1-score. These metrics are computed on the test set for both binary and multi-class classification to assess the model’s generalization capability.

Table 4.19: Overall Performance: Binary vs. Multi-class LSTM

Metric	Binary (Benign vs Attack)	Multi-class (weighted avg)
Accuracy	98.61%	97.47%
Precision	98.69%	97.65%
Recall	98.61%	97.47%
F1-score	98.65%	97.53%

Table 4.20: Multi-class Classification Report per Attack Type (LSTM)

Class	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	82.94	81.94	82.44	27,571
Brute Force	19.04	1.26	2.36	317
DDoS	99.94	98.31	99.12	851,364
DoS	94.36	99.34	96.78	203,087
Mirai	99.40	99.29	99.35	66,402
Reconnaissance	45.52	59.37	51.53	8,758
Spoofing	60.85	67.28	63.90	12,346
Web-based	8.62	6.75	7.57	622

The confusion matrices for both binary and multi-class classification are presented in Figure 4.13 and Figure 4.14. These figures provide a detailed view of the model’s predictions by comparing actual and predicted labels. While the binary matrix highlights the model’s ability to distinguish between benign and attack traffic, the multi-class matrix offers deeper insight into its performance across different attack categories.

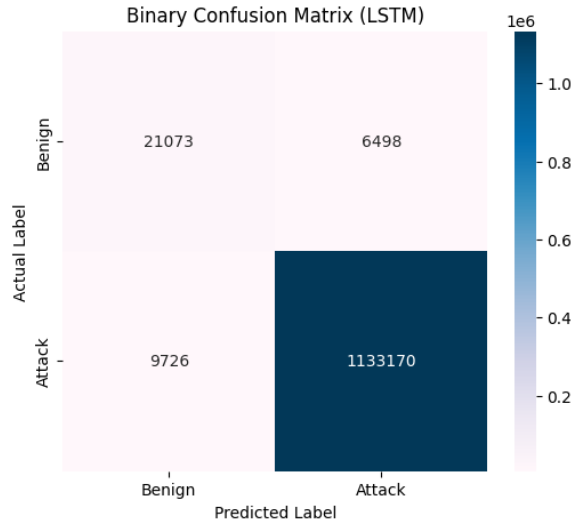


Figure 4.13: Binary Confusion Matrix (LSTM)

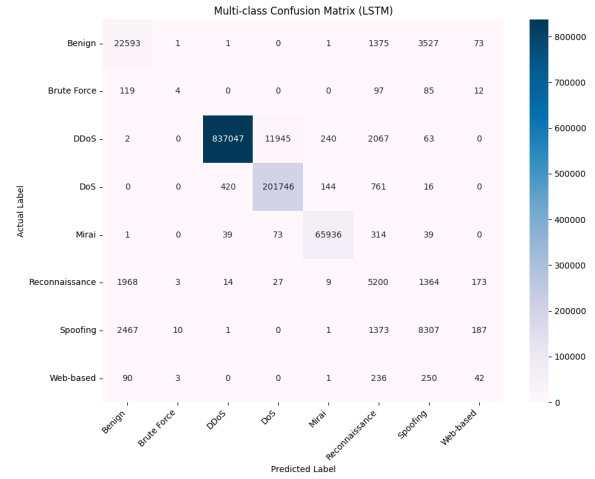


Figure 4.14: Multi-class Confusion Matrix (LSTM)

4.6 Global Model Comparison

4.6.1 Comparison Between Machine Learning and Deep Learning Models

To better evaluate the effectiveness of the proposed approaches, we conducted a comparative analysis between traditional machine learning models and deep learning models for both binary and multi-class classification tasks. This comparison aims not only to measure performance differences, but also to better understand the behavior of each model and explain the reasons behind the observed results.

In the binary classification scenario, we observed that all models achieved consistently high performance. The accuracy exceeded 98% for all approaches, with several models reaching over 99%. This indicates that distinguishing between benign and malicious traffic is relatively straightforward once appropriate preprocessing and class balancing techniques are applied. In this context, both machine learning and deep learning models are capable of learning the general patterns that separate normal traffic from attacks.

However, when moving to the multi-class classification task, the differences between the models become more apparent. We found that traditional machine learning models, including Random Forest, XGBoost, and LightGBM, consistently achieved very high accuracy (around 99%) while maintaining strong precision and recall across most attack categories. These models showed high robustness and stability, especially when dealing with imbalanced data. Their tree-based structure enables them to effectively capture complex relationships in tabular data and handle variations between majority and minority classes.

In contrast, deep learning models exhibited more variable performance. The 1D-CNN model achieved relatively competitive results, approaching those of machine learning models. This can be explained by its ability to extract local feature patterns and capture interactions between neighboring features. However, the DNN model showed a significant drop in performance in the multi-class scenario, suggesting difficulties in learning discriminative representations for multiple attack categories. Similarly, the LSTM model, although effective in binary classification, showed limitations in the multi-class case, particularly for minority classes such as Brute Force and Web-based attacks.

We believe that these differences can be explained by several factors. First, the dataset used in this work is highly imbalanced, which poses a major challenge for deep learning models. Unlike ensemble-based methods, deep neural networks generally require large amounts of well-balanced data to perform effectively. As a result, they tend to focus more on majority classes, leading to weaker performance on minority attacks.

Second, the nature of the dataset plays an important role. The CICIOT2023 dataset is structured and tabular, whereas deep learning models such as CNNs and LSTMs are typically more effective on spatial or sequential data (such as images or time series). In contrast, tree-based machine learning models are naturally well-suited for tabular data, which explains their strong performance in this context.

Finally, deep learning models are more sensitive to hyperparameter tuning and require careful architectural design. Despite applying normalization, feature selection, and balancing techniques, these models may still struggle to converge to optimal solutions within a limited training time.

Overall, we observe that machine learning models not only achieve higher and more stable performance in the multi-class scenario, but also adapt better to the characteristics of the dataset. Deep learning models, while capable of capturing complex patterns, do not provide a clear advantage in this particular case.

These findings suggest that the choice of model should not be based solely on accuracy, but also on the nature of the data and the requirements of the application. In particular, for structured network traffic data and imbalanced classification problems, traditional machine learning models remain a highly effective and reliable choice for IDS.

To further support these observations, a detailed comparison including both performance and computational efficiency metrics is presented in the next subsection.

4.6.2 Performance and Computational Efficiency Analysis

In addition to predictive performance, we conducted a comprehensive comparison of all models by considering both accuracy and computational efficiency. In real-world IDS, it is not sufficient to rely only on performance metrics. Instead, factors such as model size, training time, and inference speed must also be taken into account, especially in

resource-constrained environments such as IoT networks.

To provide a complete overview, Table 4.21 summarizes the main performance and efficiency metrics for all evaluated models in both binary and multi-class classification scenarios.

Table 4.21: Comprehensive Comparison of Models (Performance and Efficiency)

Model	Type	Task	Accuracy	Model Size (MB)	Training Time (s)	Inference (ms)
Random Forest	ML	Binary	99.40%	14.88	76.68	3.90
Random Forest	ML	Multi	99.52%	364.24	173.93	12.18
XGBoost	ML	Binary	99.38%	0.81	7.80	2.60
XGBoost	ML	Multi	99.57%	8.07	75.97	25.39
LightGBM	ML	Binary	99.35%	0.34	17.93	2.34
LightGBM	ML	Multi	99.55%	2.68	74.21	48.66
1D-CNN	DL	Binary	98.94%	0.35	446.12	53.96
1D-CNN	DL	Multi	99.04%	0.60	859.24	52.72
DNN	DL	Binary	98.95%	0.08	106.58	64.46
DNN	DL	Multi	81.15%	0.18	94.34	59.96
LSTM	DL	Binary	98.61%	0.26	2016.34	174.70
LSTM	DL	Multi	97.47%	0.26	1208.52	171.35

From Table 4.21, we can observe several important trends.

First, in terms of predictive performance, machine learning models consistently achieve the highest accuracy in both binary and multi-class classification tasks. XGBoost slightly outperforms the other models in the multi-class scenario, reaching the best overall accuracy, followed closely by Random Forest and LightGBM. These results confirm the strong suitability of ensemble-based methods for structured network traffic data.

Second, when considering model size, significant differences can be observed. Random Forest produces a very large model in the multi-class scenario (364 MB), which may limit its practical deployment despite its high performance. In contrast, XGBoost and LightGBM provide much smaller models (8.07 MB and 2.68 MB respectively), making them more suitable for real-world applications where memory constraints are important.

Third, regarding training time, machine learning models are generally more efficient than deep learning models. XGBoost and LightGBM require less than 80 seconds for training in the multi-class scenario, while Random Forest requires slightly more time. In comparison, deep learning models are significantly more expensive, with the LSTM requiring over 1200 seconds and the 1D-CNN around 850 seconds. This highlights the higher computational cost associated with deep architectures.

Fourth, inference speed is a critical factor for real-time intrusion detection. Machine learning models demonstrate faster inference, especially in the binary case, where LightGBM and XGBoost achieve the lowest latency. In contrast, deep learning models, particularly LSTM, show much slower inference speeds, making them less suitable for time-sensitive environments.

Finally, it is important to highlight that although deep learning models generally have

smaller model sizes, they do not provide a clear advantage in this context. The DNN model, in particular, shows a significant drop in performance in the multi-class scenario, indicating its inability to generalize effectively across multiple attack categories.

Based on this comprehensive analysis, we conclude that XGBoost provides the best overall trade-off between predictive performance and computational efficiency. It achieves the highest accuracy in the multi-class scenario while maintaining a relatively small model size and fast training time. LightGBM also represents a strong alternative, particularly in scenarios where memory efficiency is critical.

Overall, these results highlight that model selection should balance accuracy, efficiency, and deployment constraints. In resource-constrained environments such as IoT networks, machine learning models—especially XGBoost and LightGBM—emerge as the most practical and effective solutions.

4.7 Model Interpretability Using SHAP

In addition to evaluating predictive performance and computational efficiency, we aimed to better understand how the selected model makes its decisions. For this purpose, we applied SHAP (SHapley Additive exPlanations) [36], an explainable technique that provides both global and local interpretations of model predictions.

Grounded in cooperative game theory, **SHAP** quantifies each feature’s contribution by computing its average marginal effect across all possible subsets of the input features. Formally, the Shapley value ϕ_j for feature j is defined as:

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|! (|F| - |S| - 1)!}{|F|!} [f(S \cup \{j\}) - f(S)] \quad (4.1)$$

where F is the full set of features, S is any subset excluding feature j , and $f(S)$ is the model output using only the features in S . This formulation ensures that feature attributions are both theoretically sound and consistent across different prediction scenarios [36].

SHAP assigns an importance value to each feature based on its contribution to the final prediction. This allows us to identify which features influence the model the most and to verify whether the model relies on meaningful patterns.

In this work, we applied **SHAP** analysis only to the XGBoost model. This choice is justified by several reasons. First, XGBoost achieved the best overall performance in both binary and multi-class classification tasks, making it the most relevant model to interpret. Second, XGBoost is a tree-based model, and **SHAP** provides an optimized implementation (TreeExplainer) specifically designed for such models, ensuring fast and accurate explanations.

In contrast, applying **SHAP** to deep learning models such as DNN, CNN, or LSTM is computationally more expensive and less efficient, especially for large tabular datasets.

Moreover, since these models did not outperform XGBoost in our experiments, focusing on XGBoost provides more meaningful insights.

4.7.1 SHAP Results for Binary Classification

For the binary classification task, we computed SHAP values on a representative sample of the test data to identify the most influential features in distinguishing between benign and malicious traffic.

The SHAP summary bar plot (Figure 4.15) shows the global feature importance ranked by their average contribution. We observe that a small subset of features dominates the decision-making process, indicating that the model successfully identifies the most relevant network characteristics for detecting attacks.

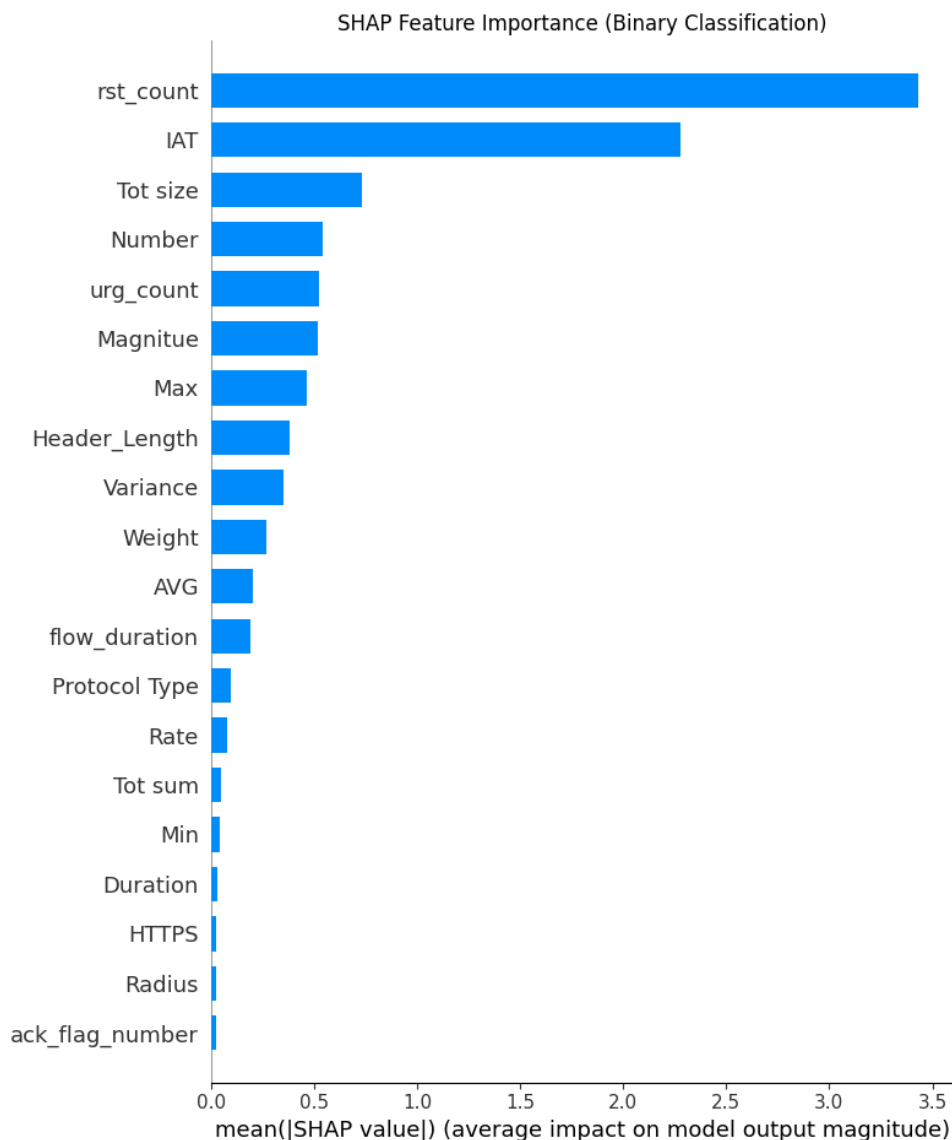


Figure 4.15: SHAP Feature Importance for Binary Classification

The detailed SHAP distribution plot (Figure 4.16) provides additional insights into

how each feature affects predictions. We observe that higher values of certain features strongly push the model toward the “attack” class, while lower values are associated with benign traffic. This confirms that the model learns meaningful and consistent patterns rather than relying on random correlations.

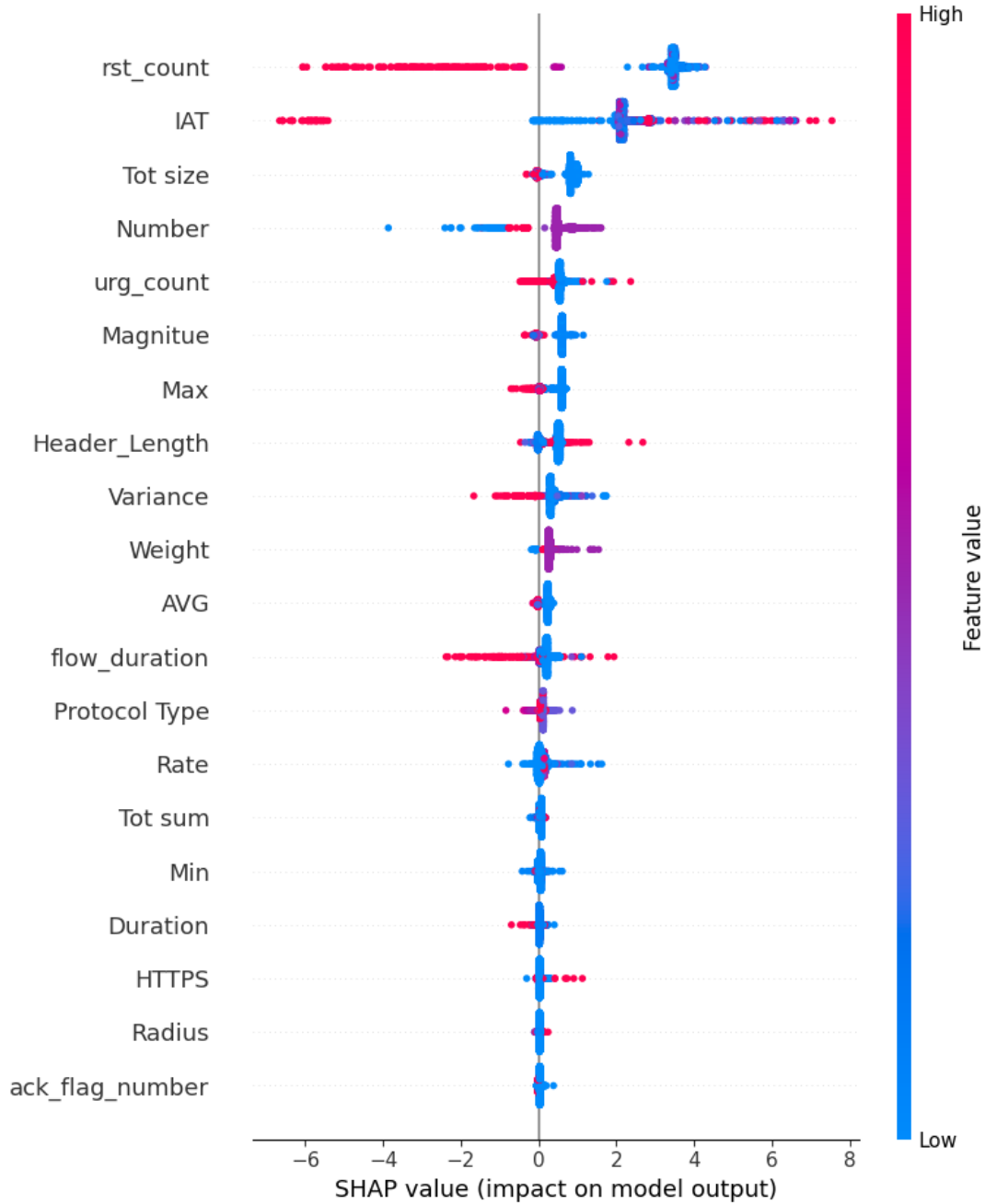


Figure 4.16: SHAP Summary Plot (Feature Impact Distribution) for Binary Classification

Overall, these results demonstrate that the XGBoost model makes reliable and interpretable decisions in the binary classification scenario.

4.7.2 SHAP Results for Multi-Class Classification

For the multi-class classification task, we performed a more detailed SHAP analysis to understand how features contribute to predicting different attack categories.

The global SHAP importance plot (Figure 4.17) highlights the most important features across all classes. We observe that the same key features identified in the binary task remain highly influential, which indicates consistency in the model’s behavior.

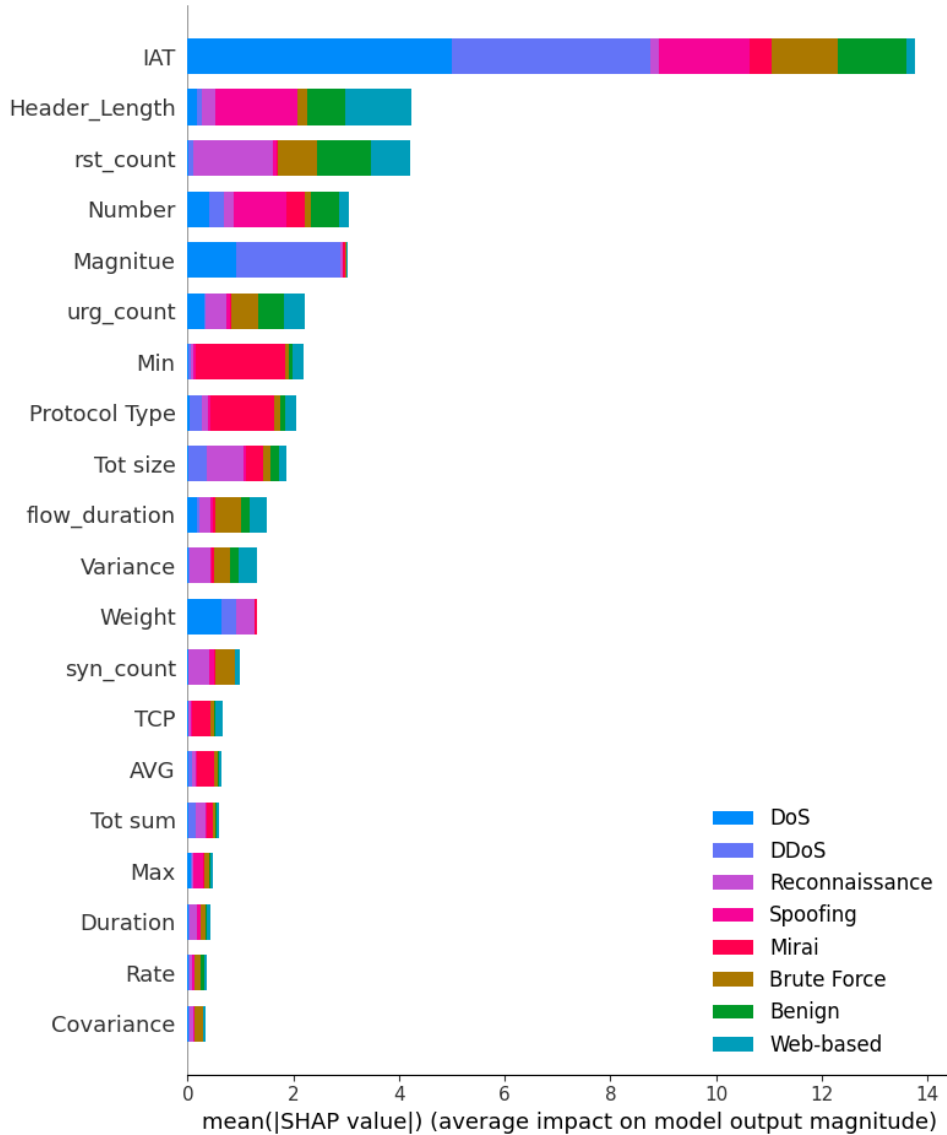


Figure 4.17: SHAP Global Feature Importance (Multi-Class Classification)

To further investigate class-specific behavior, we analyzed SHAP values for individual attack categories, such as DDoS (Figure 4.18). The results show that certain features have a strong positive impact on predicting this class, while others contribute negatively. This demonstrates that the model differentiates between attack types based on distinct feature patterns.

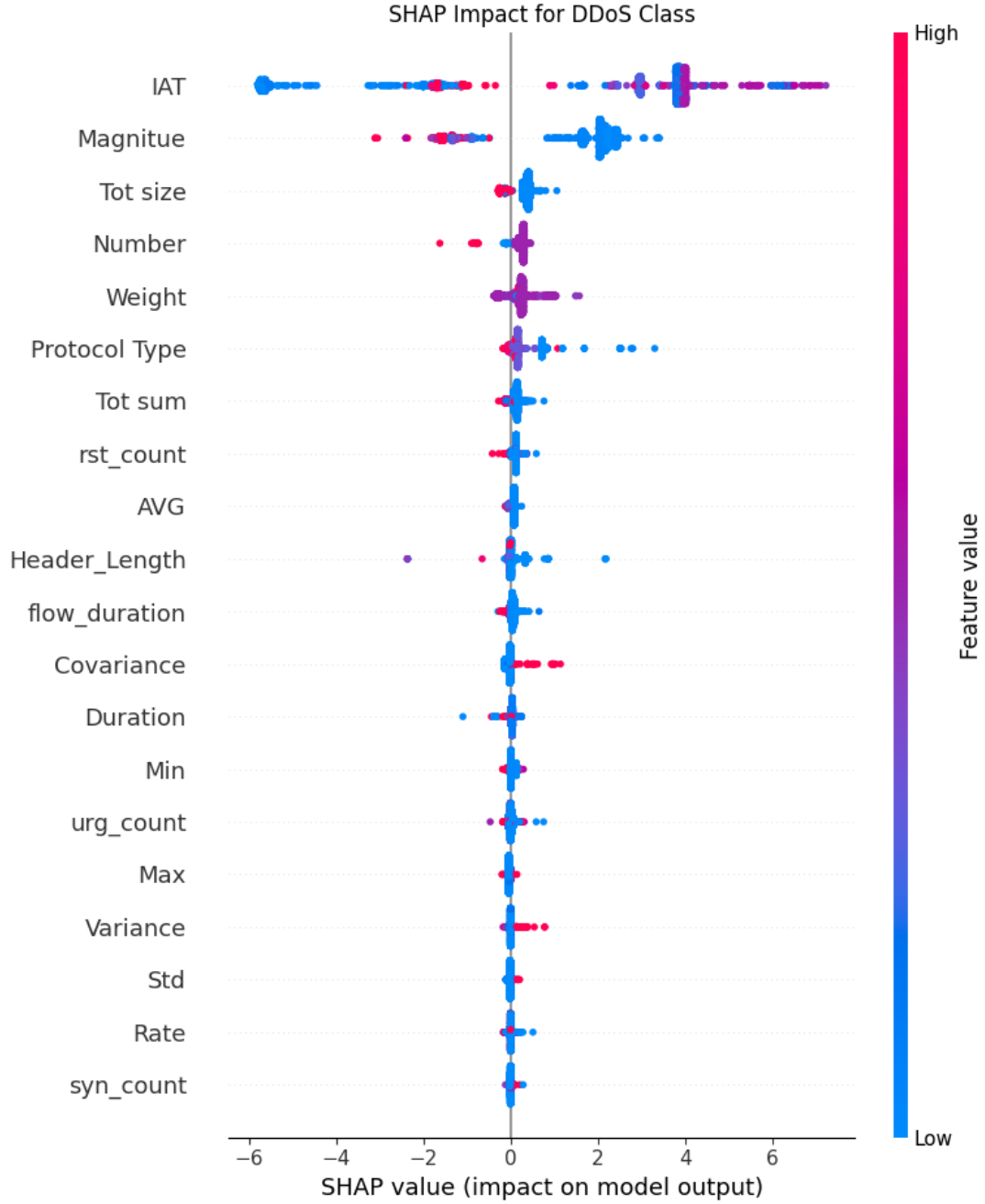


Figure 4.18: SHAP Summary Plot for Multi-Class Classification

We also observed that some features have varying impacts depending on the class, which confirms the model's ability to capture complex relationships between features and attack categories. This is particularly important in multi-class intrusion detection, where different attacks exhibit different behaviors.

4.7.3 Discussion of SHAP Findings

From the SHAP analysis, we can draw several important conclusions.

In the binary classification scenario, the SHAP analysis revealed that a small number

of features dominate the decision-making process. In particular, `rst_count`, `IAT`, and `Tot` size were identified as the most influential features. We observed that high values of `rst_count` strongly push the model toward predicting malicious traffic, which is consistent with the behavior of RST-based flooding attacks, where adversaries send large volumes of TCP reset packets to forcibly terminate connections. Similarly, `IAT` (Inter-Arrival Time) plays a critical role, where abnormal timing patterns whether unusually low values reflecting high-frequency packet flooding or irregular variance indicative of scanning behavior are consistently associated with malicious traffic.

The SHAP summary plot also showed that some features have overlapping effects between benign and malicious classes. This explains the small number of misclassifications observed in the confusion matrix. For example, certain benign samples with unusually high `rst_count` or abnormal timing behavior may be incorrectly classified as attacks (false positives). Conversely, some attack samples with normal-like feature values may lead to false negatives.

In the multi-class scenario, the SHAP analysis provides deeper insights into how the model distinguishes between different attack categories. As shown in Figure 4.17, the feature `IAT` becomes the most dominant feature across all classes, followed by `Header_Length`, `rst_count`, and `Number`. This indicates that temporal and structural characteristics of network traffic are essential for differentiating between attack types.

Unlike the binary case, we observe that feature importance is more distributed across multiple variables. This reflects the increased complexity of the task, as the model must separate several attack categories with potentially overlapping behaviors.

The class-specific SHAP analysis (e.g., for DDoS) further highlights how features contribute differently depending on the attack type. For instance, `IAT` shows the strongest influence on DDoS classification, which is consistent with the flooding nature of DDoS attacks that generate distinctive packet timing patterns. `Magnitude`, a measure of overall flow volume, also contributes significantly, reflecting the high traffic load typically generated by DDoS campaigns. Other features such as `Tot` size and `Number` contribute more moderately, confirming that the model relies on a combination of features rather than a single dominant indicator.

By connecting the SHAP results with the confusion matrix analysis, we can better understand the origin of classification errors. In the multi-class setting, some attack categories share similar feature patterns, particularly in terms of timing and packet structure. This overlap leads to confusion between classes such as DoS, DDoS, and other traffic-based attacks.

The SHAP plots clearly show that several features have overlapping SHAP value distributions across classes. As a result, the model may assign similar importance to these features for different attack types, which explains why certain samples are misclassified.

Additionally, minority classes such as Brute Force or Web-based attacks tend to have

less distinctive feature patterns, making them harder to separate. This is consistent with the lower recall observed for these classes in the confusion matrix.

Overall, SHAP confirms that the model learns meaningful patterns and explains both its high performance and its limitations, increasing our confidence in its use for intrusion detection.

4.8 Conclusion

In this chapter, we evaluated several machine learning and deep learning models for intrusion detection using the CICIoT2023 dataset, considering both binary and multi-class classification tasks.

We observed that all models achieved very high performance in the binary scenario, while the multi-class task revealed clearer differences. In particular, machine learning models—especially XGBoost and LightGBM—demonstrated higher accuracy, stability, and robustness compared to deep learning models, which showed more variability and higher computational cost.

We also introduced SHAP to explain the predictions of the best-performing model (XGBoost), allowing us to better understand feature importance and interpret some of the misclassifications observed in the confusion matrices.

In addition, a global comparison including performance and efficiency metrics confirmed that XGBoost provides the best trade-off between accuracy, model size, and computational cost, making it well-suited for real-world IoT environments.

Overall, we conclude that machine learning models, particularly XGBoost, offer the most effective and practical solution for IoT intrusion detection in this context.

Chapter 5

General Conclusion

Contents

5.1	General Conclusion	69
5.1.1	Summary of Work	69
5.1.2	Key Findings	70
5.1.3	Concluding Remarks	70

5.1 General Conclusion

5.1.1 Summary of Work

In this thesis, we addressed the growing challenge of securing IoT environments through effective intrusion detection systems (IDS). While many existing approaches focus primarily on maximizing accuracy, we aimed to go further by considering not only predictive performance, but also computational efficiency and model interpretability, which are essential for real-world deployment.

To achieve this, we designed and implemented a complete IDS pipeline based on the CICIoT2023 dataset. The preprocessing phase included data cleaning, attack grouping into meaningful categories, label encoding, hybrid resampling to address class imbalance, feature scaling, and feature selection. This allowed us to transform a large and highly imbalanced dataset into a structured form suitable for both binary and multi-class classification tasks.

We then evaluated a set of six models, including three machine learning models (Random Forest, XGBoost, and LightGBM) and three deep learning models (1D-CNN, DNN, and LSTM). In addition to standard evaluation metrics such as accuracy, precision, recall, and F1-score, we also considered efficiency-related factors such as model size, training time, and inference speed. Finally, we incorporated SHAP-based explainability to better understand the behavior of the best-performing model.

5.1.2 Key Findings

Our experimental results clearly show that machine learning models outperform deep learning models in this context, particularly in the multi-class classification task. Among all models, XGBoost achieved the best overall performance, combining high accuracy (up to 99.57% in the multi-class scenario) with relatively low computational cost. LightGBM also provided competitive results with an even smaller model size, making it highly suitable for resource-constrained environments.

In contrast, deep learning models did not offer a significant advantage when applied to structured tabular network data. While models such as 1D-CNN and LSTM produced acceptable results, they required considerably longer training times and did not surpass the performance of ensemble-based methods. The DNN model, in particular, showed clear limitations in handling multi-class imbalance.

We also observed that class imbalance remains a challenging issue, especially for minority attack categories such as Brute Force and Web-based attacks. Despite applying hybrid resampling techniques, these classes still exhibited lower recall, suggesting that more advanced imbalance handling methods are required.

The integration of SHAP provided valuable insights into the model's decision-making process. We found that a limited number of features play a dominant role in classification, and that these features are consistent with meaningful network behavior. This confirms that the model is learning relevant patterns rather than relying on noise.

5.1.3 Concluding Remarks

Overall, this work demonstrates that tree-based machine learning models, particularly XGBoost and LightGBM, provide an effective and practical solution for intrusion detection in IoT environments. They offer an excellent balance between accuracy, efficiency, and interpretability, outperforming more complex deep learning approaches in this context.

The proposed framework combines effective data preprocessing, robust classification models, and SHAP-based explainability to provide a comprehensive approach for IoT intrusion detection. The experimental results demonstrate the potential of this approach to support the development of accurate, efficient, and interpretable IDS solutions.

We believe that this work provides a solid and reproducible foundation for future research in IoT security, while emphasizing the importance of combining performance, efficiency, and explainability in the design of modern IDS solutions.

Bibliography

- [1] Arshdeep Bahga and Vijay Madisetti. *Internet of Things: A Hands-On Approach*. VPT, 2019.
- [2] Dieter Uckelmann, Mark Harrison, and Florian Michahelles, editors. *Architecting the Internet of Things*. Springer, Berlin, Heidelberg, 2011.
- [3] AKM Bahalul Haque and Sonia Tasmin. Security threats and research challenges of iot: A review. *Journal of Engineering Advancements*, 1(4):170–182, 2020.
- [4] National Institute of Standards and Technology. An introduction to information security. Technical Report NIST Special Publication 800-12 Rev.1, National Institute of Standards and Technology, Gaithersburg, MD, 2017.
- [5] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Technical report, Lockheed Martin Corporation, 2011.
- [6] Tazeen Fatima and B. Sasi Kumar. Cyber attack lifecycle analysis. *Journal of Engineering and Technology Management (JETM)*, 73:396–403, 2024.
- [7] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. Pearson, 4th edition, 2018.
- [8] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.
- [9] Hein Tun, Sergey Lupin, Htun Htun Linn, and Kyaw Nay Zaw Lin. Selection the perimeter protection equipment in security systems. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 1–5. IEEE, 2018.
- [10] Guillaume Hiet. *Détection d'intrusions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications: mise en œuvre sous Linux pour les programmes Java*. Thèse de doctorat, Université Rennes 1, France, 2008.

- [11] Elhadj Benkhelifa, Thomas Welsh, and Walaa Hamouda. A critical review of practices and challenges in intrusion detection systems for iot: Toward universal and resilient systems. *IEEE Communications Surveys & Tutorials*, 20(4):3496–3509, 2018.
- [12] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6. IEEE, 2009.
- [13] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Jill Slay. Towards developing network intrusion detection system datasets for iot embedded systems. *Future Generation Computer Systems*, 100:548–557, 2019.
- [14] Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2015.
- [15] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new dataset for cyber security dataset for intrusion detection. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 109–116, 2018.
- [16] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. IoT-23: A labeled dataset with malicious and benign IoT network traffic. <https://www.stratosphereips.org/datasets-iot23>, 2020.
- [17] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13):5941, 2023.
- [18] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Boston, 3rd edition, 2011.
- [19] Umair Shafique and Hina Qaiser. A comparative study of data mining process models (KDD, CRISP-DM and SEMMA). *International Journal of Innovation and Scientific Research*, 12(1):217–222, 2014.
- [20] Harley Bland and Scott J. Bland. *Machine Learning Foundations: An Algorithmic Pythonic Approach*. Addison-Wesley Professional, 2019.
- [21] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, Inc., 2nd edition, 2019.

- [22] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016.
- [23] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [24] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.
- [25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [26] Josue Genaro Almaraz-Rivera, Jose Alberto Perez-Diaz, and Juan Antonio Cantoral-Ceballos. Transport and application layer ddos attacks detection to iot devices by using machine learning and deep learning models. *Sensors*, 22(9):3367, 2022.
- [27] Raed J. Alzahrani and Ahmed Alzahrani. Security analysis of ddos attacks using machine learning algorithms in networks traffic. *Electronics*, 10(23):2919, 2021.
- [28] Pradeep Kumar, Hemanshu Bagga, Bhupesh Singh Netam, and Vinay Uduthalappally. Sad-iot: Security analysis of ddos attacks in iot networks. *Wireless Personal Communications*, 122(1):87–108, 2022.
- [29] Nitin V. Patil, C. Rama Krishna, and Krishan Kumar. Ssk-ddos: Distributed stream processing framework based classification system for ddos attacks. *Cluster Computing*, 25(2):1355–1372, 2022.
- [30] Yanjia Jia, Fan Zhong, Arwa Alrawais, Bei Gong, and Xiuzhen Cheng. Flowguard: An intelligent edge defense mechanism against iot ddos attacks. *IEEE Internet of Things Journal*, 7(10):9552–9562, 2020.
- [31] Sayeh Seifousadati, Meysam Naraki, Behrouz Beigizadeh, and Tuan-Manh Hoang. Efficient machine learning-based methods for ddos attack detection in iot networks. *Physical Communication*, 60:102144, 2023.
- [32] Felipe L. Becerra-Suarez, Isai Fernández-Roman, and Manuel G. Forero. Improvement of distributed denial of service attack detection through machine learning and data processing. *Mathematics*, 12(9):1294, 2024.
- [33] Sagheer Abbas, Shahid Hussain, Sayed Mohammad Jafar Jalali, Ahmad Almogren, Furqan Ahmed, and Abdullah Al-Amri. Evaluating deep learning variants for cyber-attacks detection and multi-class classification in iot networks. *PeerJ Computer Science*, 10:e1793, 2024.

- [34] Mohammad Mazyad Khan and Mansoor Alkhathami. Anomaly detection in iot-based healthcare: machine learning for enhanced security. *Scientific Reports*, 14(1):5872, 2024.
- [35] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [36] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

