INTELLIGENT NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING AND MLOPS

MAJOR PROJECT REPORT

submitted in partial fulfilment of the requirements for the award of

the degree of

MASTER OF TECHNOLOGY

in

Computer Science and Engineering

Submitted by:

AKHILESH BHORIA

(2K24/CSE/14)

Submitted to:

Prof. Shailender Kumar

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, New Delhi-110042

DEPT. OF COMPUTER SCIENCE AND ENGINEERING DELHI

TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, New Delhi-110042

CANDIDATE'S DECLARATION

I, **AKHILESH BHORIA, (2K24/CSE/14)**, student of M.Tech (Computer Science Engineering), hereby declare that the project titled--"**Intelligent Network Intrusion Detection System Using Machine Learning and MLOPS**," submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, New Delhi, is fulfilment of the requirement of the award of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship, or any other similar title or recognition.

Place: New Delhi **Akhilesh Bhoria**

Date: 2K24/CSE/14

DEPT. OF COMPUTER SCIENCE AND ENGINEERING DELHI

TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, New Delhi-110042

CERTIFICATE

I hereby certify that the project titled "**Intelligent Network Intrusion Detection System Using Machine Learning and MLOPS**" submitted by Akhilesh Bhoria - 24/CSE/14, Department of Computer Science Engineering, Delhi Technology University, Delhi in partial fulfilment of the degree of Master of Technology is a record of the project work carried out by the student under my guidance and supervision.

Place: New Delhi
Date:

Prof. Shailender Kumar
SUPERVISOR
(Professor, CSE, DTU)

DEPT. OF COMPUTER SCIENCE AND ENGINEERING DELHI

TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, New Delhi-110042

ACKNOWLEDGEMENT

I am grateful to Prof. SHAILENDER KUMAR, Professor, Department of Computer Science Engineering, and all the faculty members at DTU for their guidance and support throughout the seminar. I would like to extend my heartfelt gratitude to Prof. Anil Singh Parihar, Head of the Department, Computer Science and Engineering, Delhi Technological University, for his continuous guidance, motivation, unwavering encouragement and invaluable insights.

I appreciate the support provided by the University, including the laboratories, infrastructure, testing facilities, and environment that enabled me to work without interruption. I thank my lab assistants, seniors, and peer group for their aid and knowledge on various subjects.

I am also grateful to everyone who directly or indirectly contributed to the completion of this project, and my parents and well-wishers for their constant support, belief in me, and encouragement.

Akhilesh Bhoria
2K24/CSE/14

ABSTRACT

The fast development of digital communication and internet services poses a great threat to computer networks in terms of possible cyber-attacks and unauthorized access. The traditional security measures are not able to detect attacks promptly and effectively. This project introduces **"An Intelligent Network Intrusion Detection System Based on Machine Learning and MLOps"** that can detect cyberattacks and other malicious network operations quickly and automatically using a MLOps workflow.

In particular, Machine Learning (ML) will be used to create an efficient intrusion detection system. For this purpose, network traffic data sets that include several kinds of attacks like Denial of Service (DoS) attacks, probes, brute force attacks, unauthorized access, and others will be used. The data will be preprocessed, transformed, validated, and engineered.

Several supervised machine learning methods are employed in intrusion detection, such as the Random Forest algorithm, Decision Tree algorithm, Support Vector Machine algorithm (SVM), and the K-Nearest Neighbors (KNN) algorithm. The training and evaluation of the algorithms are done based on their performances, which include accuracy, precision, recall, F1 score, and confusion matrix. The most effective one is chosen and deployed via an automated MLOps pipeline.

The MLOps structure ensures continuous integration, continuous training, continuous deployment, and model monitoring. Features like ETL pipelines, data ingestion, model versioning, experiment tracking, automatic deployment, and monitoring help ensure that the process scales, becomes repeatable, and remains easy to maintain. The proposed solution can also detect intrusions in real-time and minimize manual effort required when updating the model.

In conclusion, the implemented solution constitutes an advanced adaptive system that is intelligent and able to detect both known and unknown types of attacks with higher levels of accuracy while reducing false positives. The current project proves that the implementation of Machine Learning and MLOps concepts can lead to improvements in the field of modern network security systems.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

As we know today's Networks have become indispensable when it comes to communication, conducting business, using cloud computing, providing online services, and exchanging information. Enterprises from different industries such as banking, healthcare, education, ecommerce, military, and many others use network infrastructure for storing, processing, and transferring critical information. Moreover, with the emergence and widespread application cloud computing, and large-scale data analytics, distributed computing, the amount of network traffic created each day has also become enormous.

At the same time, due to advances in cybercrimes, there is an increasing number of threats to computer networks. Hackers find ways to penetrate into computer networks in order to steal sensitive data and conduct malware attacks, password-guessing attempts, Denial-of-Service (DoS) attacks, and various other cyber threats that can harm enterprises. This may cause significant financial loss to businesses, expose confidential information, disrupt the work process, and result in privacy and reputational issues.

Current computer networks suffer from several types of cyber-attacks that may affect the network infrastructure, resources, sensitive data, and even the network communications. Cyber-attacks vary depending on the methods employed, their aims, and the effects they have on the network security. It is necessary to comprehend the different types of attacks in order to create an efficient IDS that detects the malicious behavior effectively. This is illustrated in Figure 1.1.

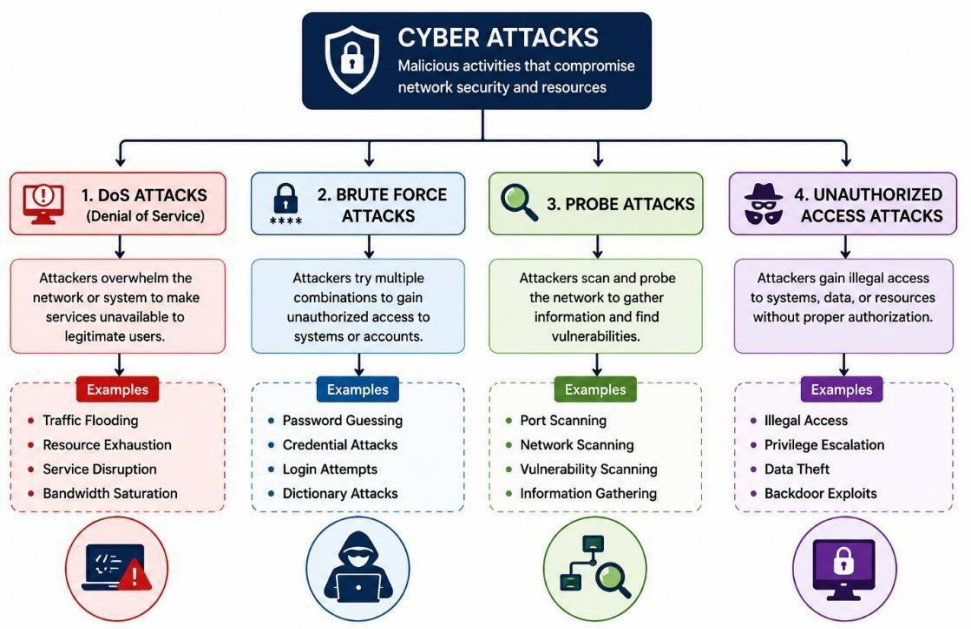

Fig 1.1 Major Categories of Cyberattacks in Network Intrusion Detection Systems

Different categories presented in Figure 1.1 above are among those common attack types that are witnessed in modern network environments. Attacks categorized as Denial of Service (DoS)

aim at disruption of network services through overloading the computing capacity of target networks, while brute force attacks target gaining access via trial-and-error login attempts. Probe attacks involve reconnaissance and collection of network information ahead of an actual attack, while unauthorized access attacks entail accessing systems illegally.

This MLOps-Based Intelligent Network Intrusion Detection System is meant to detect such attack categories using machine learning algorithms and continuous network traffic analysis.

Conventional security mechanisms, including firewalls, antivirus applications, and intrusion detection systems are only partly useful in protecting networks from modern day attacks. The conventional Intrusion Detection System (IDS) relies on the attack signatures for detecting any malicious activities. Though such systems work effectively for the known attacks, they tend to miss out on new forms of attacks or zero-day attacks. Additionally, modern day computer networks handle huge amounts of traffic, and therefore the task of analyzing the activities becomes challenging.

To solve these problems, machine learning techniques are now being applied in cybersecurity. Machine learning algorithms automatically discover patterns in the previous network activity traffic and then label them as malicious or normal. In doing so, IDSs using machine learning techniques help in increasing the accuracy and lowering rate bad alarms, as well as ensuring adaptation to emerging forms of cyber threats. Some of the popular algorithms used for this include such as SVM, KNN, Decision tree.

However, constructing a machine learning model alone is insufficient to deploy a reliable cybersecurity system into production. Machine learning models are required to continuously ingest data, preprocess them, train, test, deploy, monitor, and then repeat the process by retraining the machine learning model on updated data. Doing this manually poses …face difficulties related to performance and processing time. To address these limitations, the concept of Machine Learning Operations, known as MLOps, has become a key approach to automate the machine learning process.

Machine Learning Operations (MLOps) incorporates the principles of Machine Learning, DevOps, and Data Engineering techniques to construct automated machine learning systems. This process allows for efficient continuous integration, continuous deployment, automated retraining, monitoring, version control, and reproducibility of machine learning projects.

The rationale the intelligent IDS using an end-to-end MLOps framework capable of identifying suspicious network activities while also streamlining and automating the complete machine learning process workflow. It combines implementation of ETL pipelines, ML Models, deployment processes, CI/CD, monitoring systems, and retraining systems into one integrated solution.

This project focuses on developing a scalable and production-ready intrusion detection system So, ML techniques such as SVM, KNN to identify malicious network activities. This will be achieved through the use of automation tools that will enable automatic data ingestion, preprocessing, feature extraction, modeling, deployment, and monitoring.

1.2 Challenges in Existing Intrusion Detection Systems

It faces several challenges and limitations in cybersecurity. Signature-based IDS can identify only known attack patterns and are ineffective against new or previously unseen threats and new or evolving attack scenarios. Attackers frequently employ methods to evade the detection process, and hence, static solutions become inadequate in dealing with modern computer networks.

Conventional IDS depends on fixed rules and pre known attack to cyber threat in computer networks. Even though these traditional intrusion detection solutions are effective when it comes to dealing with attacks that have been previously detected, they are subject to many drawbacks including incapability to detect new attacks, poor scalability, lack of automation, and high false positives. Machine Learning solutions provide enhanced capabilities in terms of attack detection, but many current systems still lack automated deployment and lifecycle management processes.

The proposed MLOps-based Intelligent Network Intrusion Detection System solves these shortcomings by adopting machine learning pipelines, monitoring, deployment, and intrusion detection mechanisms. Table 1.1 shows the comparison between Traditional IDS, ML-based IDS, and the proposed MLOps-based IDS system.

Table 1.1 Difference between Traditional IDS and the Proposed MLOps-Based Intrusion Detection System

Feature	**Traditional IDS**	**ML Based IDS**	**Proposed MLOpsBased IDS**
Methods	Rule Based	Machine Learning	Machine Learning + MLOps
Detect Known Attacks	Yes	Yes	Yes
Detect Unknown Attacks	No	Partial	Yes
Real-Time Detection	Limited	Moderate	High
Automation Support	No	Partial	Yes
Scalability	Low	Moderate	High
Continuous Monitoring	No	Limited	Yes
Automated Retraining	No	No	Yes
CI/CD Integration	No	No	Yes
Deployment Support	Manual	Partial	Automated
Model Versioning	No	Limited	Yes

Reproducibility	Low	Moderate	High
Maintenance Effort	High	Moderate	Low
Adaptability to New Threats	Poor	Moderate	High

It is apparent from Table 1.1 that traditional intrusion detection systems face several limitations related to scalability, automation, adaptability, and real-time monitoring. Though ML intrusion detection systems offer enhanced capabilities of detecting attacks, yet they still have shortcomings such as not having lifecycle management and automation.

In contrast, the proposed MLOps-based Intelligent Network Intrusion Detection System overcomes these challenges by incorporating the implementation of automated workflows for data ingestion, data preprocessing, model training, deployment, monitoring, and retraining.

The other critical problem associated with existing IDSs is the creation of both a FP rate and FN rate. The former situation arises where normal activity on the network is identified as malicious, whereas the latter case refers to instances where an attack goes unrecognized by the system. In either case, the accuracy of the detection mechanism is greatly compromised.

Current enterprise networks and cloud computing systems produce huge amounts of network traffic data on a second-by-second basis. Analyzing these large volumes of data requires a scalable architecture. Traditional IDS systems do not scale well and cannot process high amounts of network traffic.

Despite the advantages offered by machine learning intrusion detection solutions, some of the implementation challenges remain unsolved. Such challenges include data collection, preprocessing, feature extraction, model retraining, deployment, performance monitoring, model version control, and non-reproducibility in machine learning workflows. Manual handling of machine learning pipelines makes development more difficult and inefficient for numerous systems.

Moreover, the machine learning model may become obsolete over time because of the changing behavior of network traffic and emerging cybersecurity threats. The absence of continuous monitoring and automated retraining can lead to serious degradation in model performance in a production environment.

Thus, an advanced and automated approach is needed in order to create an an IDS framework developed using the integration of Machine Learning and MLOps techniques.This approach should automate cybersecurity model development, deployment, and monitoring and ensure scalability and reproducibility in ML workflows.

1.3 Problem Statement

Due to the growing number of attacks and network attacks, traditional security systems have become insufficient in securing the current digital systems against network vulnerabilities.

Some of the drawbacks associated with intrusion detection systems include their inability to identify unknown attacks, lack of scalability, a high rate of false alarms, reliance on manual rules, and inefficient deployment mechanisms.

In addition, machine learning intrusion detection systems pose additional problems such as managing a data pipeline, training the model, deploying the model, monitoring and maintaining its performance, and retraining it periodically.

Therefore, it is essential to develop an intelligent, scalable, and automated Network IDS that can idientifies malicious network activities while incorporating a complete MLOps pipeline for automated data collection, preprocessing, model training, deployment, monitoring, retraining, and overall lifecycle management.

1.4 Project Objectives

The main proposed idea is to build Intelligent Network Intrusion Detection System by integrating Machine Learning and MLOps technologies to accurately and efficiently identify malicious network activities. The proposed system is designed to automate the entire machine learning workflow, including Data collection, Preprocessing, monitoring, and retraining.

The project focuses on implementing ETL pipelines for automated data processing and integrating supervised ML techniques including RF, DT, SVM, and K-Nearest Neighbors for intrusion detection purposes. The proposed system is intended to categorize traffic either normal or unNormal using extracted traffic features and behavioral analysis.

Another objective of the project is to assess the performance evaluation metrics such as accuracy, Confusion Matrix, F1-Score, Recall, and Precision analysis to determine the most effective intrusion detection model. The project also aims to implement automated deployment, monitoring, logging, and CI/CD integration using MLOps architecture to improve scalability, reproducibility, reliability, and operational efficiency.

The intended design framework also intends to minimize false positives and negatives rates while having real-time monitoring capability and automated detection features, which can be used in production-level cybersecurity.

1.5 Scope of the Project

Research covers various aspects including the design, development, implementation, and deployment of an End-to-End MLOps-based Intelligent IDS using ML techniques. System designed to build a scalable and automated cybersecurity framework capable of analyzing network traffic and detecting any malicious behavior.

The system includes several steps like data ingestion, ETL pipeline implementation Data preprocessing testing, deployment, monitoring, retraining. Project utilizes several publicly available intrusion detection datasets containing different categories of cyberattacks, including DoS activities, password-guessing attacks.

Design of the proposed solution incorporates a scalable MLOps framework that includes support for automatable processes, model versioning, CI/CD, deployment automation,

monitoring, and reproducible machine learning pipelines. This design will be deployable in enterprise networks, cloud computing environment, banking systems, educational institutes, health care infrastructure, governments, and IoT-based solutions for cybersecurity purposes.

MLOps intelligent network NID offers a scalable, efficient approach for detecting network threats intelligent solution for cybersecurity purposes in several fields and deployment scenarios. The proposed framework will facilitate automation in machine learning processes, intrusion detection, deployment, and monitoring in modern network infrastructures. Figure 1.2 shows the application scenario and areas of deployment for the proposed framework.

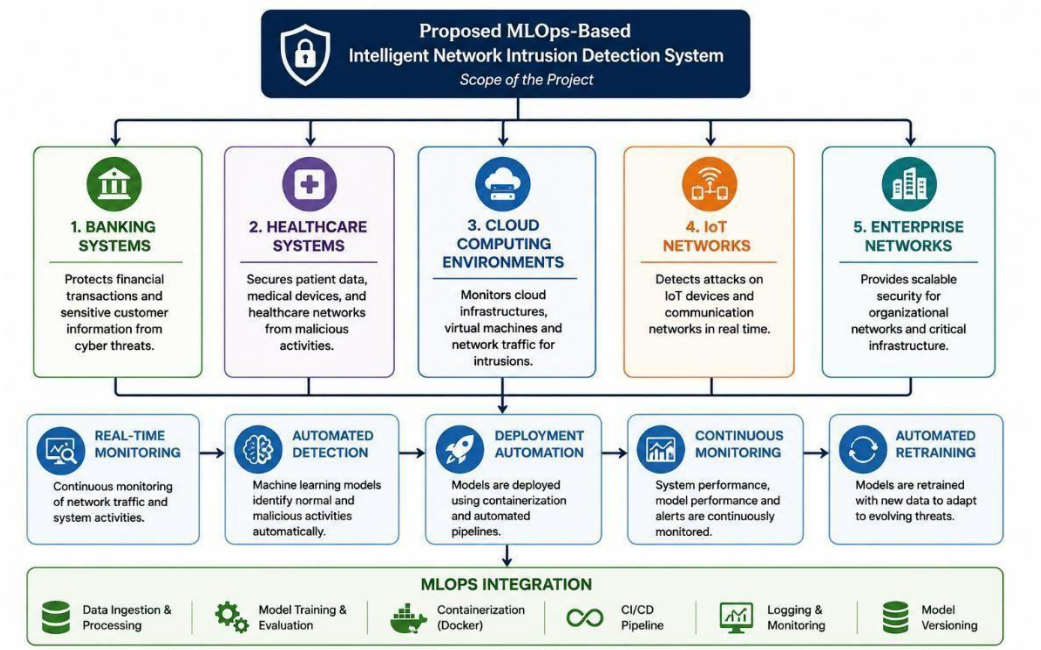

Fig 1.2 Scope and Application Areas of the Proposed MLOps-Based Intrusion Detection System

Figure tells range, areas application of the new intrusion detection framework. The new framework has been developed to help manage different applications where intrusion protection is needed. These applications include banking systems, health-care infrastructure, cloud-based services, Internet of Things networks, and other enterprise systems where intrusion prevention is required.

The intrusion detection framework uses Machine Learning and MLOps methods and techniques to offer automatic intrusion protection, deployment capabilities, and monitoring among others. Future development plans include cloud-native, Kubernetes, deep learning, and real-time stream processing abilities.

This platform will also facilitate real-time monitoring and automated intrusion detection systems for suspicious activities in real time. Some possible future developments of this system could be including deep learning algorithms, cloud-based architecture, Kubernetes, real-time streaming analytics, federated learning, AI explainability, and self-learning intrusion detection techniques.

In summary, the development The project is intended to deliver an solution that combine ML and MLOps technologies modern cybersecurity frameworks for intelligent detection.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

Internet technologies and applications and distributed systems architectures are leading to more and more cyber-attacks. In recent years, there have been countless examples of cyber-attacks on critical infrastructures such as financial institutions, hospitals, universities, retailing businesses, and governments. In order to detect and prevent attacks on network infrastructures, researchers and other organizations have created many Intrusion Detection Systems (IDSs) that utilize traditional approaches, machine learning techniques, and automated deployment mechanisms.

Traditional intrusion detection systems were often limited in terms of their functionalities, relying mostly on rule-based and signature-based techniques in detecting malicious activities in networks. Although such systems were able to detect malicious activities that had been previously identified by the developers, they could not detect new types of cyber-attacks. With developments in the advancement of AI.

In many times, In paradigm research from just focusing on improving the accuracy of the models to building scalable and production-ready machine learning systems. Current intrusion detection systems need automatic data pipelines, continuous integration and deployment, monitoring, retraining, and various other lifecycle processes required for production environments. This led to the emergence of Machine Learning Operations (MLOps), where Machine Learning, DevOps, and Data Engineering practices have been combined for automating the entire life cycle of machine learning applications.

The current chapter provides a comprehensive study on existing intrusion detection methodologies, classical as well as machine learning-based intrusion detection mechanisms, drawbacks associated with existing approaches, and developments in recent times in the area of MLOps-based cybersecurity systems.

2.2 Review of Existing Research Works

Over the years, researchers have developed various types of intrusion detection systems using different methodologies. Early detection techniques. where attack signatures and predetermined rules were utilized to identify attacks. Snort and Suricata are examples of wellknown systems that used signature-based methods to detect attacks through comparison of network traffic with pre-determined attack signatures. The systems successfully detected attacks that had already been identified; however, they were unable to identify zero-day attacks.

Researchers then developed intrusion detection systems based on anomaly detection. This was an approach that identified deviations from normal network activities as attacks. One major benefit of anomaly-based systems was their capability identify previously attacks; however, they produced many false alarms due to misclassifications of normal traffic as malicious.

The development of Machine Learning technology has resulted in many techniques have been applied in threats detection tasks. Supervised ML algorithms such SVM, Logistic Regression,

KNN, Decision tree have been extensively utilized for classifying network traffic patterns. Among these, algorithms like Random Forest and Decision Tree demonstrated strong classification performance and high detection accuracy and the potential for enhanced attack detection capabilities of intrusion detection datasets.

Throughout time, various approaches were developed for intrusion detection purposes to ensure cybersecurity and provide protection of networks in modern conditions. They are different in their capability, scalability, deployment requirements, automation capability, etc. While traditional intrusion detection was based on rules, now many of the new techniques utilize Machine Learning, Deep Learning, and automated ML frameworks to detect attacks in an efficient way. The major approaches to intrusion detection covered in the literature survey

Table 2.1 Difference of existing Intrusion Detection Approaches

Research Approach	Technique Used	Advantages	Limitations
Signature-Based IDS	Rule Matching	Detects known attacks	Cannot detect unknown attacks
Anomaly-Based IDS	Behavioral Analysis	Detects anomalies	High false positives
ML-Based IDS	Random Forest, SVM, KNN	Better accuracy	Limited deployment support
Deep Learning IDS	CNN, RNN, LSTM	Detects complex attacks	High computational cost
Proposed MLOpsBased IDS	ML + MLOps	Automation, monitoring, deployment	Future scalability challenges

Table 2.1 shows a number of disadvantages of traditional IDS, including adaptability issues, scalability, and ability to detect unknown types of attacks. Using ML and DL, the problem of detecting attacks becomes solved much better, but some intrusion detection systems miss a set of necessary functions like automation, monitoring, retraining and maintenance of ML models. This proves the necessity of applying Machine Learning combined with MLOps practices to design production-ready solutions.

There exist a number of publicly available datasets which are actively used for training and evaluating intrusion detection algorithms. Samples representing various cyberattacks, including DoS attacks, brute-force attacks, probing activities, unauthorized access attempts, and malware-related threats.

The researchers have also studied deep learning technologies including deep learning models such as ANN, LSTM, CNN, and RNN networks. These deep learning approaches algorithms automatically extract feature sets from network traffic data and help in detecting attacks.

However, although these research papers delivered improvements with respect to detection accuracy and the ability to identify cyberattacks, most of the research works neglected automation, scalability, monitoring, reproducibility, and lifecycle management issues while focusing only on the model creation and testing phases.
Current research trends involve the incorporation of Machine Learning into scalable deployment tools and automated processes. The idea of Machine Learning Operations (MLOps) is now gaining prominence as one of the major solutions for the implementation of production-ready machine learning architectures. Through the use of MLOps, it is possible to automate such processes as data intake, preprocessing, training of models, deployment, monitoring, retraining, and CI/CD integration for machine learning applications.

It has evolved considerably traditional the current era of smart and intelligent Machine Learning/MLOps-based cybersecurity systems. Initially, Intrusion Detection Systems made use of preprogrammed attack rules for detecting and preventing cyber-attacks. With time, however, the need arose to adopt a more sophisticated methodology for detecting new and evolving cyber-attacks.

Modern intrusion detection systems, due to developments in AI, deep learning, and automation technology, are now concerned not only with detecting attacks effectively, but also with scalability, automation, monitoring, and management throughout the entire lifecycle. The evolution of intrusion detection systems can be illustrated using the following diagram

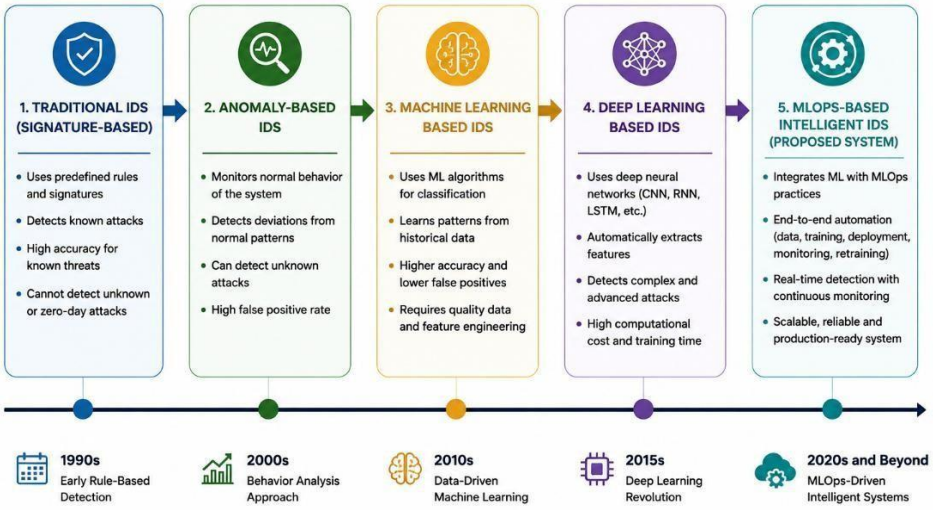

Fig 2.1 Evolution of Intrusion Detection Systems

Figure 2.1 which below provides evidence of the development process of intrusion detection systems throughout various technological generations. Classic Signature-based intrusion detection systems, they were unable previously unseen or emerging threats detect unknown or zero-day attacks. Intrusion detection systems based on anomalies helped solve the problem of adaptability due to the analysis of deviations from normal activity in a network but had a significant drawback - they usually gave many false positives.
Decision Tree, Random Forest, and Support Vector Machine (SVM) and KNN algorithms and thereby provided better results compared to previous models. The development of deep learning helped improve attack detection through automatic feature extraction from network traffic data.

Nowadays, the emergence of Machine Learning Operations made it possible to create intrusion detection systems on a large scale that can be used to monitor attacks, train continuously, and automate deployment.

Modern ML-based architectures rely on the concepts of modular pipeline development, automated ETL processes, versioning capabilities, artifact creation, monitoring, logging, and scalable deployment infrastructures. This makes ML models more reliable, repeatable, automated, and efficient when deployed into production environments.

Nevertheless, few studies have addressed the problem of integrating MLOps practices into intelligent intrusion detection systems, which is essential to enable their application to cybersecurity through automation. Therefore, intrusion detection integrates machine learningbased attack detection with MLOps pipelines.

2.3 Traditional Intrusion Detection Techniques

Traditionally, intrusion detection techniques involve the use of pre-set rules, attack signatures, statistics, and configuration of security policy to identify threats and attacks occurring on computer networks.

The methods identify attacks in the computer network by comparing the characteristics of the incoming network packets against attack signatures stored in the database. In case the network packets are similar to the attack signatures, an alert is generated to inform about a threat to the computer system. Advantage of signature-based approach is to identify previously known threats; however, it is ineffective in detecting new or unknown attacks.

Moreover, updating and maintaining signature databases involves a constant manual effort. Anomaly-based intrusion detection systems came into existence to address the shortcomings of the signature-based method. In anomaly detection systems, baseline profiles are set for normal network activity, and any deviation from these profiles is considered suspicious behavior. The anomaly detection method helps identify unknown attacks. Identifying normal network behavior in dynamic environments is difficult

Statistical intrusion detection methods involve analyzing traffic characteristics of networks, including packet sizes, duration of connections, traffic frequencies, protocols, and communications. Statistical methods involve comparing present traffic with predefined statistical patterns to detect any abnormality. While statistical intrusion detection methods offer valuable information regarding traffic analysis, their effectiveness is limited when dealing with advanced cyber-attacks.

Traditional intrusion detection systems provided a strong foundation for network security; however, their inability to adapt dynamically to evolving attack patterns and modern largescale network environments encouraged researchers to adopt intelligent Machine Learning based approaches for intrusion detection.

2.4 Machine Learning-Based Intrusion Detection Approaches

Machine learning-based intrusion detection attracted considerable automatically detect malicious activities. Unlike conventional rule-based systems, machine learning techniques learn underlying patterns and relationships from historical datasets.

Supervised Machine Learning algorithms are extensively applied in intrusion detection systems because they are trained using labeled datasets containing both legitimate and attack traffic samples. Decision Tree algorithms classify network traffic using hierarchical decision structures and provide interpretable attack classification models. Random Forest algorithms combine multiple decision trees using ensemble learning techniques to enhance prediction accuracy and minimize overfitting issues. Optimal hyperplanes are used in SVM-based algorithms to classify attacks and normal traffic, and these techniques operate efficiently in high dimensional feature space. Similarity measurement between neighbors is applied in KNearest Neighbor (KNN) algorithms to classify traffic.

In addition, there are other intrusion detection techniques, which employ unsupervised learning methods where only unlabeled data exists. K-Means clustering algorithm is one type of these algorithms that classifies samples into clusters, thereby detecting any anomalies in the sample set.

Deep Learning is another field that significantly enhances intrusion detection performance. Various types of Deep Learning-based intrusion detection systems include ANN, CNN, RNN, and LSTM.

Different Machine Learning & Deep Learning algorithms have gained popularity for intrusion detection tasks because of their ability to efficiently analyze network traffic data in bulk and detect any anomalies. These algorithms vary in terms of how they classify attacks, their processing difficulty, effectiveness, scalability, and application in cybersecurity. Some algorithms give more explanatory results than others, whereas some algorithms detect advanced threats more effectively than others.

Table 2.2 Comparison ML and DL Algorithms Applied in Intrusion Detection Systems

Algorithm	Working Principle	Advantages	Limitations
Decision Tree	Tree-based classification	Easy interpretation	Overfitting
Random Forest	Ensemble learning	High accuracy	More computation
SVM	Hyperplane separation	Works in high dimensions	Slow for large datasets
KNN	Neighbor similarity	Simple implementation	High prediction time
CNN	Feature extraction	Detects complex patterns	High training cost

LSTM	Sequential learning	Good for temporal attacks	Computationally expensive

As seen in Table 2.2, distinct advantages and limitations disadvantages in performing intrusion detection. The traditional Machine Learning approaches SVM, KNN, provide highly efficient results in classifying structured data related to network traffic; in contrast, Deep Learning methods like CNN and LSTM provide better detection capabilities in terms of detecting sophisticated attack patterns and sequences of network activities. However, Deep Learning requires high levels of computation power and higher training costs. The suggested framework the intrusion detection system primarily relies on supervised machine learning algorithms due to their high accuracy, efficiency, and ease of interpretation.
.

Despite the fact that Machine Learning greatly enhances intrusion detection effectiveness, the application of ML solutions brings a set of operational difficulties like data preparation, deployment difficulties, retraining, scaling, monitoring, reproducibility, and life cycle management.

In order to overcome these difficulties, the emerging studies are relying on ML Ops principles to automate their ML models. MLOps provides capabilities to automate the following tasks: automated pipeline for ETL, continuous integration and continuous deployment, system monitoring, model versioning, artifact management, log management, and automation of retraining process. All of these capabilities are required when designing an intrusion detection system.

2.5 Limitations of Existing Systems

While there have been considerable developments in intrusion detection techniques, some technical shortcomings remain in current systems. Emerging or unknown attacks because they rely heavily on predefined attack signatures that can identify only previously recognized threats. Maintaining and updating these signatures to match continuously evolving cyberattacks is also a challenging task.

In contrast, anomaly-based intrusion detection systems make better detections of unknown attacks by basing the classification on behavior rather than signatures. The downside is that such systems generate numerous false positives since legitimate network activity might be mistakenly classified as malicious activity.

However, many existing machine learning-based intrusion detection models pay limited attention to the deployment process. Some ML systems have not addressed critical issues in the production environment such as pipeline support.

Yet another important challenge faced by intrusion detection frameworks relates to the lack of scaling machine learning operations in a number of them. AML system needs to be constantly monitored, retrained, versioned, logged, have artifact management and CI/CD capabilities in order to sustain its efficiency and scalability. This makes ML system operation more complex and difficult to reproduce manually.

A lot of research on IDS focuses solely on experiments performed with datasets available under controlled conditions, which do not necessarily reflect how efficient such approaches can be applied to live networks.

2.6 Research Gap Analysis

As per the literature survey, there has been a considerable amount of work done on intrusion detection systems that use conventional security measures and intelligent methods like Machine Learning and Deep Learning. Current work has shown enhanced results in terms of attack detection capability and anomaly identification through intelligent learning techniques. Nevertheless, many research questions are still open in intrusion detection today.

Firstly, most of the current intrusion detection systems concentrate on enhancing classification accuracy without considering practical issues such as scalability, continuous integration, deployment, automation, monitoring, reproducibility, retraining, and lifecycle management. Second, machine learning-based solutions have been created in such a way that they can be tested experimentally but cannot be applied practically to cybersecurity scenarios.

In addition, most existing systems do not include Machine Learning Operations (MLOps) to automate the entire machine learning process lifecycle. Challenges related to automated ETL pipelines, data validation, preprocessing automation, deployment workflows, model versioning, monitoring systems, artifact generation, CI/CD integration, and continuous retraining are often neglected.

Moreover, few researchers have paid attention to integrating intelligent network intrusion detection systems with scalable MLOps frameworks that can facilitate real-time deployment, automated monitoring, and reproducible pipeline processes.

Thus, in the current scenario, the proposed research seeks to fill the above knowledge gap through an End-to-End MLOps-Based Intelligent Network Intrusion Detection System using Machine Learning project. In the proposed framework, the system focus will be on ETL automation, MLOps architecture scalability, intelligent attacks detection, deployment, CI/CD, monitoring, and lifecycle management.

CHAPTER 3
SYSTEM ANALYSIS AND REQUIREMENTS

3.1 Introduction

Here, we have are two critical stages in the development of any machine learning and cybersecurity system. This stage is characterized by studying the drawbacks of already existing systems, identification of their weaknesses, and determination of system requirements followed by the presentation of a more advanced solution. Modern cybersecurity is characterized by the appearance of tremendous amounts of data generated by cloud computing systems, enterprises' infrastructures, the IoT environment, and internet-based applications. Given the rapid evolution of cyber threats, traditional solutions for detecting such attacks are no longer able to operate effectively.

In this regard, End-to-End MLOps-Based Intelligent Network Intrusion Detection System capable of fully automating all stages of ML and cybersecurity operation while enhancing the accuracy of attacks' detection and making the system scalable in terms of its operation. Traditional IDSs operate exclusively based on pre-defined static rules and signatures. The proposed solution will leverage the power of Machine Learning and MLOps.

In this chapter, there will be discussed the analysis of the existing IDS, limitations of the current methods, architecture of the proposed system, including its functional and non-functional requirements, as well as the necessary software and hardware specifications, and feasibility analysis of the proposed system. This chapter also stresses the importance of scalable MLOps pipeline, workflow automation, and production architecture for IDS.

3.2 Existing System

Current methods include systems such as Snort and Suricata that analyze network traffic by matching incoming packets to existing attack signatures. The moment a match happens, alert messages about possible intrusion attempts are sent out. These systems offer the initial level of protection against known cyberattacks and are popularly used in various network security scenarios.

However, despite their effectiveness in dealing with known attacks, there are numerous disadvantages of using current. One key limitation is the inability of these systems to recognize and protect against unknown or zero-day attacks. As the detection process depends entirely on pre-existing attack signatures and manually created rules, any new pattern of attacks will not be detected and recognized by the detection system.

The next issue that limits the effectiveness of current systems is the creation of a high level of false positives and false negatives. The actions that take place within an organization might not always be identified correctly as either malicious or harmless, which leads to inefficiencies in the operations of the detection systems, as well as extra work for security managers.

Current enterprises have an immense amount of data traffic due to cloud computing services, IoT systems, distributed networks, and web-based software solutions. Current intrusion detection systems find it hard to process these large data traffic volumes.

The detection capabilities of ML-based IDSs were enhanced through the learning of patterns of traffic behaviors through datasets. Nonetheless, most machine learning techniques available today focus more on experiments with models as opposed to deployment and operations of these models. The vast majority of machine learning-based IDSs lack effective mechanisms for automation in their processes.

Manual implementation of ML-based workflow is often the norm, hence making it hard to implement and maintain such workflows in a production environment. Additionally, there lacks an efficient way of integrating MLOps into the machine learning workflow process. Issues with reproducibility, deployment automation, monitoring, logging, artifact management, and CI/CD are some common challenges faced today.

3.3 Proposed System

Proposed Approach: The proposed approach involves implementation of an intelligent Network Intrusion Detection System using Machine and MLOps techniques. This system will be developed for continuous network traffic monitoring and analysis, detection of any malicious activity in the network, and automation of the entire machine learning life cycle using MLOps pipelines.

The proposed framework utilizes various modules like validation, transformation, deployment, monitoring, logging, and model retraining. Network traffic data sets that include both normal and malicious network traffic logs are ingested into the machine learning pipeline for training machine learning models.

Modular Approach and Pipelines: The proposed framework employs a pipeline approach to machine learning much like production-scale MLOps. It includes modules for data ingestion, data validation, data transformation, training, evaluation, deployment, and prediction of predictions.

The framework of Machine Learning as well MLOps techniques to enable the design and deployment of a smart, scalable, and autonomous network intrusion detection system capable of supporting real-time cybersecurity monitoring and deployment. The proposed system framework comprises several pipeline modules for data ingestion, validation, pre-processing, training, evaluation, deployment, monitoring, and autonomous re-training.

It also employs several MLOps methodologies such as CI/CD integration, model versioning, workflow automation, artifact management, monitoring, and experiment management to enable scalability at a production level. Figure 3.1 illustrates architecture of MLOps-based intrusion detection framework and workflow of the proposed interaction among various pipeline components.

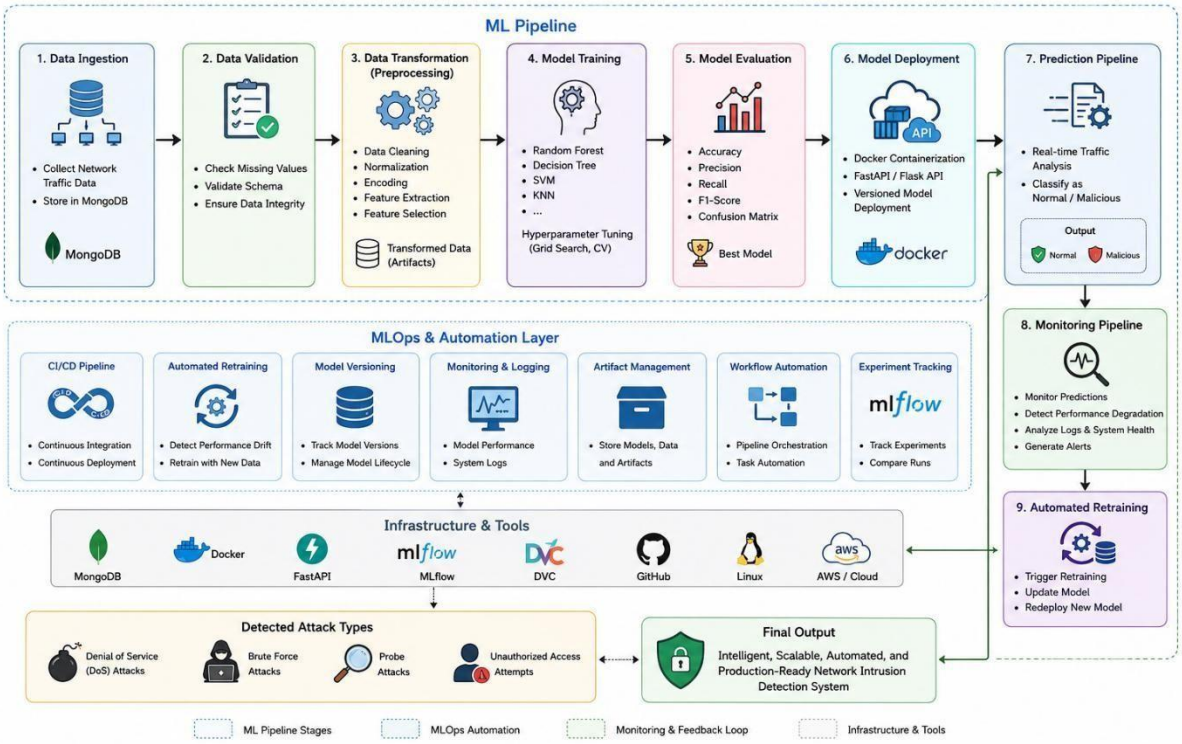

Fig 3.1 Architecture of Proposed MLOps-Based Network Intrusion Detection System

Data ingestion pipeline is tasked with gathering and loading network traffic data to the system. As a database, MongoDB was selected for use in storage and management of network traffic data because of versatility, scalability, and ability of working efficiently with big semistructured data. Data validation pipeline is tasked with validating the integrity of datasets, checking missing data, validating schema consistency, and ensuring proper feature format.

Preprocessing and transformation pipeline includes processes as data cleaning, normalization, encoding, feature extraction, at last feature selection to make datasets ready for analysis using machine learning techniques.

There are many algorithms like Decision Tree, Random Forest, SVM, KNN algorithms that are implemented for intrusion detection purposes. In other words, the algorithm studies the network traffic pattern and then decides whether the activity is normal or malicious, considering the traffic features.

In order to evaluate machine learning model, using several techniques like Precision, Accuracy, Recall, Confusion Matrix, F1 Score are used. In addition, Hyperparameter optimization methodologies such as Grid Search and Cross Validation adopted in order to obtain the best intrusion detection approach.

The main benefits that can be derived from using the suggested model include the application of MLOps to automate machine learning processes. The MLOps process have:

- CI/CD
- retraining
- Model versioning

- Monitoring and logging
- Artifact generation
- Workflow automation
- Deployment management

This system uses containerized deployment with the help of Docker containers and real-time APIs built with FastAPI or Flask frameworks. This platform can also include MLflow or DVC (Data Version Control) for managing experiments and artifacts.

The recommended intrusion detection system will have the capability of detecting real-time attacks and monitor the activities on the network. This monitoring mechanism includes monitoring the performance, behavior, and logs of the systems to make sure that it is sustainable in the long run. In case of performance degradation due to changes in the network traffic, retraining can be done with the help of an updated dataset.

The proposed framework will detect of cyberattacks:

- DoS
- Brute force attack
- Probe attacks
- Unauthorized access attempts

It provides framework as intelligent, automated, and productionready cybersecurity protection for enterprise, cloud-based network environments.

3.4 Advantages of Proposed System

There are several benefits of using MLOps-Based Intelligent Network Intrusion Detection System in comparison with other existing intrustion detection systems. One of the benefits of using system is the possibility to detect using both known and unknown attacks efficiently. Contrary to existing signature-based systems, which rely heavily on previously defined attack patterns, this framework uses ML algorithms, which can detect any patterns within traffic and classify suspicious activity accordingly.

Apart from detecting both known and previously unseen attacks, the proposed system also provides improved accuracy compared to conventional rule-based approaches. Algorithms Random Forest, Decision Tree, SVM, KNN enable system to analyze complex traffic patterns and accurately classify network activities.

Yet another key benefit of the proposed framework is the inclusion of MLOps automation pipelines that would facilitate the end-to-end management of the entire machine learning process. These pipelines automate processes associated with the acquisition of data, preprocessing, training, deployment, monitoring, and retraining of machine learning model throughout the system lifecycle.

Scalability and modularity as designing architecture of the proposed framework would enable its capacity to manage massive traffic volumes in enterprise-level networks, cloud computing architectures, and Internet-of-Things networks. Containerized deployments using Docker and API integration make the intrusion detection system more portable and scalable.

The ability to support real-time monitoring as another key feature of proposed framework. This is because the framework performs continuous analysis on all incoming traffic and raises an alert when there are any signs of malicious activities. Automated monitoring and retraining are key components that keep the intrusion detection model updated by enabling adaptation to emerging cyber threats.

The proposed solution is also capable of providing logging, artifacts creation, experiment tracking, and workflow automation – features that are necessary components of any MLOps architecture.

As we can see, the proposed solution offers an intelligent, scalable, reliable, automated, and production-grade intrusion detection framework.

3.5 Functional Requirements

Functional requirements refer to those functionalities and operational capabilities and designed as expected to perform. Design needs to provide mechanisms that automatically ingest network traffic data sets used in performing intrusion detection operations. The system needs to validate data sets, conduct data preparation steps as data cleansing, normalization, encoding, feature extraction, and feature selection before applying machine learning techniques.

Supervised machine learning algorithms that are capable of classifying network traffic into two distinct classes – normal and malicious – need to be employed within the proposed system. The system must detect different categories of cyberattacks, as DoS, brute-force attacks, probing activities, unauthorized access attempts.

The proposed design can further incorporate:

- Data ingestion pipeline
- Data validation pipeline
- Data transformation pipeline
- Model training pipeline
- Model evaluation pipeline
- Prediction pipeline

The architecture must allow performing automatic tasks of MLOps including deployment, monitoring, retraining, versioning models, logging, artifact creation, and CI/CD pipeline integration. The ability to perform real-time intrusion detection and alerts must be possible in order to see network traffic in real time.

The system must provide prediction services via an API interface, utilizing technologies such as FastAPI or Flask.

3.6 Non-Functional Requirements

It tells the quality and operational characteristics of the well-known system. The system should demonstrate high accuracy, reliability, scalability, maintainability, and efficiency. Achieving high accuracy involves minimizing both false negative, false positive. while at same time ensuring good speed when detecting intrusions.

Scalable processing of high volumes of traffic is another important characteristic that the proposed framework should exhibit. This implies the ability to process network traffic data in enterprise environments, in clouds, and distributed environments. Another key characteristic of the proposed framework is security in order to achieve secure management of the traffic data, models, and deployment pipeline.

Lastly, the proposed framework should have the capability of integrating other advanced features such as machine learning algorithms, deep learning models, cloud-native deployment system, and streaming analytics in the future. This is to allow for integration, extensibility, maintainability, and reproducibility across operating systems.

Incorporating ML-Ops pipelines is expected to enhance the processes of automation, deployment management, re-training, monitoring, and managing the life cycle of the machine learning models in production.

3.7 Software Requirements

The implementation of the project will require multiple programming languages, libraries, frameworks, databases, and MLOps software components. Python has been chosen as the main programming language because of its simplicity and effectiveness in handling machine learning, automation, and data analytics tasks.

Scikit-learn, Pandas, NumPy, Matplotlib, and Seaborn are the libraries used for pre-processing, feature engineering, modeling, evaluations, and visualization purposes. The Jupyter Notebook and Visual Studio Code are among the development, testing, and experimentation environments.

MongoDB has been chosen as the database for data storage related to the collection of the networks' traffic. MLOps includes such tools as Docker, GitHub, MLflow, DVC, FastAPI, and CI/CD pipelines. This project can be deployed on Windows, Linux, and Ubuntu.

3.8 Hardware Requirements

The required intrusion detection system would need sufficient computing resources to handle large volumes, machine learning model training, and deployment activities. The minimum hardware requirements include an Intel Core i5 or higher processor, 8 GB RAM, 256 GB storage capacity, and a stable internet connection.

In cases of more complicated machine learning or deep learning processes, higher requirements like Intel Core i7 and i9 processors, 16 GB and higher of RAM, SSD drive storage, and GPU compatibility can be highly beneficial for computation speed and training processes.
The network connectivity requirement will be also essential for deployment processes.

3.9 Feasibility Study

The feasibility of given intrusion detection framework evaluated order to see whether such a system could be successfully implemented given all current limitations associated with technical, economic, operational, and scheduling aspects.

It is technologically feasible since it is based on the latest technological advancements and uses popular programming languages and libraries including Python, MongoDB, Scikit-learn, Docker, FastAPI, MLflow, and other MLOps solutions. The modular architecture used will provide better manageability of the pipeline.

Economically, the framework is feasible since most of the required technologies are developed by open-source communities. No need for additional costs since the system could be implemented using existing computing resources.

Operationally, the feasibility of the project is associated with increased accuracy of attacks identification, automation, monitoring of processes, effective managing of deployment, and scalability.

Finally, the schedule feasibility of the project is ensured since all the development activities could be carried out within the scope of the allocated project period.

Therefore, the feasibility analysis proved that the developed End-to-End MLOps-Based Intelligent Network Intrusion Detection System would be feasible in terms of technical, economic, operational, and scheduling aspects.

CHAPTER 4
METHODOLOGY AND SYSTEM DESIGN

4.1 Introduction

The methodology and system design will determine the necessary architecture, flow, approach to implementation, and system components needed for building the proposed intrusion detection system. In today's cybersecurity landscape, the intrusion detection system needs to perform functions beyond attack detection such as automation, scalability, deployment, monitoring, and lifecycle management. Traditional intrusion detection systems tend to only cater to attack detection and do not adequately cater to automation and deployment.

The project under discussion aims to develop an End-to-End MLOps-Based Intelligent Algorithms and MLOps automation pipeline. The solution combines different components of data processing, such as ETL pipelines, data validation, preprocessing, machine learning models, deployment pipeline, monitoring, and retraining into one architecture.

The proposed system utilizes a modular pipelined architecture, which is based on recent MLOps designs. The system includes automated pipelines that cover the complete lifecycle of a machine learning workflow, starting from data ingestion, validating data, pre-processing data, training models, deploying models, making predictions, monitoring, and retraining. Technologies such as MongoDB, Docker, FastAPI, GitHub, and CI/CD pipelines are used within the framework to enhance the scalability, reproducibility, and portability of the project.

This chapter will introduce the overall system architecture, workflow of the proposed system, project folder structure, data ingestion pipeline, data validation pipeline, transformation pipeline, model trainer pipeline, prediction pipeline, artifact generation, MongoDB workflow, Docker architecture, logging, exception handling, and deployment workflow.

4.2 Overall System Architecture

MLOps architectural structure that automates the entire process of ML development and implementation cycle for cybersecurity purposes. An ML application typically includes multiple components that deal with data acquisition, preprocessing, training, deployment, predicting, and monitoring of the deployed model.

At the very beginning of the ML process, we acquire intrusion detection datasets, including NSL-KDD and CICIDS2017. All obtained datasets are saved in the central MongoDB database. In order to proceed with further actions, the Data Ingestion pipeline takes datasets from the central database and feeds them to the ML process. The next stage includes validating of datasets where their quality is checked in terms of schema conformity, presence of missing/duplicate records and data types.

Data Transformation Pipeline includes tasks related to data preprocessing, such as normalization, encoding, feature extraction, and feature selection. Transformed data is used by the Model Training Pipeline for machine learning model training and performance evaluation.

Deployment of the machine learning model is deployed using Docker containers and FastAPIbased prediction services. Prediction Pipeline, which is responsible for processing network requests in real-time. Continuous monitoring and logging of system performance are performed by the Monitoring & Logging Pipeline, which automatically triggers retraining of the models.

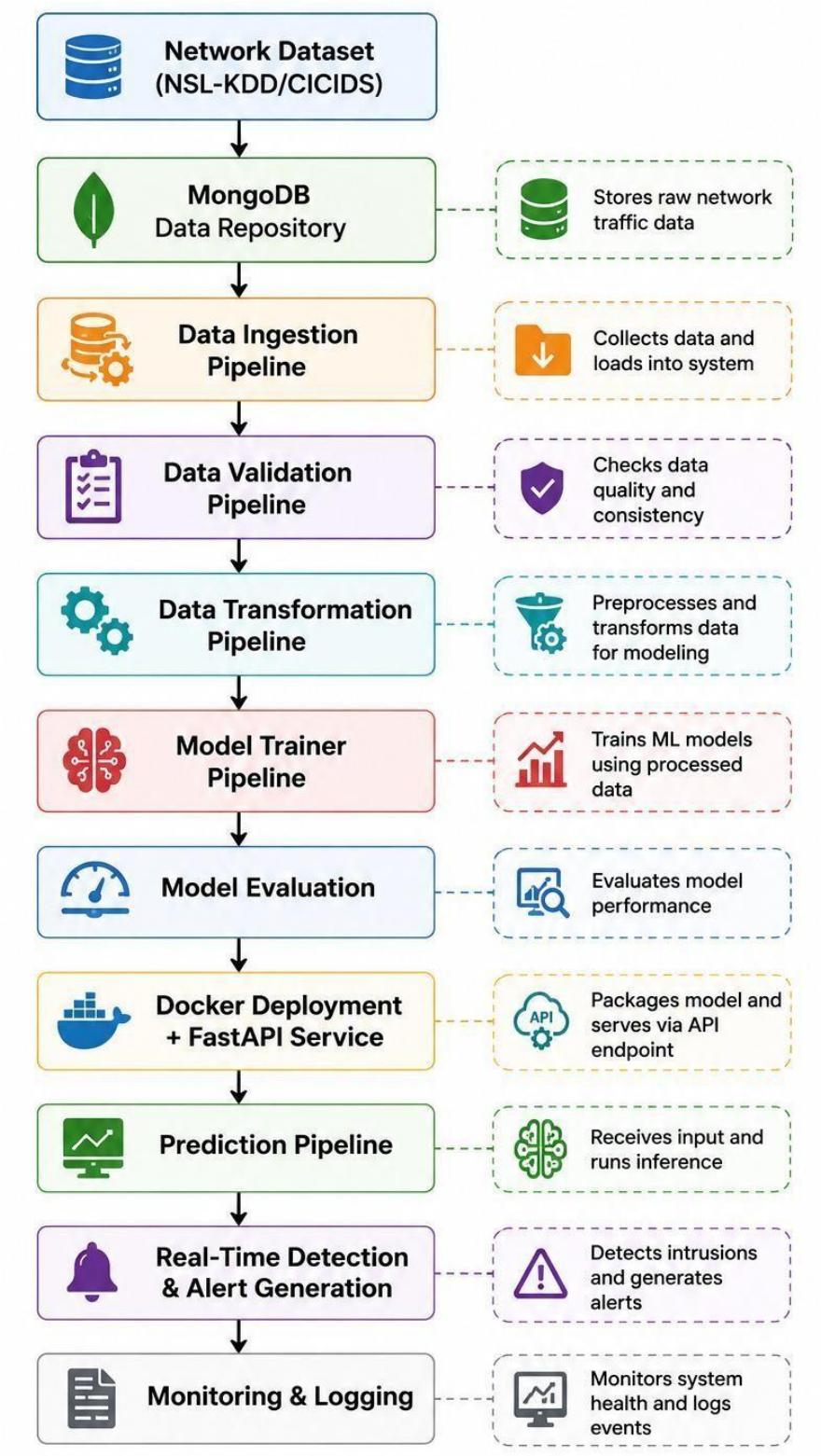

Fig 4.1: Architecture of Proposed Intrusion Detection System

It illustrates the given MLOps-based Intelligent Intrusion Detection System. Framework proposed framework employs a modular pipeline-based approach to automate the entire machine learning process cycle for the use in cybersecurity. The pipeline commences with the collection of intrusion detection datasets like NSL-KDD and CICIDS, storing them in MongoDB databases. Following that, the data undergoes ingestion, validation, processing, and transformation procedures. Machine learning models are trained using the preprocessed data. After training, the system evaluates the performance of the models and deploys the selected model using Docker containers and FastAPI services for prediction and intrusion detection tasks.

4.3 Workflow of Proposed System

It is used to automate the entire machine learning life cycle. This starts with data acquisition from freely available intrusion data sets from networks. Datasets obtained are then stored within MongoDB databases.

Data Ingestion Pipeline processes the datasets by pulling them out from the MongoDB databases into the pipeline process of data analysis. Data Validation Pipeline helps to ensure that the schema integrity is checked, check for missing values, duplication of records, and validation of data types.

Data Transformation Pipeline helps to conduct pre-processing activities data cleaning, data normalization, encoding, feature extraction, feature selection on the datasets.

The processed datasets are provided to the Model Trainer Pipeline, where machine learning models are trained and evaluated.

Deployment of the model is done using Docker Containers and FastAPI Services. All the incoming traffic is analyzed in the Prediction Pipeline to detect any intrusion at runtime. At the same time, the Monitoring and Logging Pipeline constantly monitors prediction performance, deployment status, and operation logs. Whenever the performance of the deployed model starts decreasing because of any change in traffic pattern, retraining process automatically updates the deployed model.

The suggested MLOps-Based Intelligent Network Intrusion Detection System implements a workflow that automates the whole machine learning life cycle to build cybersecurity solutions. The architecture combines several processing modules to carry out Data collection, storage, preprocessing, feature extraction, model training, deployment, prediction, monitoring tasks enable scalable and robust intrusion detection systems. Every module within the suggested workflow has its role in converting raw traffic data into intelligent prediction results.

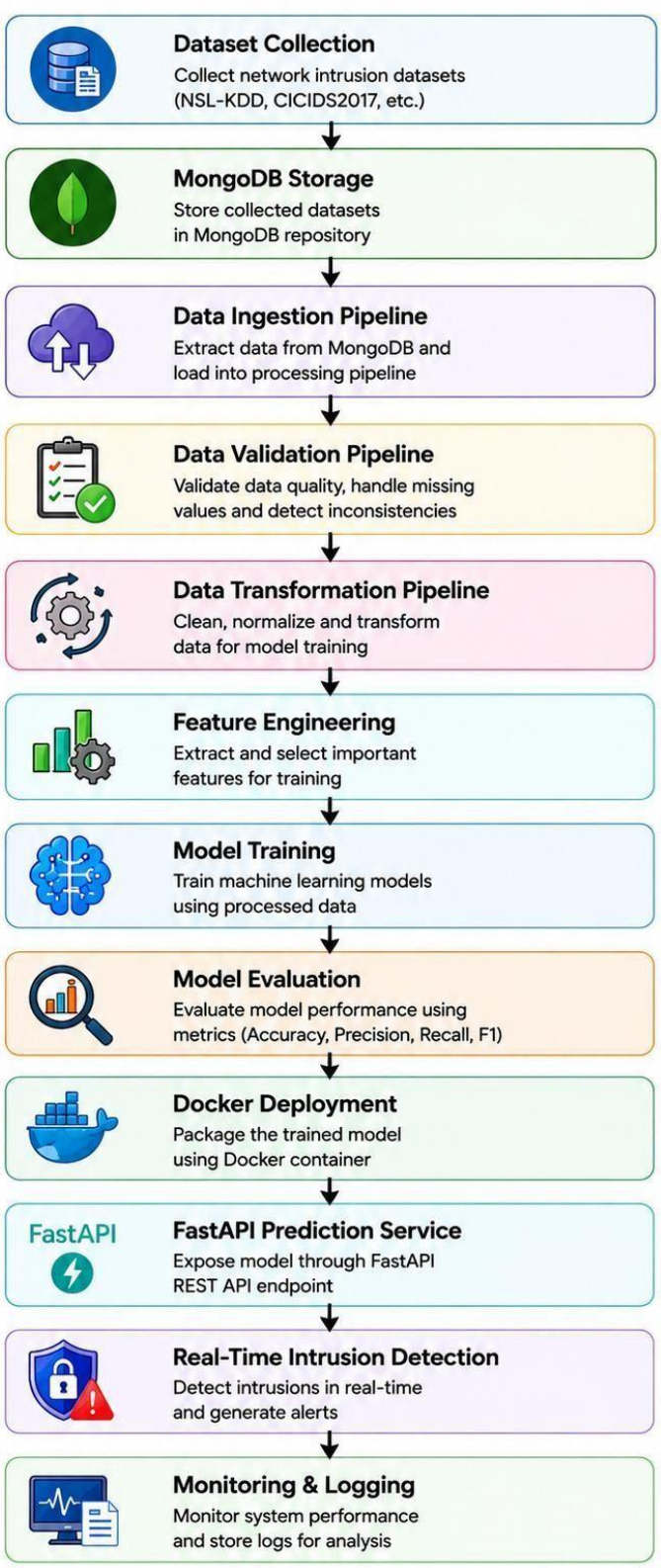

Fig 4.2 Workflow of Proposed Intrusion Detection System

4.4 Project Folder Structure

The suggested project will have a modular folder system, based on a modern MLOps design. The advantage of a modular approach is better maintainability, scalability, reusability, and clear division of tasks among the various elements of the intrusion detection system.

The project structure comprises directories containing distinct modules for pipelines, preprocessing, deployment, monitoring, logging, configuration management, and utility methods. The components directory holds machine learning related components including data ingestion, validation, transformation, and model training components. The pipeline directory manages pipeline orchestration and workflow execution.

The exception and logger directories enable centralized logging and exceptions management respectively. The utils directory houses shared utility methods used across the project. The artifacts directory keeps transformed datasets, trained models, preprocessing pipeline, and other outputs needed during deployment and retraining phases.

The Dockerfile enables containerized deployment, while requirements.txt contains the necessary dependencies and libraries for the project. Config.yaml is used to keep configurable pipeline parameters and environment configurations.

Project Folder Structure:

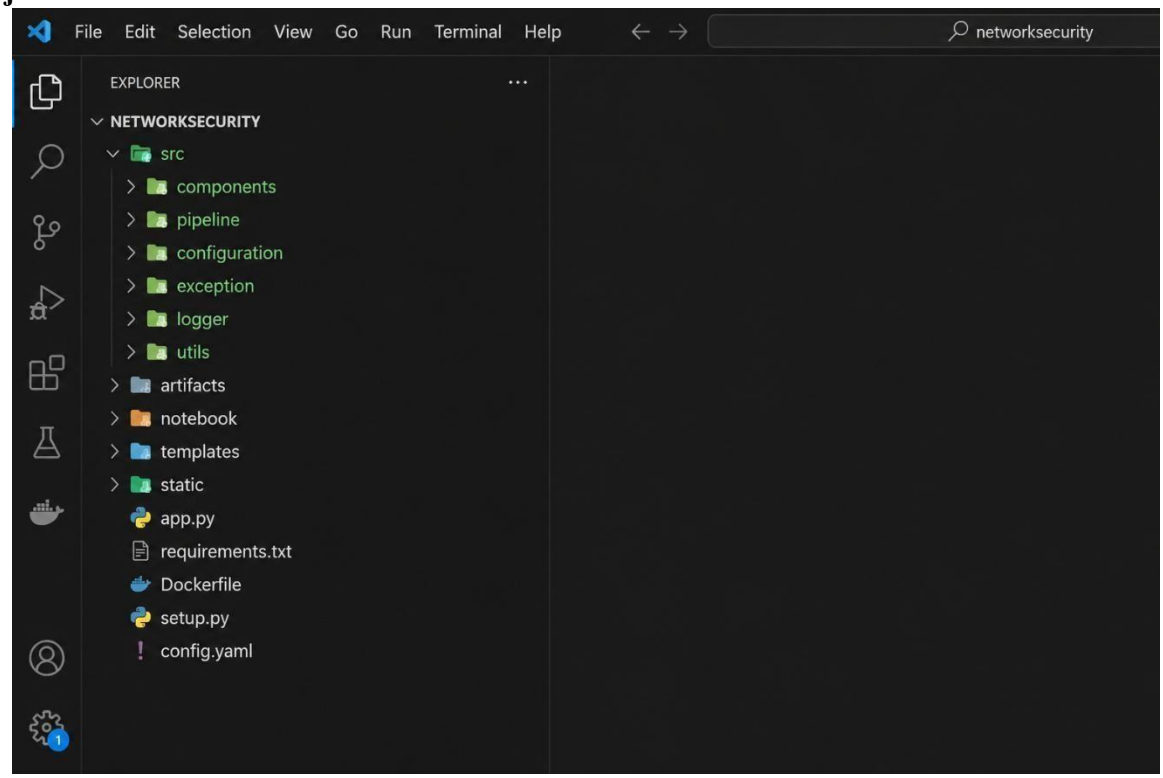

Fig 4.3 Project Folder Structure

4.5 Data Ingestion Pipeline

The process of fetching intrusion detection datasets as well as uploading them into the machine learning pipeline. The Data Ingestion Pipeline fetches datasets stored in MongoDB databases and uploads them to the processing pipeline.

The data ingestion pipeline helps improve automation and minimizes manual efforts in managing the datasets. In addition, the ingestion pipeline is also used to create dataset artifacts that are necessary during the processes of preprocessing and training. Effective data ingestion leads to better scalability and efficiency.
The data ingestion pipeline facilitates the process of data management within a central location.

4.6 Data Validation Pipeline

Validation pipeline is used to verify the integrity and consistency of data before any preprocessing actions are taken. The process of validation entails validation of schema consistency, missing data points, duplicate data entries, invalid data, and invalid data types.

The purpose of this pipeline is to ensure that only data that is free from inconsistencies is used for processing in the future.

Validation improves reproducibility workflow and less risk of train machine learning models on corrupted or inconsistent data.

4.7 Data Transformation Pipeline

Data Transformation Pipeline responsible for pre-processing and transformation of validated intrusion detection data sets. The raw data set usually consists of noisy, missing, inconsistent, and categorical data that cannot be directly processed by any machine learning algorithm.

Some of the operations that are carried out in the pipeline include data cleaning, normalization, encoding, feature extraction, and feature selection. Some of the techniques utilized include Label Encoding and One-Hot encoding.

Scaling methods are applied to minimize feature imbalance. The transformed datasets are stored as artifacts and reused during training and deployment stages.

4.8 Model Trainer Pipeline

The Model Trainer Pipeline is responsible for training machine learning models using transformed intrusion detection datasets. The pipeline automatically splits datasets into training and testing subsets before initiating model training operations.

Machine learning algorithms including Random Forest, Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) are trained using the network traffic features obtained during the preprocessing stage. The trained models are then evaluated using performance metrics such as accuracy, precision, recall, and F1-score.

Cross-Validation can also be applied to increase classification performance, minimize the overfitting problems. The best-performing model selected and saved as a trained model artifact for deployment.

4.9 Prediction Pipeline

The Prediction Pipeline is responsible for processing incoming network traffic and generating intrusion detection predictions using the deployed machine learning model. Incoming traffic features are received through API requests and passed to the prediction model for classification.

Based on prediction results, the system identifies whether network traffic is normal or malicious. Real-time intrusion detection is possible through the prediction pipeline, which allows for constant monitoring of the activities on the network.

This prediction service can be implemented in the enterprise cybersecurity infrastructure to provide automated attack detection and alerts.

4.10 Artifact Generation

Artifacts are generated at different stages of the machine learning pipeline to store both intermediate outputs and final processing results.

The artifacts that are created during processing include the following: preprocessed data, transformed data, machine learning model training, pre-processing artifacts, performance evaluation, and predictions. The above-listed artifacts are stored under the artifact folder.

Artifact management makes workflow easier and allows us to build the MLOps deployment architecture.

4.11 MongoDB Workflow

The MongoDB database is employed as a single source for intrusion detection datasets. This database ensures scalability in terms of storage capacity and efficiency in dealing with semistructured data from the traffic on the network.

Firstly, intrusion detection datasets are inserted into MongoDB collections. Then, the Data Ingestion Pipeline pulls these datasets from MongoDB and moves them to preprocessing and training pipelines.

Using MongoDB offers scalability, centralized database management, real-time database handling, and automation of pipelines. Moreover, this database facilitates fast data extraction and storage tasks involved in machine learning workloads.

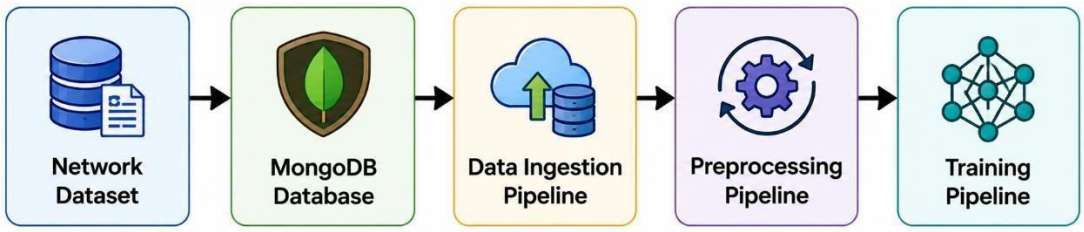

Fig 4.4 MongoDB Data Workflow

The following figure shows the data flow process that has been adopted in this paper while designing the intrusion detection model with the help of MongoDB databases. In the first stage of the proposed model, we gather intrusion data sets, which will be saved inside the MongoDB database. After saving the data set, it can be accessed via the data ingestion process, where the data sets are extracted and imported. The data is prepared for modeling purposes via the preprocessing process, which includes cleaning, normalization, and transformation of data sets. After preprocessing, the datasets are forwarded to the training phase to develop an intelligent intrusion detection model.

4.12 Docker Architecture

Docker is used for containerizing the intrusion detection framework and simplifying deployment across multiple environments. Containerization ensures portability, scalability, reproducibility, and consistency between development and production environments.

The trained machine learning model and FastAPI application are packaged inside Docker containers. Docker isolates project dependencies and simplifies deployment in productionlevel cybersecurity infrastructures.

Docker architecture improves deployment portability, environment consistency, resource isolation, scalability, and CI/CD integration.

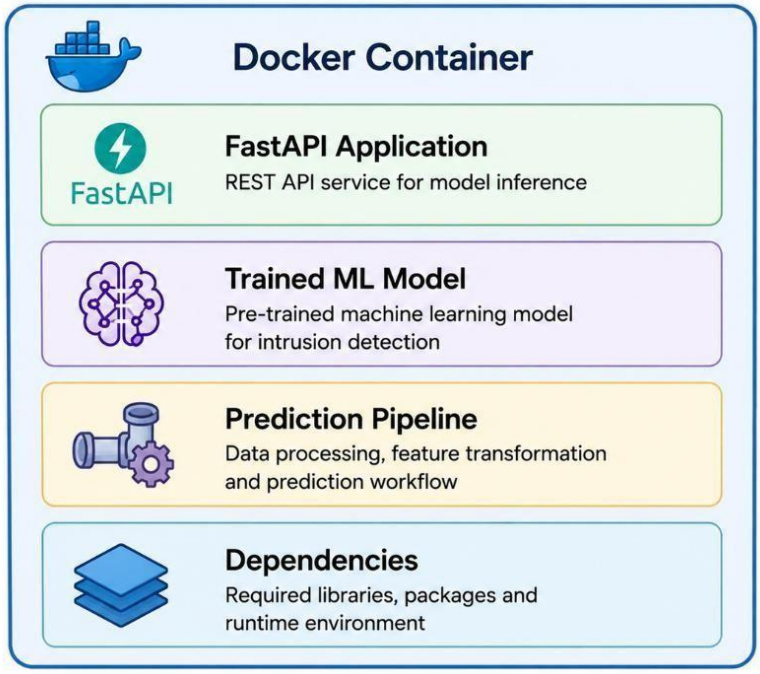

Fig 4.5 Docker Deployment Architecture

Figure 4.5 illustrates the Docker architecture used in the proposed intrusion detection framework. The Docker container encapsulates the FastAPI application, trained machine learning model, prediction pipeline, and all required dependencies within a single isolated environment. This architecture ensures portability, scalability, deployment consistency, and efficient execution of the intrusion detection system across different platforms and production environments.

4.13 Logging and Exception Handling

The proposed framework integrates centralized logging and exception handling mechanisms for monitoring pipeline execution and debugging operational failures. Logging modules continuously store pipeline execution logs, error logs, model training logs, deployment logs, and prediction logs during workflow execution.

Exception handling improves reliability and simplifies debugging during preprocessing, training, deployment, and prediction phases. Logging and exception handling tells how much system is stable

Monitoring, logging functions can also be used to detect performance degradation and problems during real-time intrusion detection.

4.14 Deployment Workflow

This automation framework uses Docker containers and API-based prediction services to automatically deploy machine learning models. The intrusion detection model that was built is packaged using Docker containers and deployed using FastAPI services for real-time predictions.

CI/CD is used to automate the process of deploying this application. This will increase its scalability, reusability, efficiency, and manageability.
The deployment process ensures automated deployment, real-time intrusion detection, continuous monitoring, CI/CD implementation, and scalable deployment at the production level.

CHAPTER 5
IMPLEMENTATION

5.1 Introduction

Implementation is one of the most important phases in the development of the intrusion detection framework under consideration since it involves translating the MLOps architecture design into an operational framework. In the course of implementation, several actions will be taken such as environment configuration, developing pipelines, integrating machine learning, setting up APIs, creating containers, and putting in place monitoring systems for intrusion detection.

The present project proposes implementing an End-to-End MLOps-Based Intelligent Network Intrusion Detection System utilizing Whereas traditional machine learning projects are concerned mainly with model training, the implementation of the project under discussion focuses much on automation, modular design, deployment, logging, and scalability in MLOps just like GitHub projects.

Implementation includes various interconnected components such as MongoDB connectivity, data ingestion pipeline, validation pipeline, transformation pipeline, model training pipeline, prediction service, Dockerization, logging mechanism, exception handling, and CI/CD pipelines. The implementation utilizes a number of programming languages such as Python and various machine learning, deployment, and MLOps tools.

This Chapter provides the implementation details for the proposed framework in the context of environment configuration, tools and technologies utilized, pipeline implementation, deployment architecture, Dockerization, API development, CI/CD pipelines, monitoring systems, and logging mechanisms.

5.2 Environment Setup

The environment setup of the intrusion detection architecture, as a properly configured environment ensures compatibility between machine learning libraries, deployment platforms, databases, APIs, and automation processes.

For the proposed solution to be implemented, Python is used to improve, flexibility, set packages for machine learning, automation, and cybersecurity analytics. Experiments and development take place using Jupyter Notebook, Visual Studio Code, and PyCharm environments.

Deployment is supported on operating systems such as Windows, Linux, and Ubuntu according to specific needs. Operating systems based on Linux should be deployed since they have better compatibility with Docker, CI/CD, and MLOps.

The Python environment configuration will be done to ensure that all dependencies will be managed well without any conflict between installed libraries. Pip and requirements.txt files will be used to install the required packages and frameworks. Git and GitHub will be used for source code management.

The infrastructure also comprises MongoDB databases, Docker, FastAPI applications, and logging modules for supporting real-time intrusion detection and deployment processes. Adequate hardware facilities like CPU architecture, memory, storage, and network connectivity are also configured for supporting the big data infrastructure and large-scale machine learning processes.

5.3 Tools and Technologies Used

The deployment framework tells the use of multiple programming languages, libraries, frameworks, databases, and software development tools for tasks such as data preprocessing, machine learning model implementation, and automation.

Python serves as the PL in use because of the availability features related to machine learning and data analysis among others. Pandas and NumPy serve as some of the main libraries that are used to process the raw data. Scikit-learn serves as the library used to implement machine learning algorithms.

MongoDB serves as the main database that is used in handling intrusion datasets. This is mainly because MongoDB is known for its scalability features.

For constructing RESTful APIs to deploy ML-based IDS models on-the-fly, FastAPI and/or Flask frameworks could be leveraged. For packaging and deploying the application, containerization techniques, such as those based on Docker, will be employed.

As version control tools, Git/GitHub can be used to keep track of changes and manage source codes. CI/CD pipeline solutions like GitHub Actions and/or Jenkins could be integrated, and logging and monitoring capabilities included for monitoring system actions, prediction results, and other types of events, including failures.

The incorporation of these technologies will lead to better scalability, reproducibility, portability, automated deployment, and operational performance.

5.4 Project Pipeline Implementation

The intrusion detection framework that is implemented employs a modular pipeline-based approach based on the architecture currently used in MLOps. The framework comprises several connected pipelines designed to perform specific tasks.

Ingestion is the 1st stage in the implementation process, during which intrusion detection datasets will be retrieved from MongoDB databases and moved into the ML workflow. Dataset ingestion will be performed by the automatically.

Next, the Validation ensures accuracy of data schema, data types, absence of missing values, duplication, and overall consistency of each dataset before moving to data transformation procedures. This step ensures data will be for model training.

Last preprocessing stage involves the Data Transformation Pipeline, where all necessary preprocessing steps like normalization, encoding, feature extraction/selection will be performed automatically.

The most effective model is then selected and stored as a trained machine learning model. In the Prediction Pipeline, the incoming network traffic is processed to make predictions regarding the possibility of an intrusion in the network by using the implemented machine learning model.

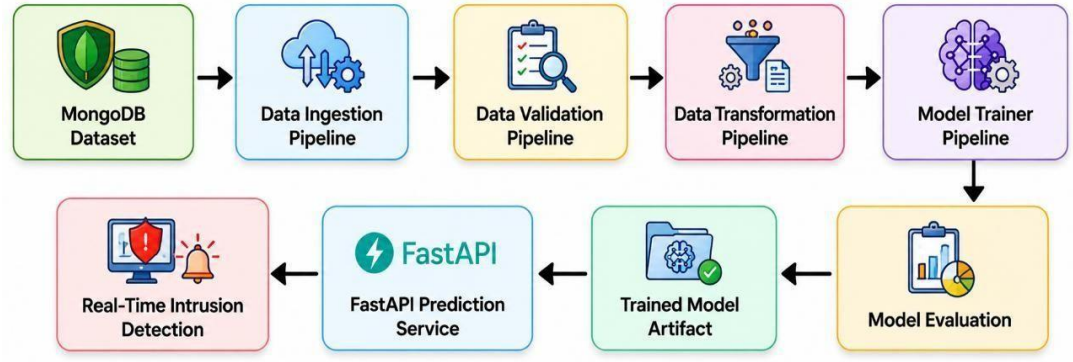

Fig 5.1 Pipeline Implementation Workflow

5.5 MongoDB Integration

The MongoDB database is incorporated into the intrusion detection framework that is being proposed to provide a platform where datasets related to intrusion detection can be stored and managed centrally.

The process starts with loading the datasets into collections within the MongoDB database. The Data Ingestion Pipeline collects the datasets stored in MongoDB and loads them into preprocessing and training pipelines automatically.

Integrating MongoDB into the system will help improve:

- Centralized data management
- Scalability
- Data retrieval efficiency
- Real-time data handling
- Pipeline automation

The database architecture also supports future integration with streaming network traffic and real-time intrusion detection systems. MongoDB plays an important role in supporting scalable MLOps workflows and automated machine learning operations.

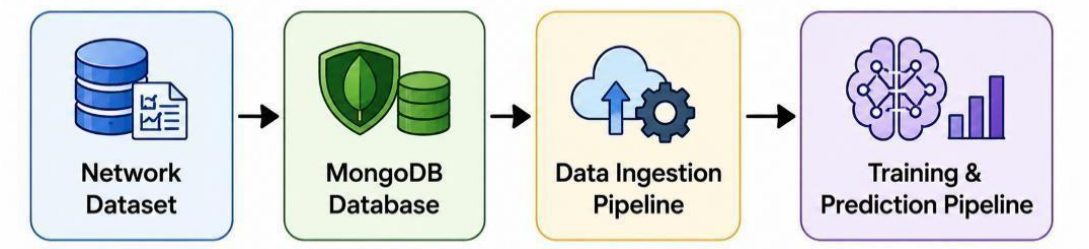

Fig 5.2 MongoDB Integration Workflow

5.6 API and Prediction Service Implementation

Framework proposed utilizes Fast API/Flask framework to create API-based predictions for the detection of intrusions in real-time.

In this regard, the prediction API takes network traffic data from HTTP requests and feeds them to the machine learning model for classification. Depending on the prediction output, the API classifies

APIs consist:

- Real-time prediction
- Scalable deployment
- Fast response handling
- Integration with monitoring systems
- Automated intrusion detection

The API implementation enables practical deployment of the intrusion detection framework in enterprise and cloud-based cybersecurity environments.

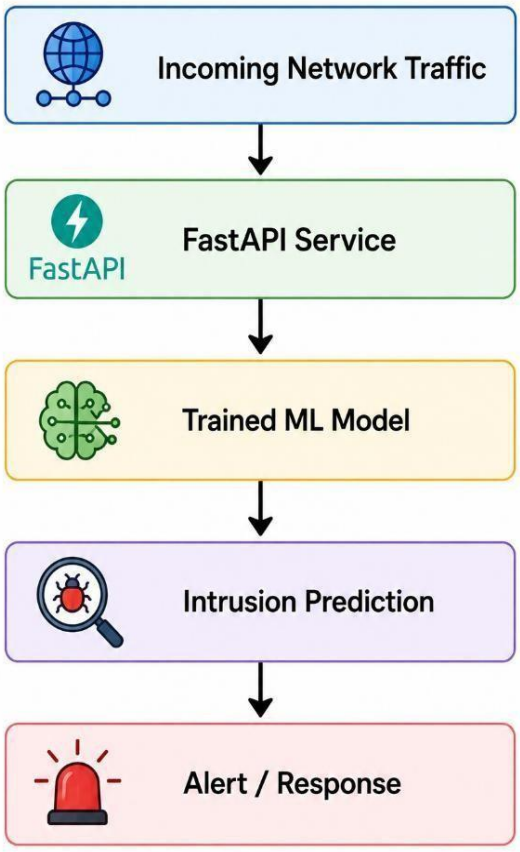

Fig 5.3 API Prediction Workflow

5.7 Docker Deployment Implementation

Docker containers can be employed to package the intrusion detection system framework. Package creation through the use of Docker containers helps achieve portability, scalability, consistency, and ease of replication.

All the components that will run in the container include the trained machine learning model, FastAPI application, dependency packages, and prediction services. The components within Docker containers help isolate the project dependencies.

Benefits of using Docker Deployment Include:

- Environment consistency
- Portability
- Resource isolation
- Deployment automation
- Scalability
- CI/CD integration

The Dockerfile defines the application environment, dependencies, execution commands, and deployment configuration required for containerized execution.

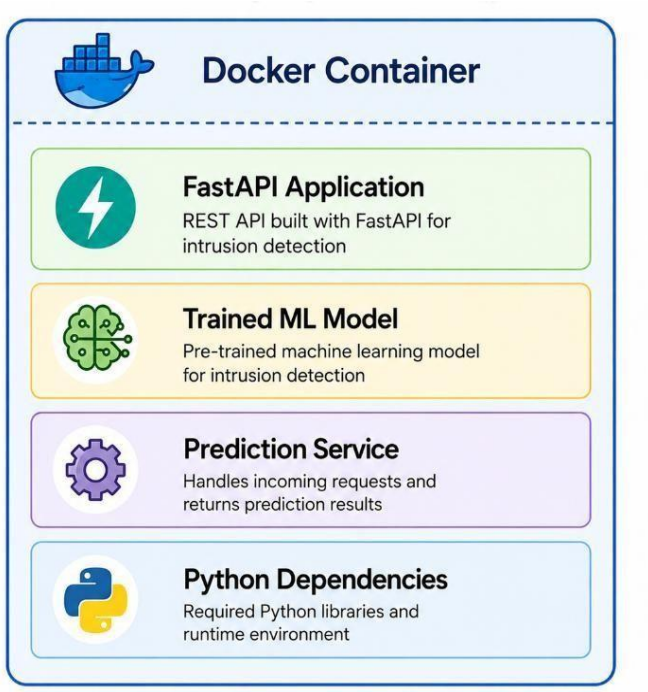

Fig 5.4 Docker Deployment Architecture

5.8 Logging and Exception Handling

In this regard, the new framework is aimed at adopting centralized logging and exception handling functionalities to monitor and debug errors arising during the process.

Logging includes:

- Pipeline execution logs
- Error logs
- Model training logs
- Prediction logs
- Deployment logs
- API request logs

In addition, custom exception handlers are created to enhance reliability and assist in debugging problems related to preprocessing, training, deployment, and prediction.

Additionally, the use of logging facilitates visibility in the process, making it easier for administrators to detect issues such as errors while deploying the project, errors at runtime, and predictions made during testing.

5.9 CI/CD Integration

Integration of CI/CD is used to automate software testing and deployment processes within the proposed intrusion detection framework.

GitHub or GitHub Actions or Jenkins can be used to manage source code and perform deployment of the code automatically. Every time any changes are introduced into the machine learning process or deployment scripts, these workflows trigger the deployment process.

Continuous Integration allows one to detect errors or compatibility issues early while the software is being developed. With Continuous Deployment, the process of deployment and updating of models becomes automated.

In addition, when new data sets become available or models degrade in their performance, the CI/CD pipeline allows performing automated retraining and deployment.

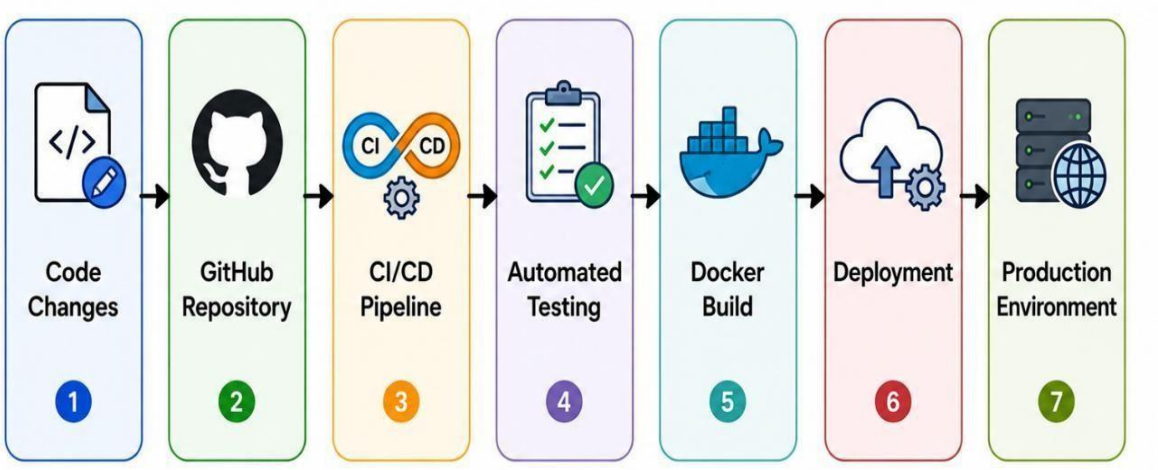

Fig 5.5 CI/CD Deployment Workflow

5.10 Monitoring and Real-Time Detection

MLOps-based architecture as they provide for the constant assessment of model's performance and operations.

Once a machine learning algorithm is deployed, it constantly monitors the traffic on the network and categorizes network activity as either normal or malicious. The logging and monitoring process assesses the accuracy of the predictions, reaction times, deployment, and operational performance.

As soon as suspicious and malicious behavior is discovered, an alert about the intrusion. Mechanisms tell one detect model drifts and performance degradation in light of everchanging attacks.

The framework allows for automated retraining processes in case of model degradation. Thus, continuous monitoring improves cybersecurity's reliability, visibility, scalability, and performance.

CHAPTER 6
RESULTS AND PERFORMANCE ANALYSIS

6.1 Introduction

Result and performance analysis stage is an integral part of the implementation proposed MLOps- intelligent framework provides the analysis of the performance of the developed solution, its deployment capabilities, prediction capabilities, and workflow execution. Unlike other intrusion detection frameworks, the proposed solution pays attention not only to machine learning classification performance but also to automation of workflow execution, deployment process, API integration, Docker deployment, monitoring, and real-time prediction capabilities.

The developed framework incorporates various components of MLOps, such as data ingestion pipeline, validation pipeline, transformation pipeline, training pipeline, prediction service, MongoDB integration, Docker deployment, logging, and CI/CD workflows. The implemented solution is assessed based on classification machine learning metrics, practical outputs related to the deployment process, and workflow execution.

The chapter provides information regarding the setup, execution results, and analysis of pipelines used for development, analysis of model performance, API prediction results, Docker deployment output, logging and monitoring analysis, CI/CD workflow execution, and general discussion of the proposed intrusion detection framework.

6.2 Experimental Setup

The experimental design refers to the environment, tools, data sets, and configuration that are required mechanism. The proposed algorithms, cybersecurity analysis, and deployment purposes has capibilities for automations.

Pandas, NumPy, and scikit-learn libraries are employed to preprocess the data, perform feature engineering, and implement machine learning algorithms. FastAPI is utilized for providing API-based prediction services, whereas MongoDB is employed to store intrusion detection data sets. Docker is used for implementing the machine learning application in a containerized environment.

The intrusion detection data sets used in the implementation process include publicly accessible datasets have records of normal and malicious traffic belonging to different types of attacks, namely:

- DoS
- Probe attacks
- Brute force attack
- Unauthorized access attempts

Implementation environment also includes GitHub for source code management, CI/CD integration for deployment automation, and logging modules for monitoring workflow execution and operational activities.

These experiments are performed on machines that have enough hardware capabilities, such as multi-core processors, memory, storage space, and Internet connection needed to conduct machine learning.

6.3 Pipeline Execution Results

The proposed architecture efficiently accomplishes each step of the MLOps workflow process starting from data ingestion to prediction. The use of a modular MLOps workflow design ensures efficiency, scalability, maintenance, and automation of the intrusion detection framework.

The Data Ingestion Pipeline efficiently ingests intrusion detection datasets from MongoDB databases into the data processing workflow. The Data Validation Pipeline then effectively validates the datasets ensuring their integrity, data consistency, and data type validation before preprocessing activities.

The Data Transformation Pipeline efficiently performs dataset preprocessing tasks like normalization, feature encoding, and feature transformation. The processed data artifacts are saved to be used for model training and deployment.

The Model Trainer Pipeline efficiently trains algo's preprocessed and saves the model artifacts inside the artifacts folder. The Prediction Pipeline efficiently processes network traffic and makes real-time intrusion detection predictions.

Therefore, the success of the entire MLOps workflow proves the effectiveness of the proposed architecture.

Pipeline Execution Output:

```
[2025-05-25 10:15:30] [INFO] - Data Ingestion Completed Successfully
[2025-05-25 10:15:35] [INFO] - Data Validation Completed Successfully
[2025-05-25 10:15:40] [INFO] - Data Transformation Completed Successfully
[2025-05-25 10:15:50] [INFO] - Model Training Started
[2025-05-25 10:18:22] [INFO] - Best Model Selected: Random Forest
[2025-05-25 10:18:25] [INFO] - Model Evaluation Completed
[2025-05-25 10:18:27] [INFO] - Model Saved Successfully
[2025-05-25 10:18:30] [INFO] - Prediction Pipeline Initialized
[2025-05-25 10:18:32] [INFO] - Deployment Successful
user@ml-pipeline:~$
```

Fig 6.1 Pipeline Execution Output

6.4 Model Performance Results

The performance of the machine learning models used in the suggested intrusion detection approach is measured using classification measures like accuracy, precision, recall, and F1

score. The experiment results show that the designed machine learning algorithms efficiently categorize network activities into either normal or malicious categories.

Random Forest demonstrates the highest efficiency among all tested algorithms due to their ensemble learning properties and robustness against the noise associated with intrusion detection data sets. Decision Tree and Support Vector Machine algorithms have also demonstrated efficient classification properties; however, KNN shows slower processing speed for large data sets.

Experiment results show that preprocessing, feature engineering, and automation of the process lead to higher intrusion detection performance.

Model Performance Table:

Table 6.1: Difference of ML Algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
Decision Tree	96.4%	95.8%	96.1%	95.9%
Random Forest	99.1%	98.9%	99.0%	98.8%
SVM	97.3%	96.7%	97.0%	96.8%
KNN	95.6%	94.9%	95.2%	95.0%

6.5 API Prediction Results

The designed intrusion detection system utilizes FastAPI-based prediction services for implementing intrusion detection capabilities. The prediction API accepts the features of incoming network traffic via HTTP requests and forwards them as train MLclassification.

Implemented API is capable accepting prediction requests and generating classification results in regard to network traffic being classified as either normal or malicious. This implementation allows the prediction API for real-time intrusion detection capability within enterprise cybersecurity scenarios.
Fast response generation and efficient integration of the deployed machine learning model and prediction API were observed.

API Prediction Output:

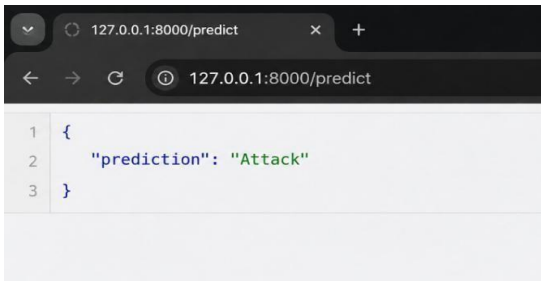

Fig 6.2 FastAPI Prediction Output

API Workflow Diagram:

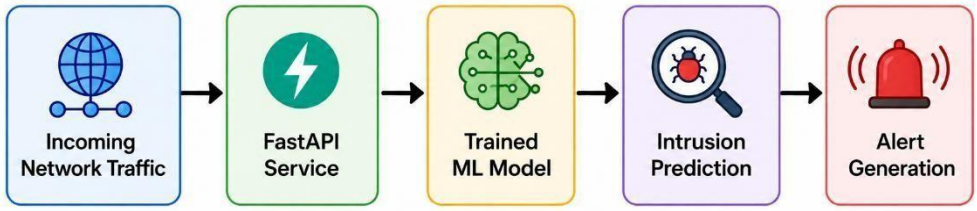

Fig 6.3 API Prediction Workflow

6.6 Docker Deployment Results

Intrusion Detection framework has been implemented using Docker containers for containerization and easy deployment.
This machine learning model with all necessary services and APIs are included within docker containers for easy deployment.

The implementation of Docker deployment makes the process of environment consistency, portable deployment and automation easier.
The application running within docker container is able to run successfully and provide real time Intrusion Detection services.

Docker Deployment also helps improve integration with CI/CD deployment processes.

Docker Deployment Output:

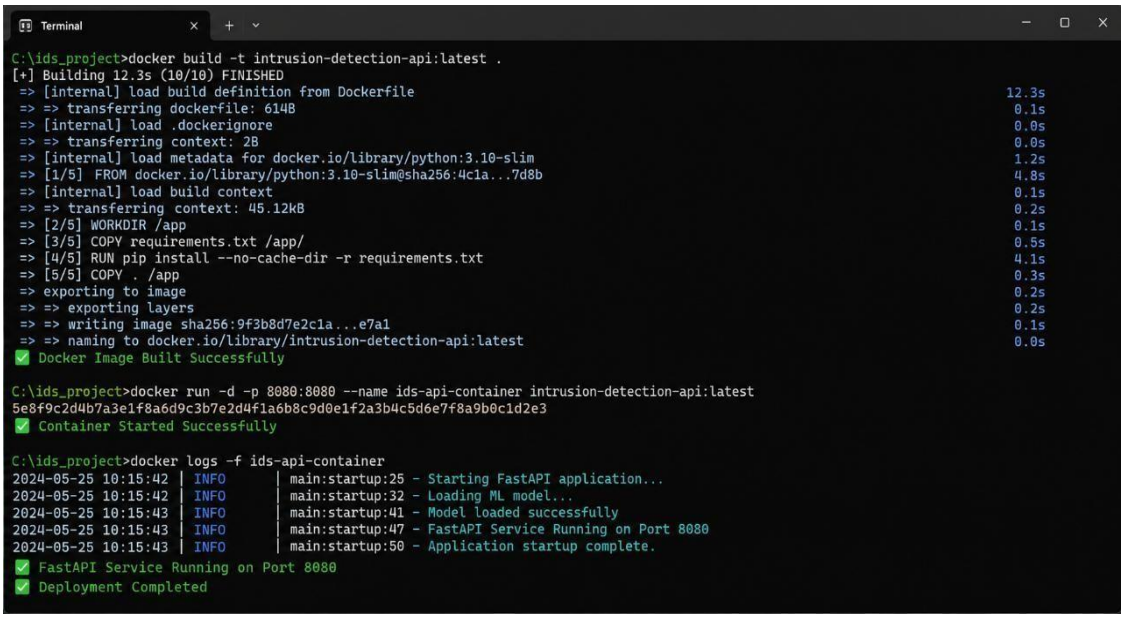

Fig 6.4 Docker Deployment Output

Docker Architecture Diagram:

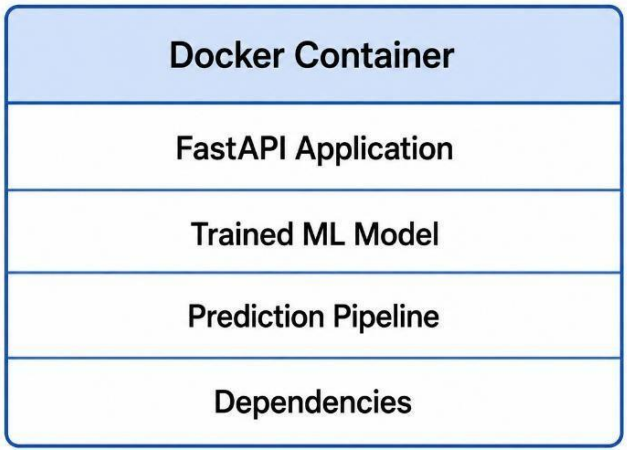

Fig 6.5 Docker Deployment Architecture

6.7 Logging and Monitoring Results

The suggested framework consists of a centralized logging and monitoring framework that helps log the execution of pipelines, predictions, deployment, and operations. The logging modules log preprocessing, training, deployment, prediction, and runtime exceptions throughout the execution process.

The monitoring framework efficiently tracks the operational performance, enabling the identification of failures, prediction errors, and deployment problems during the execution process. Effective monitoring increases system reliability and transparency.

The logging framework also plays an important role in debugging and troubleshooting of failed pipeline executions during the training and deployment processes.

Logging Output:

```
(venv) user@ids-project:~$ python main.py
2025-05-25 10:30:15,231 | INFO   | Data Ingestion Started
2025-05-25 10:30:18,924 | INFO   | Data Validation Completed
2025-05-25 10:30:45,671 | INFO   | Model Training Completed
2025-05-25 10:30:48,112 | INFO   | FastAPI Service Started
2025-05-25 10:31:02,334 | INFO   | Prediction Generated Successfully
```

Fig 6.6 Logging Output

6.8 CI/CD Workflow Results

The suggested framework allows CI/CD pipelines for automation in deployment and the execution process. GitHub and GitHub Actions can be utilized for managing the source code and for automating deployments.

When any modification is made to the repository of the project, the CI/CD process starts with testing, Docker build actions, and deploying. Automation of deployment makes it more scalable, reproducible, maintainable, and effective.

Moreover, the proposed CI/CD pipeline can also enable automated redeployment of the machine learning models when retraining is done.

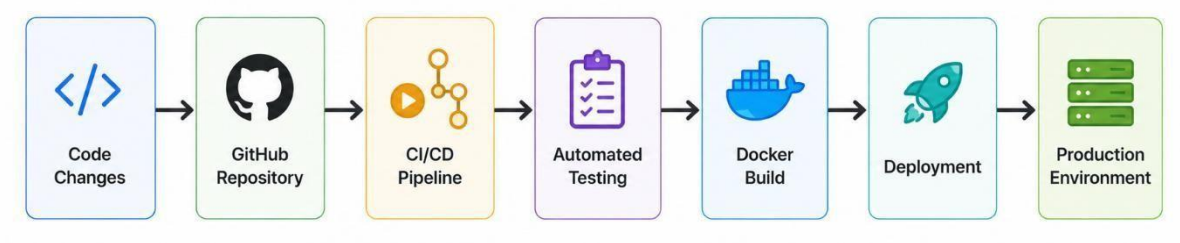

Fig 6.7 CI/CD Workflow Diagram

6.9 Discussion of Results

These findings clearly show that the MLOps-Based Intelligent Network Intrusion Detection System suggested by the author is an efficient combination of such elements of machine learning as classification, automated pipelines, deployment, APIs, Dockerization, logging, monitoring, and CI/CD, which allows implementing them to cybersecurity issues.

The use of the modular pipeline approach makes the system more scalable, automated, repeatable, and manageable. The integration with MongoDB facilitates centralized management of datasets, while Dockerization ensures portability and readiness for production.

The FastAPI service provides real-time intrusion detection and attack classification through API calls. The usage of logging and monitoring solutions increases reliability and simplifies debugging procedures during deployment and runtime.
The findings of the experiments conducted prove the effectiveness of the suggested cybersecurity framework.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

Fast advancement in internet technology, cloud computing, IoT, and digital communications have greatly raised the number of cyber security attacks that have posed a great danger to computer networks in modern network environment. Traditional intrusion detection systems are no longer effective offering protection computer networks from advanced and everevolving cyber-attacks since they depend entirely.This is because of some problems such as inability to detect new attacks, low accuracy rates, lack of automation, inefficiency in scaling up, and inefficient deployment.

The presented project entitled **"Intelligent Network Intrusion Detection System Using Machine Learning and MLOps"** manages to overcome all these weaknesses in the implementation of the intrusion detection system through the integration of machine learning and MLOPS technologies. The main difference between the intrusion detection systems developed using traditional techniques and the one proposed here is that the latter not only detects cyber-attacks but also offers full life cycle automation for machine learning techniques.

The suggested architecture is based on a modular design of a machine learning operations (MLOps) pipeline, which was inspired by modern industrial machine learning solutions. The pipeline unifies various data pipelines, such as ingestion, validation, transformation, model trainer, prediction service, deployment, logging, and monitoring pipelines into one cybersecurity solution.

The project includes successful implementation of a number of supervised machine learning algorithms, namely decision tree, random forest, SVM, and K-nearest neighbors for the intrusion detection use case. Public datasets such as NSL-KDD and CICIDS2017, that consist of various classes of attacks including DoS attacks, brute force attacks, probe attacks, and unauthorized access, are utilized for machine learning models development.

The implementation of preprocessing and transformation pipelines enhances the quality of datasets via various steps, including data cleaning, normalization, encoding, and transformation. The transformed datasets are stored as artifacts for reutilization during future deployment and retraining phases.

One of the key strengths of the suggested project is that MongoDB, Docker, FastAPI, logging services, and CI/CD pipelines have been successfully integrated into the intrusion detection system architecture. MongoDB allows for efficient data handling and scalability of the dataset, whereas Docker helps deploy containers across different environments. FastAPI supports the provision of real-time predictions for incoming network traffic.

The developed MLOps pipeline increases the scalability, automation, reproducibility, maintainability, and deployment efficiency of the intrusion detection system. The logging services continuously monitor the performance of the pipeline, predict output, deployments, and operations carried out at runtime execution.

From the results of the experiments performed, machine learning algorithms proved to be superior when it comes to intrusion detection compared to traditional methods. When considering the performance of different algorithms, Random Forest shows the most effective results due to the properties of ensemble learning, robustness, scalability, and classification accuracy.

Furthermore, the implemented architecture is capable of real-time intrusion detection and automatic alerting that allows responding immediately to any threats and unusual activities that might be happening. Finally, ongoing monitoring and automatic re-training processes contribute to higher flexibility and adaptability concerning possible changes in traffic and emerging cyberattacks.

In general, the proposed solution shows that application of Machine Learning and MLOps methods can help to create an intrusion detection system that would provide intelligent, automated, and production-ready cybersecurity services for enterprises and their networks.

7.2 Achievements of the Project

The project outlined succeeds in accomplishing many critical goals in regard to intelligent intrusion detection, machine learning lifecycle automation, and production infrastructure. The most prominent success story regarding the proposed project is that the development of an automated system for detecting intrusive behaviors based on machine learning algorithms was successfully carried out.

Modular MLOps Pipelines Implemented in the Project Include:

- Data Ingestion Pipeline
- Data Validation Pipeline
- Data Transformation Pipeline
- Model Trainer Pipeline
- Prediction Pipeline
- Deployment Workflow

These pipelines automate various stages of the ML life cycle and minimize human intervention during execution of the workflow.

Another significant accomplishment includes the successful incorporation of MongoDB for storing and managing intrusion detection dataset. The use of MongoDB increases scalability, data access efficiency, and automation of the pipeline necessary for large cybersecurity systems.

Additionally, there was successful incorporation of Docker containerization for deploying the model, which helps achieve portability and reproducibility.

Real-time predictions are provided using FastAPI prediction services that can be deployed as an API-based system.
Logging and monitoring services have been developed to track operational processes, prediction results, deployment logs, and status of pipeline execution.

Also, the project supports:

- Automated data preprocessing
- Artifact generation
- Real-time intrusion detection
- CI/CD integration
- Deployment automation
- Continuous monitoring
- Model retraining workflows

Implementation of all the above-mentioned elements of MLOps proves the efficacy of the suggested model when creating intrusion detection systems that can be used in contemporary cybersecurity.

7.3 Applications of Proposed System

The suggested MLOps-Based Intelligent Network Intrusion Detection System has many applications in current digital structures where network security and the need for timely cyberattack detection are fundamental needs. It can be applied in corporate networks, banking systems, educational establishments, healthcare networks, cloud computing systems, IoT networks, government agencies, and defense systems for enhancing their security systems.

In corporate environments, the suggested system will be able to monitor the network traffic flow and detect any suspicious activities automatically. By doing so, it helps organizations automate the process of intrusion detection, thus making operations security management easier.

In cloud computing networks, the suggested framework can monitor the cloud-based network environment and detect any threats in the network traffic flow on time. This is necessary as cloud computing deals with large amounts of data and shares resources.

In banking and finance sectors, the proposed intrusion detection framework can be utilized for monitoring online transactions, preventing any access attempts, and detecting frauds through cyber-attacks targeting financial information.

For education institutes and universities, the proposed intrusion detection system will prove useful in securing their network environment, student database, online portals, and research systems from any potential attacks from malware or unauthorized access.

For health sector, the intrusion detection framework can be used for securing patient records, healthcare systems, communication systems, and even for securing medical devices that are connected in the healthcare sector. Security of these infrastructures is crucial since attacks on hospitals can potentially affect medical operations.

Moreover, the proposed intrusion detection approach is very useful in securing IoT devices where the connected devices keep transmitting information via communication networks. The intelligent intrusion detection techniques improve the security of such devices which lack security features by themselves.
For government institutes and infrastructures, the proposed system can be used to secure communication systems and other important networks.

The proposed framework additionally supports:

- Real-time monitoring
- Automated cyber defense
- Threat prevention
- Continuous monitoring
- Automated alert generation

Machine Learning and MLOps technologies have also been integrated into the intrusion detection process to make it more automated, scalable, efficient, and adaptable.

7.4 Limitations

While there have been significant advancements made in terms of improving detection capabilities as well as automating the process, some challenges persist within the current approach to intrusion detection framework.

A challenge with the project relates to the fact that the system relies on publicly available benchmark datasets like NSL-KDD and CICIDS2017 for developing models as well as carrying out evaluations. While they do provide an ideal evaluation environment, they may not fully represent real-time behaviors and evolving cyber-attacks.

Another challenge with the current system relates to the reliance on supervised learning algorithms that need labelled datasets for training purposes. The process of collecting and labelling large datasets related to network intrusion detection is time-consuming.

Finally, the current system mainly concentrates on specific types of cyber-attacks including DoS, Brute Force, Probe, and Unauthorized Access Attacks. However, more sophisticated cyber-attacks will need larger datasets as well as complex deep learning models.

While dockerization and the adoption of MLOPs help with scaling and automation, deploying the model within highly distributed cloud or corporate environments might pose new difficulties with regard to latency, infrastructure management, distributed logging, and distributed scaling.

The current solution does not include such sophisticated features as threat intelligence engines, self-learning capabilities, real-time data streaming and analysis, and cloud orchestration systems like Kubernetes.
All the same, the presented framework is solid ground for developing an intrusion detection system that incorporates MLOPs.

7.5 Future Enhancements

Moreover, there are a number of additional enhancements that can be added to this intrusion detection system for its better performance in terms of scalability, adaptability, deployment capacity, and cybersecurity awareness.

Firstly, advanced Deep Learning frameworks including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), LSTM, and Transformer can be integrated into this model to improve its ability to detect advanced and constantly evolving cyberattacks.

In addition, one can consider implementing real-time streaming analytics by using such frameworks as Apache Kafka, Apache Spark, and other cloud-native solutions, which will allow for performing the analysis of data in real time and will result in improved intrusion detection.

As for the deployment process of this cybersecurity system, future research can imply the deployment of the software infrastructure on cloud-based environments such as AWS, Microsoft Azure, or Google Cloud, among others.

It is also reasonable to implement Kubernetes for better deployment workflow management. The MLOps pipeline can also be improved further by integrating:

- Advanced monitoring dashboards
- Automated rollback mechanisms
- Anomaly detection services
- Security orchestration tools
- Model drift detection
- Automated retraining triggers

Future improvements may additionally focus on:

- IoT security monitoring
- Blockchain-based cybersecurity frameworks
- Explainable AI for intrusion detection
- Federated learning
- Hybrid anomaly and signature-based detection
- Edge computing-based intrusion detection
- Adaptive self-learning cybersecurity systems

Such developments will greatly enhance intrusion detection framework to its intelligence, scalability, flexibility, automation, and ability to be deployed.

REFERENCES

1. Scikit-learn Official Documentation: Used for machine learning algorithms such as Random Forest, Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).
2. MongoDB Official Documentation: Used for MongoDB integration, NoSQL database management, and scalable dataset storage.
3. FastAPI Official Documentation: Used for API development and real-time prediction service implementation.
4. Docker Official Documentation: Used for Docker containerization and deployment architecture.
5. Pandas Official Documentation: Used for data preprocessing, manipulation, and analysis.
6. NumPy Official Documentation: Used for numerical computation and array processing.
7. GitHub Documentation: Used for version control, source code management, and CI/CD workflow understanding.
8. MLflow Official Documentation: Used for model tracking, experiment management, and MLOps lifecycle management.
9. CICIDS2017 Dataset Documentation: Used as a benchmark dataset for intrusion detection model training and evaluation.
10. NSL-KDD Dataset Information: Used for intrusion detection experiments and machine learning evaluation.
11. Python Official Documentation: Used for implementation of machine learning pipelines, APIs, and automation workflows.
12. BackendArchitectX Network Security Repository: Primary reference repository used for understanding the MLOps-based intrusion detection system architecture and implementation workflow.
13. Stallings, William. *Network Security Essentials: Applications and Standards*. 6th Edition, Pearson Education, 2017.
14. Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006.
15. Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd Edition, O'Reilly Media, 2022.
16. Sarker, Iqbal H. – Machine Learning: Algorithms, Real-World Applications and Research Directions SN Computer Science, vol. 2, no. 3, 2021.
17. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
18. Turnbull, James. *The Docker Book: Containerization is the New Virtualization*. 2014.
19. Kim, Gene, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook*. IT Revolution Press, 2016.
20. Alla, Sridhar, and Suman Kalyan Adari. *Beginning MLOps with MLflow*. Apress, 2021.
21. Dasgupta, D., Akhtar, Z. and Sen, S., 2022. Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, *19*(1), pp.57-106.
22. Mishra, P., Varadharajan, V., Tupakula, U. and Pilli, E.S., 2018. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials*, *21*(1), pp.686-728.
23. Tsai, C.F., Hsu, Y.F., Lin, C.Y. and Lin, W.Y., 2009. Intrusion detection by machine learning: A review. *expert systems with applications*, *36*(10), pp.11994-12000.
24. Yin, C., Zhu, Y., Fei, J. and He, X., 2017. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, *5*, pp.21954-21961.
25. Liang, P., Song, B., Zhan, X., Chen, Z. and Yuan, J., 2024. Automating the training and

deployment of models in MLOps by integrating systems with machine learning. *arXiv preprint arXiv:2405.09819*.
26. Mahida, A., 2021. A review on continuous integration and continuous deployment (CI/CD) for machine learning. *International journal of science and research*, *10*(3), pp.1967-1970.
27. Shone, N., Ngoc, T.N., Phai, V.D. and Shi, Q., 2018. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, *2*(1), pp.41-50.
28. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A. and Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. *IEEE access*, *7*, pp.41525-41550.
29. Awajan, A., 2023. A novel deep learning-based intrusion detection system for IOT networks. *Computers*, *12*(2), p.34.
30. Aminu, M., Akinsanya, A., Dako, D.A. and Oyedokun, O., 2024. Enhancing cyber threat detection through real-time threat intelligence and adaptive defense mechanisms. *International Journal of Computer Applications Technology and Research*, *13*(8), pp.11-27.
31. Ofoegbu, K.D.O., Osundare, O.S., Ike, C.S., Fakeyede, O.G. and Ige, A.B., 2024. Real-Time Cybersecurity threat detection using machine learning and big data analytics: A comprehensive approach. *Computer Science & IT Research Journal*, *4*(3), pp.478-501.
32. Lobato, A.G.P., Lopez, M.A., Sanz, I.J., Cardenas, A.A., Duarte, O.C.M. and Pujolle, G., 2018, May. An adaptive real-time architecture for zero-day threat detection. In *2018 IEEE international conference on communications (ICC)* (pp. 1-6). IEEE.
33. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M. and Xie, F., 2018. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.*, *41*(4), pp.39-45.
34. Chen, A., Chow, A., Davidson, A., DCunha, A., Ghodsi, A., Hong, S.A., Konwinski, A., Mewald, C., Murching, S., Nykodym, T. and Ogilvie, P., 2020, June. Developments in mlflow: A system to accelerate the machine learning lifecycle. In *Proceedings of the fourth international workshop on data management for end-to-end machine learning* (pp. 1-4).
35. Yang, M.T., Kasturi, R. and Sivasubramaniam, A., 2003. A pipeline-based approach for scheduling video processing algorithms on now. *IEEE Transactions on Parallel and Distributed Systems*, *14*(2), pp.119-130.
36. Charabi, Y. and Farhani, S., 2023. FPGA Application: Realization of IIR filter based Architecture. *Journal of VLSI Circuits and Systems*, *5*(02), pp.29-35.
37. Abiteboul, S., 1997, January. Querying semi-structured data. In *International Conference on Database Theory* (pp. 1-18). Berlin, Heidelberg: Springer Berlin Heidelberg.
38. Deutsch, A., Fernandez, M. and Suciu, D., 1999, June. Storing semistructured data with STORED. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data* (pp. 431-442).
39. Shoens, K., Luniewski, A., Schwarz, P., Stamos, J. and Thomas, J., 1993, September. The Rufus system: Information organization for semi-structured data. In *VLDB* (Vol. 93, pp. 97-107).
40. Rusu, O., Halcu, I., Grigoriu, O., Neculoiu, G., Sandulescu, V., Marinescu, M. and Marinescu, V., 2013, January. Converting unstructured and semi-structured data into knowledge. In *2013 11th RoEduNet international conference* (pp. 1-4). IEEE.
41. Skoutas, D. and Simitsis, A., 2007. Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *3*(4), pp.1-24