Detection, Identification and Isolation of Cyber-Attacks using Enhanced Long Short-Term Memory in Single and Network of Quadcopters

Erfan Afshar

A Thesis in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

December 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Erfan Afshar

Entitled: **Detection, Identification and Isolation of Cyber-Attacks using Enhanced Long Short-Term Memory in Single and Network of Quadcopters**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Rastko Selmic, Chair

_____ Dr. Mehdi Hojjati, Examiner

_____ Dr. Rastko Selmic, Examiner

_____ Dr. Khashayar Khorasani, Thesis Supervisor

Approved by _____

Dr. Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Date: 2024_____

ABSTRACT

Detection, Identification and Isolation of Cyber-Attacks using Enhanced Long Short-Term Memory in Single and Network of Quadcopters

Erfan Afshar

The cybersecurity of cyber-physical systems (CPS), particularly quadcopters, is critical due to their reliance on communication networks, which makes them vulnerable to cyber-attacks. This thesis addresses the security of quadcopters by introducing a novel framework for the simultaneous detection, identification, and isolation of cyber-attacks using Long Short-Term Memory (LSTM) networks. Unlike previous research that primarily focuses on detection, this work advances the field by integrating attack type identification and target isolation, enhancing overall security capabilities.

A contribution of this thesis is the emphasis on sequence generation as a pre-processing step for time-series data in LSTM models. By optimizing sequence length, overlap, and labeling methods, the proposed approach ensures the effective capture of temporal dependencies, substantially improving model performance for attack detection, identification, and isolation.

The study introduces a novel multi-output (MO) model for single quadcopters, utilizing a shared LSTM backbone with three output heads. This framework is extended to a network of quadcopters through a Multi-Input, Multi-Output (MIMO) architecture, which incorporates a flexible number of input heads for each quadcopter, enhancing scalability. The model supports both centralized and decentralized topologies, accommodating networks of varying sizes, ranging from 2 to 5 quadcopters.

Simulation results for Denial of Service (DoS), False Data Injection (FDI), and Replay attacks demonstrate the robustness of the proposed framework. The single quadcopter model achieved over 95% accuracy in attack detection, along with high precision in identifying attack types and locations. In networked setups, the centralized MIMO model delivered superior performance, while the decentralized approach also yielded promising results. These findings highlight the adaptability and effectiveness of the proposed approaches, paving the way for broader CPS applications and further advancements in sequence generation techniques.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, professor Khorasani, for his invaluable guidance and unwavering support throughout this journey.

I am also thankful to my colleagues and friends in the research group, especially Ali, for their insights and assistance, which have been instrumental in shaping this work.

Finally, I extend my deepest appreciation to my family for their constant encouragement, love, and support, which have been a source of strength during this process.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CPS Cyber-Physical System

UAV Unmanned Aerial Vehicle

DoS Denial of Service

FDI False Data Injection

LSTM Long Shrot-Term Memory

RNN Recurrent Neural Network

CNN Convolutional Neural Network

MI Multi Input

MO Multi Output

MIMO Multi Input Multi Output

SG Smart Grid

Chapter 1

1. Introduction

The increasing integration of cyber-physical systems (CPS) into various sectors, including smart grids and autonomous vehicles, underscores the critical need for robust cybersecurity measures. As CPSs become more prevalent, their vulnerability to cyber-attacks escalates, necessitating comprehensive security frameworks to safeguard these systems. Unmanned aerial vehicles (UAVs), particularly quadcopters, exemplify this trend due to their diverse applications in surveillance, disaster response, and delivery services. The reliance on intricate communication networks and control systems renders quadcopters susceptible to cyber threats, which can result in severe operational failures if not adequately addressed [1; 2; 3].

The unique dynamics of quadcopters present significant challenges for cybersecurity. Unlike many CPSs that can be modeled using linear approximations, quadcopters require sophisticated nonlinear models to accurately reflect their complex movement and control mechanisms. This necessity is highlighted in the literature, where nonlinear dynamics are emphasized as critical for understanding quadcopter behavior under both normal and attack conditions [4; 5]. The development of a nonlinear mathematical model for quadcopters not only enhances the simulation of their behavior but also provides insights into potential vulnerabilities that could be exploited during cyber-attacks [5].

Addressing the nonlinear control challenges inherent in quadcopters is vital for maintaining stability, especially when faced with cyber threats. Implementing effective control systems that manage the quadcopter's six degrees of freedom while ensuring resilience against disruptions is a primary concern. The design of six Proportional-Integral-Derivative (PID) controllers tailored for altitude, position, yaw, and attitude control exemplifies an adaptive approach to maintaining stability during both normal operations and cyber-attacks. Although PID controllers are traditionally associated with linear systems, their adaptation for nonlinear dynamics is crucial for ensuring operational reliability [6].

In the realm of CPS cybersecurity, a multi-faceted approach that encompasses detection, identification, and isolation of cyber-attacks is essential. Detection involves recognizing when a system is under attack, while identification focuses on determining the nature of the attack, be it Denial of Service (DoS), False Data Injection (FDI), or Replay attacks. Isolation further refines this process by pinpointing the specific compromised component, such as a sensor or actuator. While much of the existing research has concentrated on detection alone, a comprehensive solution that addresses all three aspects is

imperative for effective defense against cyber threats [7; 8]. This holistic approach not only enhances the resilience of quadcopters but also contributes to the overall security of CPSs.

To achieve this, the study introduces a novel multi-output (MO) framework using Long Short-Term Memory (LSTM) networks, which can handle the temporal nature of time-series data generated by quadcopter sensors and actuators. The LSTM network serves as the backbone for simultaneously detecting, identifying, and isolating various types of cyber-attacks. This multi-output approach improves the system's responsiveness to attacks, allowing for quicker and more accurate mitigation strategies. The emphasis on LSTM-based models is due to their ability to retain long-term dependencies in time-series data, which is crucial for capturing patterns that indicate attacks.

Another key contribution of this thesis is the focus on sequence generation as a pre-processing step for LSTM-based models. In time-series data, how the data is split into sequences can have a significant impact on the model's performance. This study is the first to emphasize the importance of this step in the context of quadcopter cybersecurity. By optimizing sequence generation, the LSTM network is better able to learn from the data, resulting in more accurate detection, identification, and isolation of attacks. This improvement in the pre-processing phase helps the model outperform similar approaches in the literature, offering a more robust defense mechanism for UAVs.

In addition to handling individual quadcopters, this thesis extends the multi-output approach to a network of quadcopters, referred to as the Multi-Input, Multi-Output (MIMO) model. The MIMO framework introduces LSTM-based input heads for each quadcopter in the network and new output heads for detecting, identifying, and isolating attacks across multiple units. This approach is designed to be adaptable, allowing it to scale from single quadcopters to networks of varying sizes, whether operating in centralized or decentralized configurations. This flexibility enables the system to be deployed in a variety of real-world applications where multiple quadcopters are used in coordinated operations.

By using both a single quadcopter and a network of quadcopters, this research is the first to introduce a dataset that captures cyber-attacks in multi-UAV systems. The MIMO model's adaptability, combined with the robust LSTM architecture, allows for superior performance in detecting, identifying, and isolating cyber-attacks across both configurations. Simulations conducted in this study demonstrate the model's ability to manage attacks such as DoS, FDI, and Replay across various quadcopter maneuvers, offering a comprehensive cybersecurity solution that can be applied to future UAV operations.

1.1 Problem Statement

The primary goal of this work is to enhance the security of nonlinear cyber-physical systems under cyber-attacks, with a focus on quadcopters, a use case that has gained significant attention in recent years. Quadcopter systems are connected to a Ground Control Station (GCS) that supervises their operation and sends desired reference points. Cyber-attacks targeting the communication link between the command and control and the quadcopters compromise their security, potentially leading to severe consequences.

This study employs enhanced Long Short-Term Memory (LSTM) models for the simultaneous detection, identification, and isolation of cyber-attacks on both single and networked quadcopters. To effectively

train and test the proposed machine learning models, two datasets were created by applying three types of cyber-attacks: Denial of Service (DoS), False Data Injection (FDI), and Replay attacks. The first dataset involves a single quadcopter, while the second extends to a network of five quadcopters, addressing the unique challenges of multi-unit implementation. Each dataset includes 50 distinct movement scenarios, with sensor and actuator data for each packet labeled in three ways: first, with a detection label indicating whether the data is normal or under attack (0 for normal, 1 for attack); second, with an identification label specifying the type of attack, such as Denial of Service, False Data Injection, or Replay; and third, with an isolation label identifying the specific target of the attack, such as a sensor or actuator. These detailed labels facilitate comprehensive detection, identification, and isolation of cyber-attacks.

1.2 Literature Review

This section reviews literature related to the cybersecurity of quadcopters, divided into three parts: Cybersecurity of CPS, exploring foundational studies on security in cyber-physical systems; Cybersecurity of Quadcopters, covering research specific to quadcopter systems; and RNNs for Cybersecurity of Quadcopters, examining works that apply recurrent neural networks to address cybersecurity challenges in these systems.

1.2.1 Cyber-security of CPS

In this section, papers related to the cybersecurity of cyber-physical systems (CPS) are evaluated and explained. These studies encompass different systems, attack types, and methodologies, reflecting the diverse landscape of research in this critical field. Each paper contributes unique insights and approaches to enhancing the security and resilience of CPS against an array of cyber-attacks, highlighting the ongoing challenges and advancements in protecting interconnected infrastructures.

Haider *et al*. [9] developed a deep CNN ensemble framework to enhance the detection of Distributed Denial of Service (DDoS) attacks within Software Defined Networks (SDNs). Their approach leverages CNN ensemble models trained on the CICIDS2017 dataset, a Flow-based benchmark suited for SDN environments, achieving high accuracy (99.45%) with minimal computational complexity. The framework was benchmarked against existing ensemble and hybrid deep learning approaches, demonstrating improved performance in detecting diverse, flow-based DDoS attack patterns. The research highlights the importance of adaptive machine learning frameworks in SDNs, which separate control and forwarding layers to optimize network resource allocation. By focusing on Flow-based features, this approach efficiently handles SDN traffic patterns, proving both scalable and cost-effective. This study contributes a CNN-based approach for DDoS defense, which could be foundational for future network security in SDN environments.

Mousavi *et al*. [10] presented a distributed neural network-based method for detecting false data injection (FDI) cyber-attacks in discrete-time, nonlinear multi-agent systems, focusing on agent sensors, actuators, and inter-agent communication channels. They employ a radial basis function neural network (RBFNN) observer to generate a residual signal for attack detection and use Lyapunov stability analysis to ensure the system's stability, establishing conditions for uniform ultimate boundedness (UUB) of residuals and formation error. This work distinguishes itself by addressing simultaneous attacks across multiple communication channels within a formation control context, where agents only share data with

neighbors. Through simulations, they demonstrate the efficacy of the proposed observer-based approach in maintaining stable formations, even under unknown nonlinearities in system dynamics. The method adds a robust layer of security for multi-agent systems by detecting and mitigating potential disruptions in real-time.

Bitirgen and Filik [11] proposed a hybrid deep learning model combining particle swarm optimization (PSO) with CNN-LSTM networks to detect false data injection attacks (FDIAs) and distinguish between physical disturbances in smart grids (SGs). Using phasor measurement unit (PMU) data, the model optimizes hyperparameters with PSO for high detection accuracy and speed, improving upon standalone LSTM and CNN-LSTM models. Testing on a diverse dataset, the hybrid approach achieved robust performance in binary, three-class, and multiclass classifications, showing promise as a scalable SG security mechanism. Their research emphasizes the significance of machine learning in cybersecurity for SGs, particularly for FDIA detection across interconnected communication and power networks. By leveraging CNN-LSTM architectures and PSO optimization, the model enhances reliability in SGs, demonstrating a powerful tool for real-time security in cyber-physical systems with multiple disturbance types.

Bharathi and Kumar [12] proposed an ensemble classifier-based framework for real-time cyber-attack detection in healthcare cyber-physical systems (HCPS), focusing on securing patient data in IoT-connected healthcare devices. Using a wise greedy routing technique for sensor node configuration, the model applies agglomerative clustering for data grouping and multi-heuristic cyber ant optimization to identify anomalies. Finally, an ensemble crossover XGBoost classifier detects attacks with high accuracy, improving true positives and reducing false positives. Simulations validate the model's effectiveness, highlighting its potential to enhance cloud-based healthcare security and real-time patient monitoring. This approach addresses the unique security challenges in HCPS by integrating cloud architecture for decentralized security and employing machine learning for rapid data analysis, ensuring minimal delays and low communication costs. The framework demonstrates the viability of ensemble learning in safeguarding sensitive medical information and supporting uninterrupted, secure healthcare services.

Yin *et al*. [13] proposed a privacy-preserving, subgrid-oriented framework using a spatial-temporal neural network for false data injection attack (FDIA) detection in smart grids. Unlike previous methods that focus on temporal data alone, their approach models both spatial and temporal relationships between measurement points across bus and line data, as well as interactions between subgrids. Using a microservice-based architecture, the framework enhances data privacy, supports parallel processing, and ensures low latency. The spatial-temporal network architecture captures interdependencies within the smart grid, facilitating robust FDIA detection. By employing fully connected layers for spatial relationships and long short-term memory layers for temporal aspects, this model achieves high detection accuracy in ac-model power systems, as demonstrated with the SimBench benchmark dataset. This method represents an improvement over existing centralized detection frameworks in terms of privacy, performance, and adaptability.

Li *et al.* [14] introduced an adaptive hierarchical framework for cyber attack detection and localization in active distribution systems with distributed energy resources (DER). Their approach uses electrical waveform analysis combined with a sequential deep learning model to detect attacks, even minor ones. A two-stage localization method first identifies the general sub-region of an attack using modified spectral clustering, then pinpoints the attack location with a normalized impact score based on

waveform properties. This framework provides high-fidelity detection and localization, accommodating the complex topology of DER networks. Extensive testing across multiple scenarios demonstrates improved detection accuracy compared to traditional methods, showcasing its adaptability and precision in real-world applications.

Sun *et al.* [15] proposed the CNN-LSTM with Attention Mechanism (CLAM) model for enhancing anomaly detection in in-vehicle networks, addressing the security gaps in the CAN bus protocol. The model leverages convolutional layers to extract key features, bidirectional LSTM for temporal dependencies, and an attention mechanism to focus on significant time steps, making it adaptable across different vehicle models without requiring CAN message parsing. By processing time-series data from CAN frames and reducing redundancy, CLAM achieves efficient, high-performance detection with low error rates. Extensive testing reveals that the model's design offers rapid response times and improved accuracy over comparable approaches, demonstrating its capability for real-time in-vehicle security applications.

Goh *et al.* [16] presented an unsupervised anomaly detection model for cyber-physical systems (CPS), utilizing a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to predict time-series data patterns within a water treatment system. By modeling the typical behavior of CPS sensor data, the LSTM-RNN captures temporal sequences, enabling it to distinguish anomalies, with anomaly detection enhanced by the Cumulative Sum (CUSUM) technique. This combined approach allows the model not only to detect attacks but also to identify the specific sensor compromised, achieving low false positive rates. The proposed method was validated on the Secure Water Treatment (SWaT) testbed, replicating real-world CPS challenges, particularly in critical water infrastructure. By employing LSTM-RNN and CUSUM for anomaly detection, this model addresses CPS security effectively without requiring labeled attack data, making it suitable for real-time monitoring where abnormal events are rare. The approach's precision in identifying specific compromised sensors further strengthens its utility in critical infrastructure security.

Baul *et al.* [17] proposed a hybrid model, XTM, that combines transformer and LSTM architectures to detect and locate False Data Injection (FDI) attacks in smart grids. The model leverages real-time sensor data analysis to identify FDI attacks by detecting anomalies in data patterns without relying on system parameters. It introduces a new threshold-based method for attack detection and a multilabel classifier to pinpoint attack locations. Baul's unique approach allows it to manage both hourly and minutely data, achieving high detection accuracy on the IEEE-14 bus system. Comparative analysis shows that XTM outperforms other deep learning models, showcasing its adaptability and effectiveness in real-world smart grid scenarios.

Namavar Jahromi *et al.* [18] proposed a two-stage ensemble deep learning framework for detecting and attributing cyber-attacks in IoT-enabled cyber-physical systems (CPS), with a focus on industrial control systems (ICS). The first stage uses a decision tree combined with deep representation learning to detect attacks, even within imbalanced data environments. In the second stage, deep neural networks classify attack types, enhancing attribution accuracy. This model's ability to detect previously unseen attacks makes it robust in real-world applications, as shown in tests on gas pipeline and water treatment datasets. Their approach leverages automated feature learning, addressing the data imbalance common in ICS environments. By focusing on both detection and attribution, the model enhances cybersecurity resilience in critical infrastructure, with computational efficiency similar to other DNN-based solutions but improved performance.

Lu *et al.* [19] introduced a representation-learning-based CNN (RL-CNN) model for detecting and localizing multiple cyber-attacks in cyber-physical power systems (CPPS). The RL-CNN treats the localization of diverse cyber-attacks, such as false data injection (FDI), denial of service (DoS), jamming, and hybrid attacks, as a multilabel classification problem, using representation learning to improve attack detection. Once attack locations are identified, a minimum mean-squared-error estimator (MMSE) supports system recovery by filtering compromised measurements and providing accurate state estimations for real-time CPPS stability. This approach addresses both detection and recovery in CPPS, providing a comprehensive solution to maintain system integrity amid complex attack scenarios. Extensive simulations demonstrate RL-CNN's efficiency in accurate attack localization and effective system recovery, outperforming traditional multilabel classifiers in CPPS environments.

Ravi *et al.* [20] proposed a recurrent deep learning-based ensemble meta-classifier to enhance network intrusion detection in cyber-physical systems (CPS). The approach employs recurrent neural networks (RNN, LSTM, GRU) to extract hidden-layer features, then applies kernel principal component analysis (KPCA) to select the most relevant features. These optimal features are fused to build an ensemble meta-classifier, achieving improved detection accuracy on various network intrusion datasets. Their model demonstrates effective detection and classification of network attacks, with performance exceeding traditional machine learning and deep learning methods. Visualization using t-SNE further confirms the relevance of learned feature representations, offering a robust, data-driven approach to CPS security.

Sakhnini *et al*. [21] introduced an ensemble deep learning approach for detecting and localizing physical-layer cyber-attacks in smart grids. The model addresses vulnerabilities arising from the integration of IoT and communication networks, with specific attention to attacks on the physical layer, an area that has received limited focus. The method utilizes representation learning to enhance attack pattern recognition, leveraging multiple machine learning classifiers and the chi-square algorithm to pinpoint attack sources and correlate them with specific system features. The ensemble model demonstrated high accuracy in both attack detection and localization when evaluated on a smart grid dataset. By incorporating chi-square-based localization, the model provides insights into attack-specific measurements, a critical capability for cyber-physical grid security. This research marks a step in advancing smart grid cybersecurity by combining multi-attack classification with precise localization of attacks within the operational layer, filling a gap in existing security solutions.

Shen *et al*. [22] presented a method for localizing False Data Injection Attacks (FDIAs) in smart grids using a CNN optimized by a Sparrow Search Algorithm (SSA). Recognizing that many current FDIA solutions struggle with localization precision and computational complexity, this model leverages measurement vectors to pinpoint the specific attacked buses or lines within smart grids, thereby enhancing localization accuracy and reducing false alarms. The proposed SSA-CNN approach efficiently adjusts CNN hyperparameters, achieving optimal localization performance across both IEEE 14-bus and IEEE 118-bus systems. The authors highlighted their method's comparative success in localization accuracy by testing it on simulated FDIA datasets and contrasting it with other advanced localization techniques. This SSA-CNN approach introduces a flexible, data-driven solution that addresses both invisible FDIAs and traditional localization limitations, underscoring its applicability for real-world smart grid security. This research notably advances the accuracy and applicability of FDIA localization in larger, complex power systems.

Panigrahi *et al*. [23] proposed a hybrid intrusion detection model combining Decision Table and Naive Bayes (DTNB) to enhance security in cyber–physical systems (CPS). Their approach leverages Multi-Objective Evolutionary Feature Selection (MOEFS) to isolate five critical features from the CICIDS2017 dataset, targeting efficiency in detecting various network attacks, including DoS, DDoS, and Brute Force. The model addresses issues in traditional intrusion detection, such as feature overload and class imbalance, achieving high accuracy by using only essential data features to streamline detection and reduce false alarms. Through their hybrid model, the authors demonstrate enhanced detection of complex attacks like GoldenEye, Heartbleed, and SQL Injection. By effectively balancing performance for binary and multiclass scenarios, this approach addresses both high-class imbalance and signature-based challenges, providing a robust IDS that achieves a detection accuracy of 96.8% on the CICIDS2017 dataset. The study emphasizes the model's capability for real-time network defense in complex CPS environments.

1.2.2 Cyber-security of Quadcopters

In this section, papers focused on the cybersecurity of quadcopters are evaluated and explained. This focus on quadcopter-related research serves as the primary use case for this work, examining different attack scenarios, methodologies, and security challenges specific to these aerial systems. Each study provides valuable insights into enhancing the security and resilience of quadcopters against a range of cyber threats, underscoring the importance of effective defense mechanisms in safeguarding these increasingly prevalent technologies.

Heidari *et al*. [24] proposed a secure intrusion detection platform tailored for the Internet of Drones (IoD), leveraging blockchain technology and Radial Basis Function Neural Networks (RBFNNs) to enhance data integrity and network resilience. The platform's decentralized structure, powered by blockchain, enables trust, transparency, and security across drone interactions, which is essential for public acceptance and operational safety in IoD applications. By integrating RBFNN, the authors address the challenges of detection accuracy and efficiency within the IoD's dynamic environment, where traditional intrusion detection methods often fall short. They also introduce a novel IoD architecture that facilitates secure data transfer through both drone-to-drone (D2D) and drone-to-X (D2X) communication, with a zero-knowledge proof protocol for enhanced registration and verification. Transfer learning is employed to reduce model convergence time, a critical feature for real-time IoD scenarios. Through experimental validation, the study demonstrates that the blockchain-based model surpasses existing methodologies in specificity, recall, precision, F1-score, and accuracy, offering a robust cybersecurity framework for IoD systems amid rising cyber threats.

Xiao and Feroskhan [25] presented a cyber-attack detection and isolation approach for quadrotor UAVs, focusing on vulnerabilities in unmanned aerial vehicles' cyber-physical systems (CPS). They propose the Modified Sliding Innovation Sequences (MSIS) detector, a novel state-estimation-based method that uses extended Kalman filtering to detect and isolate cyber-attacks in real time. This detector is optimized to handle random attacks, False Data Injection (FDI) attacks, and Denial-of-Service (DoS) attacks on both sensors and actuators. MSIS utilizes the operator norm of normalized innovation sequences within a sliding window, triggering alerts when values exceed a predefined threshold, which helps maintain accuracy even during rapid UAV maneuvers. The MSIS detector introduces an iterative calibration process to mitigate false alarms, especially during complex trajectories involving rapid rotation. Unlike

prior models limited to simplified linear scenarios, Xiao and Feroskhan's model addresses a full-state, nonlinear quadrotor, accounting for translational and rotational dynamics. By isolating attack types through statistical analysis of mean and covariance, their approach enables more accurate detection in high-speed, dynamic scenarios. Simulation results affirm that the MSIS detector outperforms previous state estimation-based detectors, providing a comprehensive solution tailored to the unique security demands of UAVs in real-time operations.

Alferaidi *et al*. [26] introduced a distributed intrusion detection model tailored for IoT-based vehicles, which combines deep convolutional neural networks (CNN) with long short-term memory (LSTM) networks within the Apache Spark framework. This model addresses the need for high-speed, high-accuracy detection on the Internet of Vehicles, where traditional methods struggle due to the complex data flow and varying intrusion patterns. The CNN-LSTM structure is designed to process large-scale network traffic efficiently, distinguishing between normal and abnormal behavior with accuracy and speed. Using Spark, the system enables real-time parallel processing of data, which is critical in the dynamic and large-scale environments typical of IoT-based vehicle networks. Through experimentation on benchmark datasets like NSL-KDD and UNSW-NB15, Alferaidi et al. demonstrate that their distributed model achieves high detection accuracy while reducing detection time. This setup is particularly effective in identifying complex, multidimensional cyber threats specific to IoT vehicle networks. By integrating distributed processing with deep learning, the proposed approach addresses the limitations of existing machine learning techniques in handling extensive datasets, offering a scalable, robust solution for securing the Internet of Vehicles against evolving cyber threats.

Eshmawi *et al*. [27] proposed a machine learning ensemble approach to secure small UAVs from GPS spoofing attacks, focusing on the vulnerabilities inherent in GPS-dependent navigation. Their model utilizes a stacked ensemble technique, combining traditional machine learning with deep learning models to enhance detection accuracy without the need for additional hardware, a practical solution given the limited capacity of small UAVs. The model relies on a dataset of 13 GPS signal characteristics, preprocessed with z-score normalization, and validated through controlled simulations, achieving an impressive detection accuracy. This ensemble-based framework addresses the need for robust GPS spoofing detection in scenarios where UAVs play critical roles, such as security and logistics, without requiring complex or hardware-intensive solutions. To tackle the challenges posed by sophisticated spoofing techniques, the proposed model applies ensemble learning for superior performance over conventional machine learning approaches, which often falter under complex spoofing conditions. The study's contributions include an in-depth methodology for dataset acquisition, data preparation, and comparative model evaluation, which highlights the advantages of ensemble classifiers in recognizing and categorizing GPS spoofing incidents. Their approach presents a scalable, efficient intrusion detection framework for small UAVs, capable of enhancing operational reliability in crucial sectors like delivery and surveillance, where GPS spoofing can pose significant security risks.

Ramadan *et al*. [28] proposed a deep learning-based intrusion detection framework specifically for the Internet of Drones (IoD), leveraging Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) architectures to address the rising security concerns in Flying Ad Hoc Networks (FANET). The framework consists of two main RNN modules: a distributed module installed on each drone to locally monitor communication traffic and a centralized module at the base station for comprehensive attack verification and response coordination. This dual-module setup allows for efficient detection of suspicious activities within individual drones and across the entire IoD network, enabling real-time

response to various cyber threats. The architecture's use of big data analytics aids in anomaly detection by processing large volumes of real-time data for accurate threat identification. The study demonstrates the proposed framework's robustness and efficiency through extensive experiments using multiple datasets, showing that the LSTM-RNN configuration offers superior intrusion detection performance over traditional methods. The real-time, distributed setup provides a scalable solution for FANETs, where drones operate in complex, dynamic environments vulnerable to sophisticated intrusion attempts. Their approach addresses the need for effective anomaly-based intrusion detection within IoD, showing improvements in detection accuracy and response time, thereby enhancing network resilience in critical applications such as surveillance, delivery, and communication.

Li *et al*. [29] explored jamming detection and classification in OFDM-based UAVs using a machine learning (ML) approach that combines feature-based and spectrogram-based models. Focusing on four types of jamming attacks, barrage, protocol-aware, single-tone, and successive-pulse. the authors utilize software-defined radio (SDR) to simulate attacks and collect data from UAVs. Their feature-based model uses conventional ML techniques, drawing on parameters like signal-to-noise ratio and OFDM characteristics, while the spectrogram-based model employs convolutional neural networks (CNNs) to analyze jamming spectrums through spectrogram images. Experimental results show that the spectrogram-based CNN model outperforms the feature-based model, achieving a classification accuracy of 99.79% with a false alarm rate of just 0.03%, compared to 92.2% accuracy and 1.35% false alarm for the feature-based model. This dual-model framework offers a practical, high-performance solution for jamming detection in UAVs without requiring hardware modifications, making it well-suited for real-world implementation. By providing an additional dataset of spectrogram images for ML model training, they enhance the resources available for UAV cybersecurity research. Their approach allows for the reliable classification of jamming types, essential for developing adaptive countermeasures to protect UAVs from interference-based threats. This work highlights the advantages of spectrogram-tailored deep learning in UAV cybersecurity, particularly in detecting subtle attack patterns that feature-based methods may miss, thereby improving UAV operational security in sensitive applications.

Gasimova *et al*. [30] conducted a comparative study on ensemble learning techniques for detecting GPS spoofing attacks on UAVs, focusing on three models: bagging, stacking, and boosting. With GPS spoofing posing a significant threat to UAV navigation and safety, particularly due to unencrypted civilian GPS signals, the authors aim to enhance detection accuracy and reduce misdetection rates through ensemble models. Each model combines the predictions of multiple classifiers to improve performance, leveraging hyperparameter tuning with Grid search and Pearson’s Correlation Coefficient for optimal feature selection. The study evaluates the models across seven metrics, including accuracy, detection and false alarm probabilities, memory size, and processing and prediction times, finding that the stacking model outperforms bagging and boosting across all metrics. The study highlights the limitations of existing GPS spoofing detection techniques, many of which rely on hardware solutions or traditional machine learning methods without hyperparameter optimization. By utilizing ensemble models, Gasimova et al. provide a robust detection mechanism that offers high accuracy and computational efficiency, making it suitable for UAVs with limited processing capabilities. This research contributes to UAV cybersecurity by providing an efficient and scalable machine learning framework for GPS spoofing detection, emphasizing ensemble learning’s ability to enhance detection reliability in UAV navigation systems.

Basan *et al*. [31] introduced a data normalization technique for detecting cyber-attacks on UAVs by analyzing the unique cyber-physical parameters of these systems. UAVs, as cyber-physical systems (CPS), face threats that impact their functionality rather than just information integrity. To enhance UAV cybersecurity, Basan et al. propose a normalization framework that processes and standardizes various UAV data inputs, allowing for more effective detection of anomalies and attacks. This normalization approach supports the integration of different types of data into a consistent format, which aids in identifying abnormal changes in parameters, potentially signaling attacks. The model also formalizes UAV subsystems and parameter relationships, laying the groundwork for developing a CPS-based digital twin for UAV cybersecurity research. The study further contributes by presenting an experimental analysis of the effects of specific attack types, such as integrity, availability, confidentiality, resource exhaustion, and access attacks, on UAV parameters. By developing a software module to collect and normalize data, the authors create a system that supports intrusion detection and improves the consistency of UAV attack datasets. This standardized format facilitates future machine learning applications for UAV cybersecurity, enabling the classification of attack patterns and enhancing real-time detection capabilities in UAV systems.

Almotery [32] explored the potential of blockchain technology as a solution for enhancing cybersecurity in drone systems, particularly in the context of Industry 4.0, where drones are increasingly integrated into various sectors such as healthcare, military, and e-commerce. With the rise of the Internet of Drones, the demand for secure communication and data exchange between UAVs and other devices has intensified. Blockchain, with its decentralized structure and tamper-resistant features, is posited as an effective method for improving data integrity, authentication, and communication security in drone networks. Almotery emphasizes that blockchain can support secure coordination among drones by ensuring traceability, transparency, and autonomy, which are essential as UAV usage expands beyond centralized control. In addition to outlining blockchain's potential benefits, the study surveys perspectives from industry experts on integrating blockchain into UAV systems, providing qualitative insights into its viability. The findings suggest that blockchain could address various drone security challenges, such as data tampering, unauthorized access, and system malfunctions, particularly in sensitive applications like medical deliveries and battlefield operations. By demonstrating blockchain's applicability to UAV security, Almotery's research highlights a scalable approach to building trust within IoT ecosystems, encouraging further exploration and experimentation with blockchain as a cybersecurity solution in the UAV industry.

Talaei Khoei *et al*. [33] presented two dynamic selection techniques: Metric Optimized Dynamic (MOD) and Weighted Metric Optimized Dynamic (WMOD), for detecting GPS spoofing attacks on UAVs. GPS spoofing, which involves transmitting counterfeit signals to mislead UAVs, presents significant risks in both military and civilian UAV applications. The authors propose MOD and WMOD as adaptive classifier selection methods that dynamically identify the optimal classifier for detecting spoofing attacks in real time. Both methods employ ten machine learning models, including Support Vector Machine, Decision Tree, and Random Forest, to identify attack patterns based on a dataset comprising 13 GPS signal features derived from real-time experiments and MATLAB simulations. MOD and WMOD outperform traditional ensemble methods, achieving detection accuracy of 99.6%, with low probabilities of misdetection (1.09%) and false alarm (1.56%). The study highlights the limitations of existing GPS spoofing detection methods, such as cryptographic, signal processing, and external UAV characteristic-based techniques, which often require additional hardware or extensive computation. The dynamic

selection methods MOD and WMOD, by contrast, adaptively optimize model selection, reducing processing time and improving detection precision. The authors' work advances UAV cybersecurity by providing a flexible, high-performance approach for real-time GPS spoofing detection that is suitable for resource-constrained UAVs, offering a robust alternative to traditional static and hardware-dependent methods.

Aissou *et al*. [34] evaluated five instance-based machine learning models for detecting GPS spoofing attacks on Unmanned Aerial Systems (UAS), specifically focusing on K-Nearest Neighbor (KNN), Radius Neighbor, Linear SVM, C-SVM, and Nu-SVM. The study addresses the vulnerabilities of unencrypted GPS signals, which are susceptible to spoofing attacks that pose severe risks, such as hijacking or redirecting UAS. Using software-defined radio units, the authors collected GPS data and simulated three spoofing attack types: simplistic, intermediate, and sophisticated. Their results demonstrate that Nu-SVM achieved the highest performance in accuracy, detection probability, and computational efficiency, making it particularly suitable for real-time GPS spoofing detection in UAS. Hyperparameter tuning was applied to ensure optimal model performance across key metrics, including probability of false alarm, misdetection rate, processing time, and memory usage. The study highlights the limitations of existing GPS spoofing detection techniques, such as reliance on additional hardware, which may be impractical for lightweight UAS. The instance-based models evaluated offer a software-based alternative that aligns well with the Size, Weight, and Power constraints of UAS. By identifying optimal features for spoofing detection and leveraging instance-based learning, Aissou et al. provide an efficient approach for robust GPS spoofing mitigation, contributing valuable insights to the development of lightweight, real-time cyber defense mechanisms for UAS in both civilian and military applications.

Bouhamed *et al*. [35] proposed a lightweight intrusion detection and prevention system (IDPS) for UAV networks based on Deep Reinforcement Learning (DRL) with a focus on energy-efficient and periodic offline learning to enhance security. Utilizing Deep Q-Learning (DQN), the IDPS module is designed to autonomously detect and respond to cyber threats, while a custom reward function addresses the class imbalance in intrusion data, ensuring minor classes are effectively identified. Given the resource constraints of UAVs, the framework leverages a periodic offline learning approach to update the IDPS model, enabling UAVs to adapt to new attack patterns without relying on continuous real-time learning, which would otherwise be energy intensive. In contrast to online learning, which updates models based on individual UAV data, the offline learning approach aggregates recent data from multiple UAVs, training a robust global model at a central station. This global model is then deployed to individual UAVs during docking, allowing consistent and up-to-date threat detection across the network. The study's results indicate that this periodic DRL approach outperforms classical machine learning and online learning methods, providing higher accuracy in intrusion detection and lower energy consumption. By enabling autonomous and adaptive cybersecurity for UAVs, the proposed framework strengthens network security with minimal resource expenditure, effectively safeguarding UAV operations across various environments.

Whelan *et al*. [36] explored a novel approach to intrusion detection for Unmanned Aerial Vehicles (UAVs) using one-class novelty detection methods. This strategy leverages UAV flight logs, which typically lack attack labels, allowing the system to learn normal sensor behavior without relying on labeled attack data. They employ three one-class classifiers: One-Class Support Vector Machine (SVM), Autoencoder Neural Network, and Local Outlier Factor to monitor deviations in sensor data that may indicate potential security threats, such as GPS spoofing. To manage high-dimensional data from UAV sensors,

Principal Component Analysis (PCA) is applied for dimensionality reduction before classifier training, optimizing the system's performance. This study showcases high F1 scores, with results reaching up to 99.73% for detecting malicious sensor readings across various UAV platforms, highlighting the adaptability and effectiveness of this approach. By focusing on novelty rather than anomaly detection, the model sidesteps the need for labeled intrusion data, making it adaptable across multiple UAV configurations and sensor types. This methodology provides a promising direction for IDS in UAV applications, especially in unpredictable, high-stakes environments like military or industrial control settings, where rapid adaptation to novel threats is essential.

Baig *et al*. [37] proposed a machine learning-based approach to detect cyber threats against drones operating within smart city environments. With the increasing adoption of drones for services like surveillance and traffic monitoring, the vulnerability of these unmanned aerial vehicles (UAVs) to cyber attacks has become a pressing concern. The authors focus on detecting three specific types of attacks: hijacking, GPS signal jamming, and denial of service (DoS) attacks. The study utilizes a dataset derived from the DJI Phantom 4 drone, encompassing both normal flight behaviors and malicious attack signatures. The authors employ various machine learning algorithms to classify the synthesized data, demonstrating effective detection capabilities. Their results indicate high classification performance, achieving F1 scores of up to 99.56% for benign sensor readings and 99.73% for malicious readings. This research emphasizes the critical need for robust cybersecurity measures in UAV operations, especially as smart cities increasingly integrate drone technology. By leveraging machine learning techniques, the proposed method aims to enhance the safety and reliability of drone services, ultimately contributing to the overall resilience of smart city infrastructures against cyber threats. The paper is structured to cover the problem background, dataset acquisition, the proposed detection framework, and a detailed analysis of simulation results, concluding with insights into future research directions.

Aldaej *et al*. [38] discussed the vulnerabilities and privacy concerns associated with the integration of the Internet of Things (IoT) into drone technology, particularly focusing on small drones. They emphasize the need for a secure network of drones (NoD) to mitigate risks such as interception and intrusion. The study proposes a hybrid machine learning approach, combining logistic regression and random forest techniques, to classify data instances effectively and enhance cybersecurity within drone networks. The proposed framework aims to address the security challenges faced by drone systems, leveraging advanced machine learning models to ensure reliable operation in various applications, from industrial surveillance to disaster response. The authors present a comprehensive modular framework that includes essential components such as a drone module, edge computational module, and security module, each playing a crucial role in maintaining data integrity and privacy. Their approach utilizes IoT sensors and drone data to manage security attacks effectively. The framework is evaluated using two challenging datasets, demonstrating impressive performance metrics, including an accuracy of 98.58% and a precision of 97.68%. The study highlights the potential for implementing machine learning techniques in securing drone operations and addresses the importance of adapting to evolving security threats in the rapidly advancing field of drone technology

1.2.3 RNNs for Cyber-security of Quadcopters

This section explores the application of recurrent neural networks (RNNs) which is the focus of this work in enhancing the cybersecurity of quadcopters. The focus is on studies that leverage RNN architectures

to address various cyber threats faced by these aerial systems. Each paper examines unique methodologies and approaches, highlighting how RNNs can improve detection, isolation, and response to cyber-attacks. The insights gained from this body of research underscore the potential of RNNs as effective tools for safeguarding quadcopters in an increasingly complex cyber landscape.

Hassler et al. [39] proposed a novel intrusion detection system (IDS) specifically designed for unmanned aerial vehicles (UAVs), addressing the critical issue of cybersecurity in these increasingly utilized systems. The authors highlight the limitations of current IDSs, which typically focus on either cyber or physical features but fail to integrate both dimensions, leading to suboptimal detection capabilities. To tackle this, they developed a comprehensive testbed that includes a UAV, controller, and data collection tools, enabling the execution of various cyber-attacks such as denial-of-service and false data injection. Additionally, they created and publicly shared a dataset capturing both cyber and physical data under normal and attack conditions, facilitating further research in the field. The study examines the impact of fusing cyber and physical features on the effectiveness of IDSs trained using different machine learning techniques, including support vector machines and convolutional neural networks. The authors conduct extensive experiments to determine whether this fusion enhances detection performance, particularly when the models are trained on a single attack type and tested on unseen attacks of varying complexities. The findings aim to provide insights into the capabilities of IDSs that incorporate a holistic view of UAV systems, emphasizing the need for methodologies that balance model complexity with detection performance.

Viana et al. [40] addressed the critical challenge of attack identification in 5G unmanned aerial vehicle (UAV) communications, proposing a novel deep learning architecture called Deep Attention Recognition (DAtR). Recognizing the inherent vulnerabilities in UAV operations despite robust 5G security features, the authors focus on detecting jamming attacks using two key parameters: Signal to Interference plus Noise Ratio (SINR) and Received Signal Strength Indicator (RSSI). Their approach involves a deep network that can effectively identify attacks under various conditions, including Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS). By leveraging these parameters, which reflect channel variations and wireless conditions, the DAtR model aims to provide reliable attack detection even in complex urban environments where additional terrestrial users may interfere with communication. The study contributes to the field by introducing innovative methods such as Time Series Augmentation (TSA) and Majority Voting Algorithm (MVA) to enhance classification accuracy and reduce false alarms. The authors also explore the integration of Long Short-Term Memory (LSTM) and Attention layers within the deep network architecture, demonstrating its effectiveness in recognizing jamming attacks compared to traditional machine learning classifiers. Through extensive evaluation, the DAtR architecture achieves superior accuracy, even outperforming well-known classifiers, while maintaining efficiency in resource-constrained UAV environments. The findings emphasize the potential of deep learning techniques to bolster UAV security in 5G networks, paving the way for safer UAV operations amid evolving cyber threats.

Miao et al. [41] proposed a deep-meta-heuristic system for intrusion detection in unmanned aerial vehicles (UAVs), addressing the increasing vulnerability of UAV communications to various cyber threats. Recognizing the critical role of UAVs in applications such as emergency management and wildlife conservation, the authors highlight the significance of an effective Intrusion Detection System (IDS) that monitors and identifies suspicious activities within UAV communication networks. Their approach combines a Greedy-based Genetic (GG) algorithm for optimal feature selection with a modified Deep

Convolutional Neural Network and Bi-Long Short-Term Memory (CNN-BiLSTM) model enhanced by an attention mechanism. This architecture aims to improve classification accuracy by emphasizing relevant data while suppressing noise, ultimately addressing limitations in traditional IDS approaches that often struggle with accuracy, computational speed, and noise handling. The study's contributions are multifaceted, including the development of an effective feature selection process using the GG algorithm to enhance the classification capabilities of the IDS. The proposed modified CNN-BiLSTM architecture is designed to efficiently detect intrusions, leveraging the strengths of both deep learning and machine learning methodologies. The authors evaluate their system using various performance metrics, including accuracy, sensitivity, and precision, demonstrating its effectiveness in comparison to conventional intrusion detection methods. By establishing a comprehensive framework for UAV intrusion detection, this work underscores the need for advanced algorithms that can adapt to the complex and dynamic nature of UAV communications while ensuring robust security against potential attacks.

Tlili et al. [42] introduced a hybrid adaptive framework for detecting faults and attacks in unmanned aerial vehicles (UAVs), addressing the growing need for robust security measures as UAV applications expand across various sectors. The authors note that while numerous artificial intelligence techniques have been applied to enhance UAV security, there has been a lack of comprehensive studies focusing on hybrid frameworks that integrate both fault and attack detection. Their proposed framework operates on centralized and decentralized architectures, utilizing two distinct input flows to learn high-level features from data related to faults and attacks. The empirical results demonstrate that their framework achieves over 85% accuracy for fault detection and an impressive 96.7% accuracy for attack detection. The study emphasizes the importance of identifying abnormal behaviors in UAV operations, particularly in environments where multiple UAVs operate collaboratively. By leveraging deep learning architectures, the proposed hybrid adaptive framework aims to enhance UAV resilience against cybersecurity threats and operational failures. The authors advocate for a comprehensive approach that encompasses both types of detection to improve overall UAV security, making a contribution to the existing literature on UAV cybersecurity. Through their framework, they aim to provide solutions that ensure the integrity of UAV operations, thus supporting the safe and efficient deployment of UAVs in complex environments.

Hickling et al. [43] presented a novel approach to detect adversarial attacks on uncrewed aerial vehicles (UAVs) employing explainable deep reinforcement learning (DRL). As UAVs increasingly utilize AI techniques, the authors highlight the associated risks posed by adversarial attacks that can confuse the decision-making processes of these autonomous systems. The paper proposes a DRL-based guidance and planning framework that utilizes a Deep Deterministic Policy Gradient (DDPG) algorithm, augmented by Prioritized Experience Replay (PER) and an Artificial Potential Field (APF) to enhance training efficiency and obstacle avoidance. A simulated environment is created to evaluate the UAV's performance under various adversarial attacks generated by the Basic Iterative Method (BIM), which significantly reduces the UAV's operational success rate. To counter these adversarial attacks, the authors propose two detection mechanisms: a Convolutional Neural Network Adversarial Detector (CNN-AD) achieving 80% detection accuracy and a Long Short-Term Memory (LSTM) network reaching 91% accuracy with faster computing times. These detectors leverage explainability techniques, specifically SHapley Additive exPlanations (SHAP), to provide insights into the UAV's decision-making process. By monitoring slight variations in these decision processes, the proposed system aims to enable real-time detection of adversarial attacks, thereby ensuring the safe and reliable operation of UAVs in the presence

of potential threats. The study underscores the importance of integrating explainability into AI systems, particularly in the context of autonomous vehicles, to foster trust and reliability in critical applications.

Wu et al. [44] introduced a highly interpretable framework for detecting attacks on unmanned aerial vehicles (UAVs), focusing on the challenges posed by increasing cyber threats. Their proposed model, a CNN-BiLSTM-Attention (CBA) architecture, leverages real-time sensor data, including GPS and inertial measurement unit (IMU) readings, to detect various types of attacks such as denial-of-service (DoS) and GPS spoofing. The study emphasizes the limitations of existing detection methods, which often lack transferability and interoperability, and introduces the SHapley Additive exPlanations (SHAP) technique to enhance the interpretability of the detection model. This approach enables UAV security experts to better understand the model's decision-making process, thus improving trust and usability in practical applications. The authors reveal important insights into the relationships between UAV sensor data and different attack types through both local and global SHAP explanations. This understanding aids in identifying the most effective features for timely attack detection while operating within the limited computational resources of small commercial UAVs. The results demonstrate the CBA model's effectiveness and stability, showcasing its potential as a robust solution for UAV attack detection. The study highlights the importance of interpretability in AI-driven security systems, paving the way for more reliable UAV operations in the face of evolving cyber threats.

Ahmad et al. [45] explored the pressing need for robust cybersecurity measures in unmanned aerial vehicles (UAVs) by presenting a novel deep learning-based network intrusion detection system (NIDS). As UAVs increasingly rely on network connectivity for various applications, they become susceptible to cyber threats that can compromise their operational integrity and data confidentiality. The authors propose a hybrid model that utilizes convolutional neural networks (CNNs) for effective feature extraction and recurrent neural networks (RNNs) for sequence modeling, enabling the system to detect and classify network intrusions efficiently. Their approach is validated using a comprehensive dataset that simulates various attack scenarios, demonstrating high detection accuracy and low false-positive rates. The study emphasizes the adaptability of the proposed NIDS, showcasing its capability to learn and improve in response to evolving attack techniques. By leveraging deep learning methodologies, the system enhances the cybersecurity of UAVs and contributes to the overall safety and reliability of UAV operations. The authors underscore the critical importance of developing advanced NIDS tailored specifically for UAVs, particularly as these devices become increasingly integrated into critical infrastructure and autonomous systems. This research represents an advancement in ensuring the integrity and security of UAV networks, addressing a critical gap in the current literature on UAV cybersecurity.

Wang et al. [46] proposed an intelligent detection algorithm to combat GPS spoofing attacks on unmanned aerial vehicles (UAVs), a growing concern given the reliance of these systems on GPS for navigation and positioning. The study highlights the vulnerabilities of UAVs to spoofing attacks, where malicious actors can transmit false GPS signals, leading UAVs to deviate from their intended flight paths. The authors critique existing detection methods, which often suffer from low efficiency, limited application scenarios, and the need for costly equipment upgrades. To address these challenges, they introduce a novel approach utilizing Long Short-Term Memory (LSTM) networks, a machine learning technique that allows for the accurate detection of spoofing attacks. The proposed algorithm consists of two main components. First, it employs LSTM to predict the UAV's flight trajectory based on historical data, capturing key variables such as speed and direction. Second, it utilizes predefined flight paths to

validate the UAV's current GPS data against expected values. If discrepancies arise, the algorithm can swiftly identify potential GPS spoofing attacks. This method is particularly advantageous as it does not require significant computational resources or updates to existing GPS equipment, making it highly applicable in various UAV scenarios. Experimental results indicate that this approach effectively detects GPS spoofing attacks in a timely manner, marking an advancement in UAV security methodologies and the application of machine learning in this context.

1.3 Thesis Contributions

This thesis makes several significant contributions to the cybersecurity of quadcopters, particularly in developing multi-output deep learning models for cyber-attack detection, identification, and isolation. First, this work introduces a novel multi-output (MO) approach that simultaneously addresses the detection, identification, and isolation of cyber-attacks on quadcopters. Using a shared LSTM backbone with three output heads dedicated to these specific tasks, the model efficiently manages multiple outputs to produce comprehensive threat detection. Unlike previous studies, this approach enables a single model to handle all three functions, enhancing the performance and implementation flexibility in cyber-physical security systems.

Another unique aspect of this research is its pioneering use of sequence generation as a pre-processing task for time-series data, tailored specifically for LSTM-based models. The proposed sequence generation approach optimizes the data preparation stage, allowing for improved model accuracy and better handling of complex quadcopter data. This method outperforms existing implementations in the field, providing a foundation for future advancements in sequence-based cyber-attack detection methods for time-series data. Moreover, this study introduces the detection of three different types of cyber-attacks: False Data Injection (FDI), Denial of Service (DoS), and replay attacks, demonstrating the robustness of the proposed approach under diverse threat scenarios.

Expanding beyond a single-quadcopter model, this research extends the multi-output (MO) approach to support a network of quadcopters, introducing an adaptable LSTM-based architecture designed to process data from multiple quadcopters simultaneously. The proposed architecture includes an LSTM input head for each quadcopter in the network and adds three new output heads dedicated to cyber-attack detection, identification, and isolation. This multi-input multi-output (MIMO) framework allows the model to accommodate varying numbers of quadcopters, making it adaptable for both centralized and decentralized network topologies. Notably, the MIMO model has been successfully tested on networks of two to five quadcopters, demonstrating scalability and applicability in a range of operational environments.

This work introduces two extensive datasets to support cyber-attack research on quadcopters, enhancing the scope and applicability of this study. The first dataset focuses on a single quadcopter and includes 50,000 data points, capturing 10 sensor and actuator features with three labels dedicated to detection, identification, and isolation tasks. This single-quadcopter dataset provides a streamlined, focused resource for investigating attack scenarios on individual drones. The second dataset, designed for a network of quadcopters, also comprises 50,000 data points collected from five quadcopters, with 50 sensor and actuator features and 15 labels. This comprehensive network dataset is particularly valuable for training and testing models suited to multi-quadcopter setups, supporting both centralized

and decentralized network configurations. Both datasets are useful, with the single-quadcopter dataset optimizing model performance in individual attack scenarios and the network dataset enabling complex model development.

A key feature of the MIMO model is its ability to manage output complexity by using output reduction, decreasing the output count from 15 to 3, which significantly improves model manageability and learning capacity. This reduction approach streamlines the training process while maintaining high model performance. Furthermore, the MIMO model is uniquely capable of supporting simultaneous attacks, a feature does not present in previous MO models. This capability significantly enhances the system's resilience by allowing it to recognize to multiple concurrent threats, a critical requirement for ensuring the cybersecurity of modern drone networks in dynamic environments.

The proposed MIMO model, with its novel LSTM-based input heads, shared backbone, and flexible architecture, provides a new benchmark for cyber-attack detection and management in multi-drone networks. By supporting centralized and decentralized network topologies, this model offers practical flexibility and can adapt to diverse operational requirements. Moreover, the research findings suggest that centralized topologies yield the highest performance, though the model is sufficiently adaptable to function effectively in decentralized settings. This adaptability makes it highly suitable for real-world applications where network configurations may vary.

In summary, this thesis contributes a novel and comprehensive framework for cybersecurity in quadcopter networks, from dataset creation and model architecture to the implementation of a robust detection, identification, and isolation approach. These advancements lay the groundwork for future research in multi-drone cyber-attack resilience and represent a meaningful step forward in the integration of deep learning for the security of cyber-physical systems.

1.4 Thesis Outline

This thesis is structured into five chapters, each addressing a specific component of the research in cyber-attack detection, identification, and isolation within quadcopter systems and networks. Chapter 2 begins by presenting essential background information, covering the modeling and control mechanisms of quadcopters, which are fundamental to understanding the impact of cyber-attacks on these systems. This chapter also includes an overview of the types of cyber-attacks relevant to this study, such as False Data Injection (FDI), Denial of Service (DoS), and replay attacks.

Chapter 3 delves into cyber-attack detection, identification, and isolation specifically within a single quadcopter context. This chapter describes the process of dataset creation, focusing on capturing attack and normal scenarios for effective model training. It introduces the novel multi-output (MO) approach, leveraging LSTM architecture to detect, identify, and isolate cyber-attacks simultaneously. Detailed explanations of sequence generation and its role as a pre-processing step for time-series data are provided, emphasizing its impact on model accuracy and robustness. The chapter terminates with a thorough evaluation of the proposed methodology, using multiple experiments to assess its efficacy in handling single-quadcopter attack scenarios.

Chapter 4 extends the MO approach to a more complex environment, a network of quadcopters. This chapter addresses the new challenges that arise in multi-quadcopter systems, such as handling increased

data complexity. It introduces the Multi-Input, Multi-Output (MIMO) model, which accommodates multiple quadcopters within a shared framework for detecting, identifying, and isolating cyber-attacks across a network. This chapter presents a detailed explanation of the MIMO model architecture, highlighting how it leverages input heads for each quadcopter and a shared backbone to produce outputs. Experimental results are provided, demonstrating the model's effectiveness in networked environments and showcasing its adaptability to both centralized and decentralized topologies.

Finally, Chapter 5 concludes the thesis by summarizing the key findings from each chapter and discussing their implications for the cybersecurity of quadcopters networks. This concluding chapter also proposes directions for future research, including potential improvements to the model architecture, the exploration of additional attack scenarios, and the application of the proposed methodology to other types of cyber-physical systems.

Chapter 2

2. Background Information

2.1 Cyber-Physical System Overview

Cyber-Physical Systems (CPS) represent a convergence of embedded computing and communication technologies designed to monitor, control, and interact with physical elements. These systems integrate physical infrastructure such as sensors and actuators, communication networks for data transmission, and computational frameworks for processing and decision-making. A commonly used framework for representing CPS divides them into three interconnected layers. The physical layer includes components like sensors, actuators, and plants that facilitate the monitoring and execution of physical processes. The network layer serves as the intermediary, transmitting data between the physical and computational layers, enabling seamless communication. At the top lies the cyber layer, which abstracts, processes, and analyzes received data to make intelligent decisions and issue control commands [47].

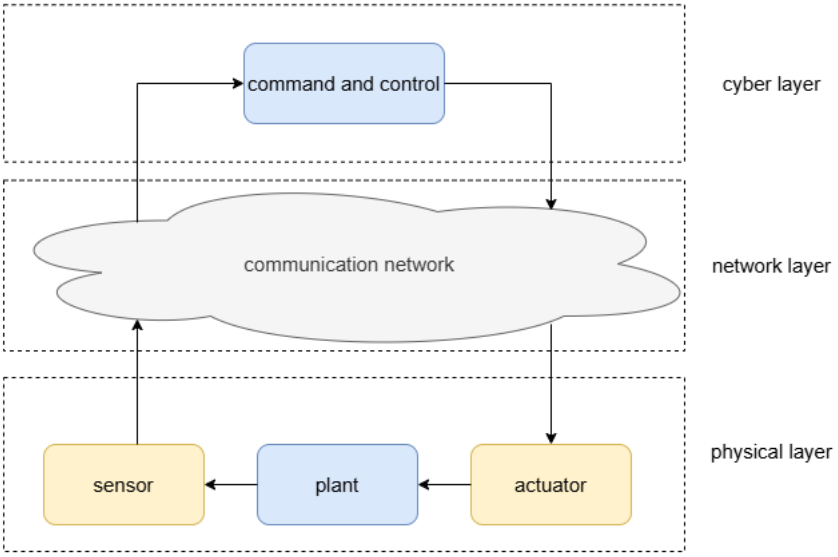


Figure 2.1: Overview of CPS.

As depicted in the Figure 2.1, the architecture of CPS emphasizes the interplay between these layers. Sensors in the physical layer gather data from the plant and relay it to the cyber layer through the network layer. The cyber layer, in turn, processes this information to generate control commands, which are transmitted back through the network layer to actuators in the physical layer for execution. This layered structure ensures efficient real-time operation and control. However, the interconnected nature of CPS also introduces vulnerabilities, particularly in the network layer, which is often the target of cyber-attacks. This highlights the importance of robust designs to safeguard these systems, ensuring both their functionality and security.

2.1.1 CPS Use-Case Selection

Cyber-Physical Systems (CPS) represent a transformative integration of computational elements with physical processes, enabling real-time monitoring and control across various domains. This integration facilitates seamless interaction between physical systems and their digital counterparts, allowing for enhanced functionality and efficiency. CPS encompasses a wide array of applications, including smart grids, autonomous vehicles, healthcare systems, and unmanned aerial vehicles (UAVs), each leveraging the synergy between physical and cyber components to improve performance and reliability [48; 49].

In the context of smart grids, CPS enhances the management and distribution of electricity by integrating information and communication technologies with traditional power systems. This integration enables two-way communication between utilities and consumers, facilitating real-time monitoring, demand response, and improved grid reliability. The Cyber-Physical Power System (CPPS) allows for advanced

functionalities such as automated fault detection and self-healing capabilities, which are crucial for maintaining the stability and efficiency of modern energy systems [48].

Autonomous vehicles are another prominent application of CPS, where the interplay between sensors, control algorithms, and physical dynamics is critical for safe navigation. These vehicles utilize a network of sensors and communication technologies to perceive their environment, make real-time decisions, and interact with other vehicles and infrastructure. The development of Cooperative Open Cyber-Physical Systems (CO-CPS) in this domain emphasizes the importance of wireless communication and collaboration among multiple stakeholders to ensure safety and efficiency in dynamic driving conditions [50].

In healthcare systems, CPS plays a vital role in patient monitoring and management, particularly highlighted during the COVID-19 pandemic. The integration of IoT devices, cloud computing, and machine learning within CPS frameworks enables real-time health monitoring, data collection, and analysis, facilitating timely interventions and personalized care. This approach not only enhances patient outcomes but also optimizes resource allocation within healthcare facilities [51].

Lastly, UAVs exemplify the application of CPS in aerial operations, where the combination of physical flight dynamics and cyber capabilities allows for a wide range of functionalities, from surveillance to delivery services. The system-of-systems approach in UAVs enables the coordination of multiple drones to perform complex tasks collaboratively, enhancing operational efficiency and effectiveness in various applications, including disaster response and environmental monitoring [52].

The selection of Unmanned Aerial Vehicles (UAVs), particularly quadcopters, as a use case for Cyber-Physical Systems (CPS) is driven by their versatility, maneuverability, and the growing demand for autonomous operations in various sectors. Quadcopter UAVs are characterized by their ability to perform complex aerial maneuvers and operate in diverse environments, making them suitable for applications ranging from surveillance and disaster response to delivery services and agricultural monitoring. Their reliance on advanced communication systems for navigation and control underscores their role as a quintessential CPS, where the integration of physical flight dynamics with cyber capabilities is essential for real-time decision-making and operational efficiency [53; 54]. Moreover, the increasing sophistication of wireless technologies enhances the potential for quadcopters to communicate and collaborate, further solidifying their relevance in CPS frameworks [54].

Quadcopters can be deployed in two primary formats: as single units or as networks of multiple units. A single quadcopter operates independently, utilizing onboard sensors and control algorithms to navigate and perform tasks autonomously. This format is particularly effective for applications requiring localized data collection or targeted interventions, such as aerial photography or environmental monitoring [55]. In contrast, a network of quadcopters, often referred to as a swarm, leverages cooperative strategies to execute complex missions that would be challenging for a single unit. This networked approach enhances operational capabilities, allowing for coordinated tasks such as search and rescue operations, where multiple UAVs can cover larger areas more efficiently and share data in real-time to improve situational awareness [31]. The integration of multiple quadcopters into a cohesive system exemplifies the potential of CPS to enhance performance and reliability through collaboration and communication among distributed agents [56].

2.2 Quadcopter Modeling

2.2.1 Modeling of the Nonlinear Quadcopter

This section introduces the nonlinear model of quadcopter kinematics and rigid-body dynamics. Quadcopters are underactuated systems with four control inputs and six degrees of freedom. The quadcopter rigid body dynamics is analysis and simplified to develop a control system at equilibrium point. The modeling of quadcopter is based on following assumptions [57; 58].

- The range of pitch movement and roll movement is small.
- The quadcopter frame is symmetrical.
- The whole quadcopter is a rigid body.
- The inertia of the motor is small and neglected.
- The center of frame matches the center of mass.

Kinematics

There are two primary configurations for quadcopter frames: the "X" configuration and the "+" configuration, as illustrated in Figure 2.2. In pitch or roll maneuvers, quadcopters with the "+" configuration use only two rotors, while those with the "X" configuration engage all four rotors. The "X" configuration is used in this work to maximize the available torque for roll and pitch movements by utilizing all four rotors [57].

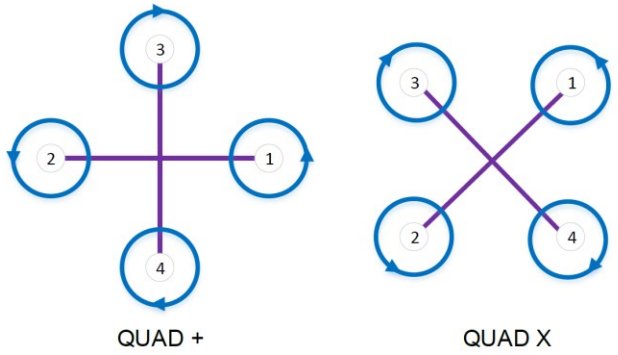


Figure 2.2: Two different quadcopter frame setups [57].

Table 2.1 presents the mathematical symbols employed within the modeling.

Table 2.1: Mathematical symbols for modeling [57].

Symbol	Meaning
$[x, y, z]$	Position of quadcopter in the inertial frame.
$[x_b, y_b, z_b]$	Position of quadcopter in body frame.
$[v_x, v_y, v_z]$	Velocity of the quadcopter in the inertial frame.
$[\Phi, \theta, \Psi]$	Angular position of the quadcopter in the inertial frame.
$[p, q, r]$	Angular rate of the quadcopter in body frame.

In inertial frame, x axis points east, y axis points north, and z axis points up. Roll angle (Φ) represents rotation along x axis. Pitch angle (θ) represents rotation along y axis. Yaw angle (Ψ) represents rotation along z axis.

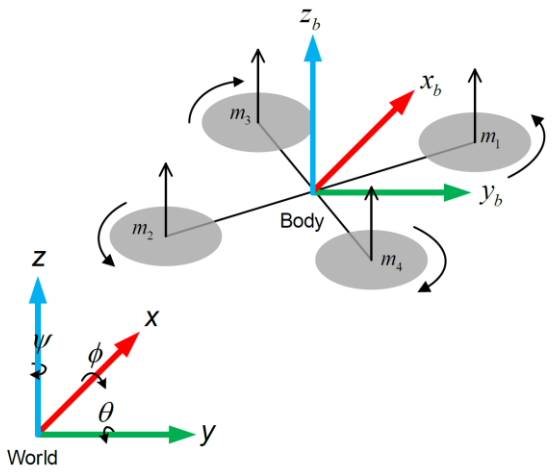


Figure 2.3: Quadcopter coordinate diagram in body and inertial frame [57].

Rotation matrix is based on Z-Y-X Euler angles to present rigid-body vector that rotates from body frame to inertial frame as demonstrated in Figure 2.3.

Rotation about z axis is given by:

$$R_\psi = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Rotation about y axis is given by:

$$R_\theta = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}. \tag{2.2}$$

Rotation about x axis is given by:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{2.3}$$

Then the rotation matrix from body coordinate to inertial coordinate is given by:

$$R_{E\to B} = R_\phi R_\theta R_\psi, \tag{2.4}$$

This is equivalent to:

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & \sin(\theta) \\ -\cos(\phi)\sin(\psi)+\cos(\psi)\sin(\phi)\sin(\theta) & \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\theta)\sin(\phi) \\ -\cos(\phi)\cos(\psi)\sin(\theta)-\sin(\phi)\sin(\psi) & \cos(\psi)\sin(\phi)-\cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}. \tag{2.5}$$

The above matrix is orthonormal. So, the rotation matrix from body coordinate to inertial coordinate can be calculated by taking its transpose.

$$R_{B\to E} = R_{E\to B}^{T}, \tag{2.6}$$

This is equivalent to:

$$\begin{bmatrix} \cos(\psi)\cos(\theta) & -\cos(\phi)\sin(\psi)+\cos(\psi)\sin(\phi)\sin(\theta) & -\cos(\phi)\cos(\psi)\sin(\theta)-\sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\psi)\sin(\theta) & \cos(\psi)\sin(\phi)-\cos(\phi)\sin(\psi)\sin(\theta) \\ \sin(\theta) & -\cos(\theta)\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}. \tag{2.7}$$

Therefore:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_{B\to E}\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}. \tag{2.8}$$

The Euler rates of the quadcopter is same as other aircrafts. It can be used to calculate the attitude of the quadcopter [57]. The relation between the Euler rates and the body angular rates is:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_{\phi}\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_{\phi}R_{\theta}\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \Omega_{E\to B}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.9}$$

In which:

$$\Omega_{E\to B} = \begin{bmatrix} 1 & 0 & \sin(\theta) \\ 0 & \cos(\phi) & -\sin(\phi)\cos(\theta) \\ 0 & \sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}. \tag{2.10}$$

Additionally:

$$\Omega_{B\to E} = \Omega_{E\to B}^{-1} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & -\cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\dfrac{\sin(\phi)}{\cos(\theta)} & \dfrac{\cos(\phi)}{\cos(\theta)} \end{bmatrix}. \tag{2.11}$$

Finally:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{cases} p + \sin(\phi)\tan(\theta)q - \cos(\phi)\tan(\theta)r, \\ \cos(\phi)q + \sin(\phi)r, \\ -\dfrac{\sin(\phi)}{\cos(\theta)}q + \dfrac{\cos(\phi)}{\cos(\theta)}r, \end{cases} \tag{2.12}$$

Dynamics

Based on Newton's second law of motion, the mass center dynamic equation of quadcopter is given by:

$$\frac{d(m\vec{V})}{dt} = \vec{F}, \tag{2.13}$$

The thrust generated by each motor is T_i ($i = 1, 2, 3, 4$). Therefore, the total thrust is given by:

$$T = T_1 + T_2 + T_3 + T_4, \tag{2.14}$$

The differential thrust generated by 4 motors generates pitch moment and roll moment.

l is the distance between each motor and the center of the frame.

Roll Movement: For moving in positive y direction, the rotation speed of motor 1 and 4 is increased and that of motor 2 and 3 is decreased as illustrated in Figure 2.4.

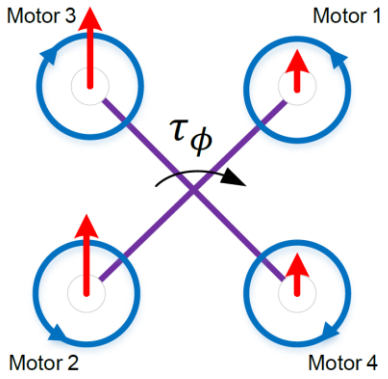


Figure 2.4: Roll movement [57].

Pitch Movement: For moving in positive x direction, the rotation speed of motor 2 and 4 is increased and the rotation speed of motor 1 and 3 is decreased as illustrated in Figure 2.5.

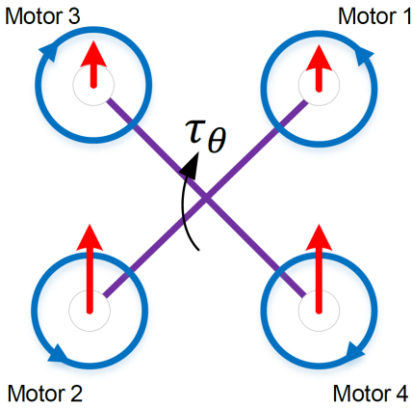


Figure 2.5: Pitch movement [57].

Yaw Movement: For moving quadcopter around z axis in body frame, the rotation speed of motor 1 and 2 is increased and that of motor 3 and 4 is decreased as illustrated in Figure 2.6.

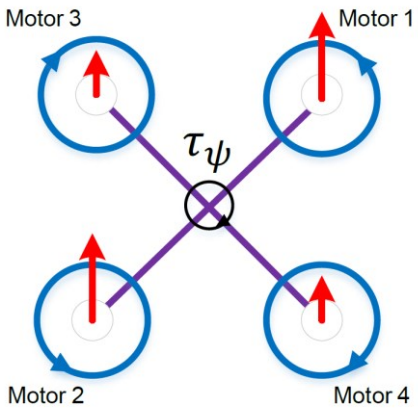


Figure 2.6: Yaw movement [57].

Therefore:

$$\tau_\phi = \frac{\sqrt{2}}{2} l(T_2 + T_3 - T_1 - T_4), \tag{2.15}$$

Additionally,

$$\tau_\theta = \frac{\sqrt{2}}{2} l(T_1 + T_3 - T_2 - T_4), \tag{2.16}$$

The inverse torque required to generate yaw moment is generated by each motor, where m_i $(i = 1, 2, 3, 4)$. The total inverse torque generated by four motors is given by:

$$\tau_\psi = m_1 + m_2 - m_3 - m_4, \tag{2.17}$$

The dynamics equation of motion is given by:

$$m \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = mg \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} + R_{B \to E} T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

Therefore, position movement of the quadcopter can be expressed as:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}\dot{v}_x\\ \dot{v}_y\\ \dot{v}_z\end{bmatrix} = \begin{bmatrix}\frac{T}{m}(-cos(\psi)\,sin(\theta)\,cos(\phi) - sin(\psi)\,sin(\phi))\\ \frac{T}{m}(cos(\psi)\,sin(\phi) - sin(\psi)\,sin(\theta)\,cos(\phi))\\ \frac{T}{m}cos(\phi)\,cos(\theta) - g\end{bmatrix}. \quad (2.18)$$

The rotation kinematics equation of quadcopter is given by:

$$\frac{d(Jv)}{dt} = M, \quad (2.19)$$

In the above equation, $J = \mathrm{diag}[J_x, J_y, J_z]$ is quadcopter moments of inertia for 3 axes of body coordinate system. M is the moment applied on the quadcopter.

Body torque generated by rotors is given by:

$$\begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} = \begin{bmatrix}\frac{\sqrt{2}}{2}l(T_2 + T_3 - T_1 - T_4)\\ \frac{\sqrt{2}}{2}l(T_1 + T_3 - T_2 - T_4)\\ m_1 + m_2 - m_3 - m_4\end{bmatrix}. \quad (2.20)$$

Aerodynamic drag torque is given by:

$$\tau_{af} = K_{af}v, \quad (2.21)$$

where $K_{af} = \mathrm{diag}[k_{afx}, k_{afy}, k_{afz}]$.

Therefore, dynamics equation of torque is given by:

$$M = \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} = \begin{bmatrix}\dot{p}J_x + qr(J_z - J_y)\\ \dot{q}J_y + pr(J_x - J_z)\\ \dot{r}J_z + pq(J_y - J_x)\end{bmatrix} = \begin{bmatrix}\tau_\phi - \tau_{afx}\\ \tau_\theta - \tau_{afy}\\ \tau_\psi - \tau_{afz}\end{bmatrix}. \quad (2.22)$$

Finally, the equations of angular movement of the quadcopter are given by:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}\frac{J_y-J_z}{J_x}qr+\frac{\tau_\phi}{J_x}-\frac{\tau_{afx}}{J_x}\\ \frac{J_z-J_x}{J_y}pr+\frac{\tau_\theta}{J_y}-\frac{\tau_{afy}}{J_y}\\ \frac{J_x-J_y}{J_z}pq+\frac{\tau_\psi}{J_z}-\frac{\tau_{afz}}{J_z}\end{bmatrix}. \tag{2.23}$$

Then the whole nonlinear quadcopter equations of kinematics and dynamics given by:

$$\dot{X}=\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{v}_x\\ \dot{v}_y\\ \dot{v}_z\\ \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\\ \dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}=\begin{cases}\dot{x}=v_x,\\ \dot{y}=v_y,\\ \dot{z}=v_z,\\ \dot{v}_x=\frac{T}{m}(-\cos(\psi)\sin(\theta)\cos(\phi)-\sin(\psi)\sin(\phi)),\\ \dot{v}_y=\frac{T}{m}(\cos(\psi)\sin(\phi)-\sin(\psi)\sin(\theta)\cos(\phi)),\\ \dot{v}_z=\frac{T}{m}(\cos(\phi)\cos(\theta))-g,\\ \dot{\phi}=p+\sin(\phi)\tan(\theta)\,q-\cos(\phi)\tan(\theta)\,r,\\ \dot{\theta}=\mathrm{qcos}(\phi)+\mathrm{rsin}(\phi)\,,\\ \dot{\psi}=-\frac{\sin(\phi)}{\cos(\theta)}q+\frac{\cos(\phi)}{\cos(\theta)}r,\\ \dot{p}=\frac{(J_y-J_z)}{J_x}qr+\frac{\tau_\phi}{J_x},\\ \dot{q}=\frac{J_z-J_x}{J_y}pr+\frac{\tau_\theta}{J_y},\\ \dot{r}=\frac{J_x-J_y}{J_z}pq+\frac{\tau_\psi}{J_z},\end{cases} \tag{2.24}$$

This system is nonlinear, it should be linearized for controller design in the next section. The linearization should be at a near-hover state where ϕ, θ, ψ are close to zero.

2.2.2 Quadcopter Modeling Linearization

Original nonlinear or linearized model of the quadcopter can be employed, following is a comparison between using linear or nonlinear model of quadcopter.

Linear Models: Linear models are favored due to their ease of comprehension and implementation. Controllers required for linear models are relatively simpler to tune and design. Their straightforward nature facilitates better understanding and straightforward implementation within the system.

Nonlinear Models: Contrarily, nonlinear systems provide more accurate representations of real-world scenarios. However, their complexity poses challenges in understanding and implementation, particularly in formulating these intricate models. Controllers designed for nonlinear systems are notably more complex and demand a deeper understanding of their design and implementation.

In this work, while the nonlinear model of the quadcopter is considered, the complexity of designing a full controller for this system necessitates the use of a linearized model for controller design. The controller, developed based on the linearized system, is then applied to control the actual nonlinear system. The nonlinear model of quadcopter is represented as follows:

$$\dot{X} = f(X, U), \tag{2.25}$$

$$X = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{bmatrix}. \tag{2.26}$$

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{2.27}$$

where X represents the state vector and U represents the input vector.

Linearization is performed around steady-state operating conditions, that is trim points, where the quadcopter is in equilibrium. At these trim points:

$$X_{trim} = [0,0,0,0,0,0,0,0,0,0,0,0]^T, \tag{2.29}$$

$$U_{trim} = [mg, 0,0,0]^T, \tag{2.30}$$

$$f(X_{trim}, U_{trim}) = 0, \tag{2.31}$$

where X_{trim} is state variable and U_{trim} is control inputs in equilibrium point, and the equation represents steady state condition.

The characteristic matrix A and the input matrix B are calculated as follows:

$$A = \frac{\delta f}{\delta \mathrm{X}}|_{\mathrm{X}=\mathrm{X}_{\mathrm{equil}},\mathrm{U}=\mathrm{U}_{\mathrm{equil}}}, \tag{2.32}$$

$$B = \frac{\delta f}{\delta \mathrm{U}}|_{\mathrm{X}=\mathrm{X}_{\mathrm{equil}},\mathrm{U}=\mathrm{U}_{\mathrm{equil}}}, \tag{2.33}$$

A and B matrixes are represented as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.34}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \end{bmatrix}. \tag{2.35}$$

Therefore, the linearized quadcopter rigid-body state-space formula is given by:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ -g\theta \\ g\phi \\ \frac{T}{m} - g \\ p \\ q \\ r \\ \frac{\tau_\phi}{J_x} \\ \frac{\tau_\theta}{J_y} \\ \frac{\tau_\psi}{J_z} \end{bmatrix} = A\mathrm{X} + B\mathrm{U}, \tag{2.36}$$

C and D matrixes are represented as follows:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{2.37}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.38}$$

Therefore, output of the system will be position and angular position of quadcopter in inertial frame.

$$Y = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = C\mathrm{X} + DU, \tag{2.39}$$

Table 2.2 presents the constant values utilized to configure the quadcopter.

Table 2.2: Quadcopter configuration values [57].

Parameter	Definition	Value
J_x	Moment of Inertia in x axis	0.0019005
J_y	Moment of Inertia in y axis	0.0019536
J_z	Moment of Inertia in z axis	0.0036894
m	Mass	0.551
g	Gravity constant	9.8

2.3 Control System Design

2.3.1 Determine Control System Objective

The control of a nonlinear quadcopter to follow simple trajectories around a designated start point (0, 0, 0) involves several critical control objectives, namely attitude control, altitude control, position control, and yaw control. Each of these components plays a vital role in ensuring the quadcopter maintains stability and executes its flight patterns effectively.

1) Attitude Control: This aspect focuses on managing the quadcopter's orientation along its three axes: roll, pitch, and yaw. Effective attitude control is essential for maintaining stability during flight and is often achieved through various control strategies, including PID (Proportional-Integral-Derivative) controllers. The complexity of quadcopter dynamics, characterized by high maneuverability and nonlinearity, necessitates sophisticated control algorithms to ensure precise orientation management [59]. Research indicates that optimal PID control methods can significantly enhance the quadcopter's ability to maintain its desired attitude during various maneuvers [59].
2) Altitude Control: Regulating the quadcopter's vertical position is crucial for achieving desired flight characteristics. This is accomplished by adjusting the thrust produced by the rotors, allowing the quadcopter to ascend, descend, or maintain a specific altitude. The altitude control system must respond dynamically to changes in weight and environmental conditions, which can affect lift and stability [60]. The integration of sensors such as accelerometers and inertial measurement units (IMUs) is vital for accurate altitude measurement and control [60].
3) Position Control: Position control governs the quadcopter's movement in horizontal space, ensuring it follows a predetermined trajectory or remains in a specific location. This control objective is closely linked to both altitude and attitude control, as maintaining a stable position requires coordinated adjustments across all axes. The implementation of waypoint navigation systems has been shown to enhance position control, allowing quadcopters to autonomously navigate to specified coordinates [61]. The use of advanced algorithms and real-time feedback from sensors is critical for achieving high accuracy in position control [62].
4) Yaw Control: While yaw control is not the primary focus of this work, it is still an important aspect of quadcopter dynamics. Managing the yaw angle involves tuning input values to the

actuators to ensure the quadcopter can turn smoothly and maintain its intended flight path. However, for the purpose of following simple trajectories, the emphasis remains on altitude and position control, as these are more critical for basic flight operations [63].

The distinction between simple and complex trajectories is also significant in the context of quadcopter control. Simple trajectories involve straightforward maneuvers such as hovering, ascending, or following straight paths. These predictable flight patterns require less sophisticated control algorithms, making them suitable for initial training and basic operations [63]. In contrast, complex trajectories demand advanced control strategies due to their intricate nature, which includes sharp turns and high-speed navigation through obstacles. Mastering these complex trajectories enhances the quadcopter's capabilities for specialized applications, such as aerial surveillance [62].

2.3.2 Controller Type Selection

Quadcopters utilize a variety of control mechanisms to effectively manage their flight dynamics, with several prominent controllers designed to enhance their stability and maneuverability. Among these, the Proportional-Integral-Derivative (PID) Controller, the Linear Quadratic Regulator (LQR) Controller, and the State Feedback Controller are widely recognized for their respective advantages and applications in quadcopter control.

1) Proportional-Integral-Derivative (PID) Controller: The PID controller is a fundamental component in quadcopter control systems, operating by adjusting the output based on proportional, integral, and derivative terms. The proportional term addresses the current error, the integral term accumulates past errors, and the derivative term anticipates future errors. This combination makes the PID controller both versatile and effective, particularly in maintaining stable flight characteristics [64]. Its simplicity and ease of implementation have made it a preferred choice among researchers and practitioners in the field of drone technology [64]. Furthermore, the ability to stack PID controllers into cascaded configurations allows for enhanced control over complex flight dynamics [65].
2) Linear Quadratic Regulator (LQR) Controller: The LQR controller is primarily applied in linear systems and is designed to compute optimal control gains that minimize a quadratic cost function while ensuring system stability. Although the LQR controller is effective in controlling quadcopters, its reliance on a linear system model and the complexity of its implementation can limit its widespread application [66]. Research indicates that LQR can be effectively combined with other control strategies, such as backstepping control, to enhance performance in quadcopter applications [66]. This adaptability allows for improved control in various flight conditions, although it may require more computational resources compared to simpler controllers like PID.
3) State Feedback Controller: This controller leverages feedback from the entire system's state variables, providing enhanced flexibility and performance by directly accessing the system's state space for control law design. While the State Feedback Controller can offer superior performance, its complexity and higher computational demands may restrict its use in some quadcopter applications [66]. The integration of state feedback with LQR methods has been explored to optimize control strategies further, demonstrating the potential for improved maneuverability and stability in quadcopter flight [66].

In summary, the PID controller remains the most widely utilized control mechanism in quadcopter applications due to its simplicity, effectiveness, and ease of tuning. The LQR and State Feedback Controllers, while offering advanced capabilities, present challenges in terms of implementation complexity and computational requirements. As quadcopter technology continues to evolve, the integration of these control strategies will play a crucial role in enhancing flight performance and expanding the operational capabilities of these versatile aerial vehicles.

2.3.3 Controller Design

The designed controller comprises three primary components:

1) Altitude Controller: This controller regulates the height of the quadcopter. It employs a single PID controller, which gets the difference between the actual altitude (z) and the reference altitude (z_ref) as input to determine the necessary thrust output.
2) Yaw Controller: This component adjusts the yaw angle, or rotation around the z-axis, of the quadcopter. Although controlling the yaw angle is not a primary objective of this work, it utilizes a PID controller that receives zero input, thereby maintaining a constant yaw value. The desired yaw angle is consistently set to zero.
3) Position Controller: This controller manages the quadcopter's x, and y coordinates and represents the most complex aspect of the overall control system. It employs a cascaded control structure consisting of four PID controllers. The outer controller functions as the position controller, while the inner controller serves as the attitude controller. Tuning is conducted first on the inner controller, followed by the outer controller.
    a. Attitude Controller: This component consists of two parts: the first gets the difference between the roll angle and the roll reference as input, outputting the required torque for roll; the second does the same for pitch. The roll and pitch reference inputs are provided by the outer loop position controller.
    b. Position Controller: This part also contains two components: the first gets the difference between the actual x position and the reference x position as input, producing a pitch reference output; the second calculates the difference for y, producing a roll reference output. The outputs from this controller feed into the inner attitude controller.

These PID controllers are crucial for achieving precise control over the quadcopter's movements, enabling stable flight and the execution of desired trajectories.

The PID parameters are optimized using the Simulink auto-tuner tool, without the need for specific criteria related to system response, such as rise time, settling time, peak overshoot, or steady-state error. Simulink automatically identifies the optimal response for the specified variable, eliminating the necessity for manual adjustments. To fine-tune each controller, all other controllers are temporarily set to a zero reference, allowing for precise tuning of the specified PID controller using a step input as a reference. Experimental results indicate that these controllers are functioning effectively.

The controller design and tuning are initially conducted on a linearized system, facilitating easier control and adjustment of the six PID controllers. Following the tuning process, the overall controller exhibits adequate control over the linearized quadcopter around the equilibrium point, defined as (0, 0, 0).

Subsequently, the same controller is applied to the nonlinear quadcopter, yielding satisfactory results across all designed test trajectories.

Given that the designed controller transitions from a linearized to a nonlinear system, it presents certain limitations, such as an inability to perform complex maneuvers and a tendency to drift far from the trim point. In simulations, the quadcopter is allotted 10 seconds to transition to the next position and is constrained to a maximum deviation of 5 units along each axis from the trim point.

Values of controller gains are demonstrated in Table 2.3.

Table 2.3: Gain values of PID controllers.

Number	Controller Output	Proportional	Integral	Derivative
1	Thrust	0.995151437850186	0.128436617904908	1.76652121748459
2	Torque Roll	0.00283727985172173	0.000300746607233676	0.00592713753900793
3	Torque Pitch	0.00376352907272366	0.000541941810599823	0.00599672187002239
4	Torque Yaw	0.00651371850229212	0.000840676044468491	0.0115626843326264
5	Reference Roll	0.0344011339376346	0.00030706355489176	0.0770084502146799
6	Reference Pitch	-0.0353666022144165	-0.000286794158433156	-0.0628063352706131

Figure 2.7 depicts the designed controller for the quadcopter system.

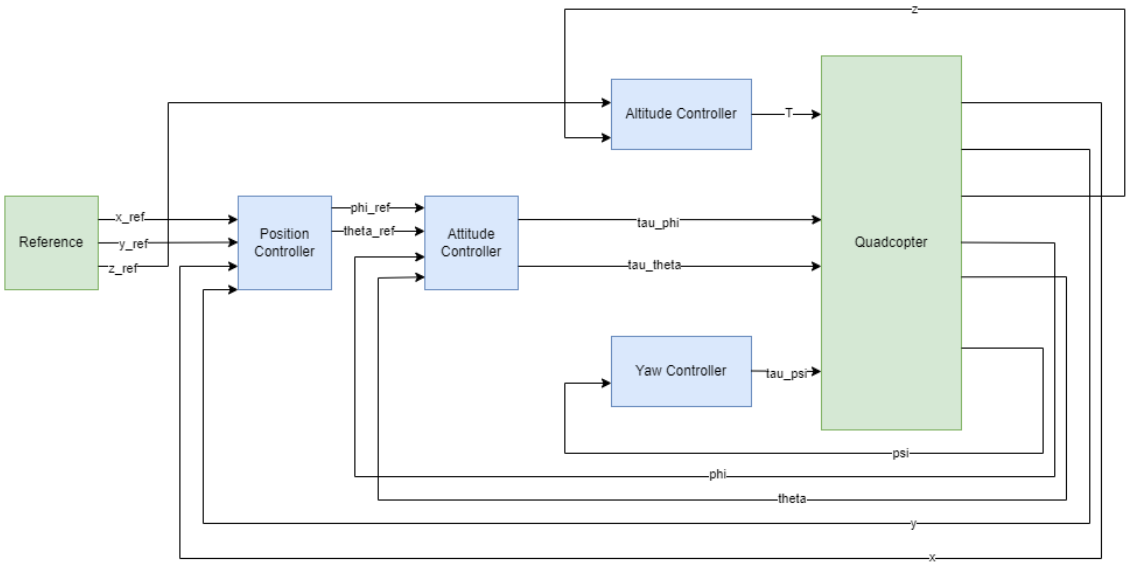


Figure 2.7: Designed PID controller.

2.4 Cyber-Attacks

2.4.1 Cyber Security of Cyber-Physical Systems

Cyber-physical systems (CPS) represent a convergence of computational and physical processes, necessitating robust cybersecurity measures to protect against a range of vulnerabilities. The inherent complexity of CPS, which includes interconnected sensing, computation, and actuation components, exposes them to sophisticated cyber threats that can lead to significant physical damage. Notable incidents such as the StuxNet attack and the Maroochy Shire sewage control incident exemplify the catastrophic consequences of cyber-attacks on industrial systems, highlighting the urgent need for effective cybersecurity strategies [67]. Furthermore, as the security landscape evolves, the focus has shifted from traditional physical security concerns to a more integrated approach that considers both cyber and physical layers of security [68]. This transition underscores the necessity for comprehensive security frameworks that can address the unique challenges posed by the tight coupling of cyber and physical elements within CPS [69]. The growing reliance on Internet of Things (IoT) technologies within CPS further amplifies these vulnerabilities, necessitating advanced detection systems capable of identifying and mitigating cyber-physical threats [70].

Cyber-attacks on cyber-physical systems pose significant risks, particularly in critical infrastructures such as the smart grid, where the interplay between cyber and physical components can lead to cascading failures [71]. The nature of these attacks can vary widely, from denial-of-service attacks that disrupt system operations to more sophisticated intrusions aimed at manipulating control objectives [67]. Research indicates that the detection of such attacks is crucial for maintaining the safety and reliability of CPS, as traditional monitoring methods may not suffice in identifying complex threats that exploit both cyber and physical vulnerabilities [68]. The integration of hardware-in-the-loop simulations has proven effective in evaluating the impact of cyber-attacks on CPS, revealing the necessity for a dual-layered security approach that encompasses both cyber defenses and physical safeguards [68]. As the landscape of cyber threats continues to evolve, a comprehensive understanding of the attack vectors and potential defenses is essential for securing CPS against emerging risks [71].

Five distinct types of cyber-attacks are defined as follows:

1) Denial-of-Service (DoS) Attack: A Denial-of-Service attack aims to disrupt the normal functioning of a system by overwhelming it with excessive requests or by jamming communication channels. This results in the inability of legitimate users to access the services provided by the system, effectively blocking the exchange of information between system components. DoS attacks can lead to significant operational failures, particularly in critical infrastructures such as smart grids and industrial control systems. The necessity for robust security measures to ensure uninterrupted service delivery is underscored by the prevalence of DoS attacks in CPS [72].
2) False Data Injection (FDI) Attack: False Data Injection attacks involve the malicious manipulation of data inputs to a system, which can lead to erroneous control actions. In smart grids, for instance, FDI can cause the control center to execute incorrect commands, potentially resulting in system instabilities and even blackouts. This type of attack is particularly concerning because it can go undetected while causing significant damage to the operational integrity of the system. The implications of FDI attacks highlight the need for effective detection and mitigation strategies in CPS [73].

3) Replay Attack: Replay attacks involve the interception and subsequent retransmission of valid data messages. This can mislead the system into believing it is receiving legitimate information, thus allowing an attacker to manipulate system responses without detection. Such attacks exploit the temporal nature of data communication in CPS, making them a significant concern for system security. The stealthy nature of replay attacks necessitates the implementation of robust authentication and timestamping mechanisms to mitigate their impact [74].
4) Zero Dynamic Attack: Zero dynamic attacks are a sophisticated form of attack that targets the system's control dynamics. By exploiting the system's response characteristics, attackers can manipulate the system's behavior without raising alarms. This stealthy approach can lead to severe consequences, as the system may operate under false assumptions of safety and stability. The complexity of zero dynamic attacks necessitates advanced detection techniques that can identify subtle deviations in system behavior [75].
5) Covert Attack: Covert attacks are designed to be stealthy and undetectable, allowing attackers to manipulate system operations without triggering alarms or detection mechanisms. The Stuxnet malware is a prime example, as it was engineered to damage specific physical systems while remaining hidden from operators. Such attacks pose a critical threat to the security of CPS, as they can lead to catastrophic failures without any immediate indication of compromise. The need for comprehensive monitoring and anomaly detection systems is paramount to counteract covert attacks [76].

2.4.2 Applied Cyber-Attacks

This study focuses on three attack types: Denial of Service (DoS), Replay, and False Data Injection (FDI) attacks, which are applied to the quadcopter system. These attacks were selected due to their prevalence and significant impact on Cyber-Physical Systems (CPS), particularly in quadcopter applications, where ensuring reliable operation and security is critical. In contrast, Zero Dynamic and Covert Attacks were not considered in this work as the primary focus was on more commonly occurring and impactful attack types. While these excluded attack types are important, their analysis is beyond the scope of this study and has been identified as a potential direction for future research to further enhance CPS security.

The nonlinear quadcopter model can be formulated as:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ \quad y(t) = Cx(t), \end{cases} \tag{2.41}$$

where *x* is the state variable, *u* is the input, *y* is the output and is a *C* constant matrix.

DoS: DoS attacks, when implemented in control systems, disrupt communication by either blocking sensor and actuator data entirely or by replacing missing data with previously received values. In the zero-input strategy, all sensor and actuator data are set to zero, simulating a complete data loss. This approach can be formulated as:

$$\begin{cases}\tilde{y}(t) = 0, \\ \tilde{u}(t) = 0, \end{cases} \tag{2.42}$$

where $\tilde{y}$ is outputs of the system after attack. and $\tilde{u}$ is input of the system after attack.

Alternatively, in the hold-input strategy, the last known values of the sensor or actuator data are retained and fed back into the system. These strategies simulate the effects of DoS attacks on control systems by either completely interrupting communication or causing delayed responses through data retention. This approach can be formulated as:

$$\begin{cases}\tilde{y}(t) = y_{\tau_y}, \\ \tilde{u}(t) = u_{\tau_u}, \end{cases} \tag{2.43}$$

where y_{τ_y} denotes the most recent actuator data before the attack, and u_{τ_u} denotes the most recent sensor data before the attack [77].

In this work the second approach is used to apply DoS on the system.

FDI: In the implementation of False Data Injection (FDI) Attacks, an attacker injects malicious data into sensor measurements to mislead the state estimation process. This can be modeled by modifying the system's sensor outputs as according to following formula:

$$\begin{cases}\tilde{y}(t) = y(t) + y_a, \\ \tilde{u}(t) = u(t) + u_a, \end{cases} \tag{2.44}$$

where y_a represents the injected malicious sensor data and u_a represents the injected malicious actuator data [77].

Replay: In the implementation of replay attacks, an adversary intercepts and records real-time sensor measurements or control actions, and later replays this recorded data to mislead the system. The attack occurs in two stages:

- **Data Collection (Eavesdropping)**: Initially, the attacker collects sensor data without altering it allowing the system to function normally while gathering information. This can be formulated as:

$$\begin{cases}\tilde{y}(t) = y(t), \\ \tilde{u}(t) = u(t), \end{cases} \quad t_0 \le t \le t_0 + w. \tag{2.45}$$

  where t_0 is start of record time and w is record duration.

- **Replay Stage**: The attacker then replays the recorded data to replace real-time sensor measurements and manipulates actuator control signals. This allows the attacker to disguise the attack while preventing the system from detecting anomalies.

$$\begin{cases} \tilde{y}(t) = y(t - t_1 + t_0), \\ \tilde{u}(t) = u(t) + u_a, \end{cases} \quad t_1 \le t \le t_1 + w. \tag{2.46}$$

where t_1 is start of replay time and w is record duration [77].

2.4.3 Cyber-Attacks Implementation

Three different types of attacks are applied on the system which are DoS, replay, False Data Injection (FDI). Specifically, they are all applied on communication networks between quadcopter plant and controller. These attacks can be applied on different sensors or actuators.

All potential attack targets from sensors and actuators are as follows:

Table 2.4: Attack targets of quadcopter.

Number	Type	Target	effects
1	Actuator	T	z
2	Actuator	τ_ϕ	y
3	Actuator	τ_θ	x
4	Actuator	τ_ψ	x, y
5	Sensor	x	x
6	Sensor	y	y
7	Sensor	z	z
8	Sensor	ϕ	y
9	Sensor	θ	x
10	Sensor	ψ	x, y

Cyber-attacks are applied on all 6 sensors and 4 actuators.

The following assumptions and rules govern the application of cyber-attacks on the system:

- Each attack is applied individually on either a single sensor or actuator.
- The severity of each attack is carefully calibrated to ensure that the system maintains stability and does not experience a complete loss of control.
- The duration of each attack can be 1, 2, or 3 seconds.
- The detector is located within the command-and-control unit, receiving only sensor data transmitted through the communication link and actuator data sent from the controller.

The implementation of each attack is described as follows:

- **Denial of Service (DoS)**: This attack is executed using a zero-order hold block in Simulink. The block retains its output value constant, updating only at specific sampling intervals, effectively preventing the transmission of new data.
- **False Data Injection (FDI)**: The FDI attack is implemented by modifying the original signal through the addition of a constant value during the designated attack period, thereby altering the data being communicated.

- **Replay Attacks**: This attack involves the use of a memory block to initially record sensor data. Subsequently, a delay block is employed to replay the recorded sensor data at the specified time during the replay attack. During this process, the FDI attack is concurrently applied to the actuators as described above.

2.5 Conclusion

This chapter presents the comprehensive modeling, control, and simulation of cyber-attacks on a quadcopter system. The quadcopter is modeled based on a full nonlinear dynamic representation, ensuring a detailed and realistic simulation of its behavior. To control the system, a Proportional-Integral-Derivative (PID) controller is implemented. The PID controller is initially designed using the linearized version of the quadcopter's model but is subsequently applied to the nonlinear system to maintain control robustness. Six PID controllers are utilized in three distinct control loops to manage altitude, position, and yaw. The system is then subjected to three different types of cyber-attacks: Denial of Service (DoS), False Data Injection (FDI), and replay attacks. In the DoS attack, data transmission is delayed during the attack period. The FDI attack involves injecting false data into the system's actual values, while the replay attack records sensor data and replays it, concurrently applying the FDI on actuators.

Chapter 3

3. Detection, Identification, and Isolation of Cyber-Attacks in Single Quadcopter

3.1 Introduction

3.1.1 Cyber-Attack Detection, Identification and Isolation

Cyber-attack detection is a critical component of cybersecurity, particularly in the context of critical infrastructures such as smart grids. The ability to detect cyber-attacks in real-time is essential for maintaining the reliability and resilience of these systems. Various methodologies have been proposed for effective detection, including machine learning techniques that can analyze large datasets for anomalies indicative of cyber threats. For instance, a deep and scalable unsupervised machine learning system has been developed to enhance cyber-attack detection capabilities in large-scale environments, emphasizing the need for continuous monitoring to identify targeted attacks promptly [78]. Furthermore, the literature highlights the importance of developing detection mechanisms that can operate effectively even in the presence of sophisticated attack strategies, such as quasi-covert attacks, which are designed to evade detection [79]. This underscores the ongoing challenges in the field, where the sophistication of cyber threats necessitates advanced detection strategies that can adapt to evolving tactics.

Once a cyber-attack has been detected, the next critical step is the identification of the specific type of attack that has occurred. This process involves analyzing the characteristics of the detected anomalies to classify the nature of the attack, which can range from data injection to replay attacks. Various techniques have been proposed for this purpose, including statistical classifiers and machine learning algorithms that can differentiate between different attack vectors based on their signatures [80]. For example, the identification of malicious URLs has been studied extensively, showcasing how spatial analysis can aid in recognizing patterns associated with specific types of cyber threats [80]. Additionally, the identification process is complicated by the emergence of undetectable attacks, such as covert and zero dynamics attacks, which do not produce observable effects on system outputs, thereby complicating the identification process [8]. This highlights the necessity for robust identification frameworks that can accurately classify attacks even when they are designed to remain hidden.

Following the identification of a cyber-attack, the isolation of the affected parts of the system is crucial to mitigate damage and prevent further exploitation. Isolation involves determining which components of the cyber-physical system (CPS) are under threat and implementing measures to contain the attack. Research has shown that effective isolation strategies must account for both malicious cyber activities and machine-induced faults, as both can compromise system integrity [8]. The challenge lies in developing methodologies that can distinguish between the effects of genuine faults and those induced by cyber-attacks, particularly in systems lacking robust security mechanisms [3]. This necessitates the integration of advanced fault detection and isolation techniques that can operate in real-time to ensure that only the compromised components are isolated, thereby maintaining overall system functionality while addressing security concerns. The ongoing development of these methodologies is vital for enhancing the resilience of CPS against a diverse array of cyber threats.

3.1.2 Machine Learning for Cyber-Attack Detection, Identification and Isolation

Machine learning algorithms have emerged as powerful tools in cyber-attack detection, offering the ability to analyze vast amounts of data to identify patterns indicative of malicious activity. These algorithms can be trained on historical data to learn the characteristics of normal and attack traffic, enabling them to detect anomalies in real-time network traffic. Supervised learning algorithms, such as Random Forest and Long Short-Term Memory (LSTM), are commonly used for this purpose, as they can classify network packets into normal and attack categories with high accuracy. Additionally, unsupervised learning algorithms, like k-means clustering and autoencoders, can be employed to detect unknown or novel attacks by identifying patterns that deviate significantly from normal behavior. The integration of machine learning techniques has significantly improved the efficiency and effectiveness of cyber-attack detection, enabling organizations to better protect their networks and systems from cyber threats [81].

Artificial Neural Networks (ANNs) come in various forms, each suited for different types of data and tasks in cyber-attack detection. Recurrent Neural Networks (RNNs), known for their ability to handle sequential data, are effective in analyzing time-series sensor and actuator data to detect anomalies indicative of cyber-attacks. Feedforward Neural Networks (FNNs) are well-suited for pattern recognition tasks and can classify network traffic data into normal and attack categories based on learned features. Convolutional Neural Networks (CNNs) excel in extracting spatial patterns from complex data, making them valuable for detecting attacks that exhibit spatial characteristics in network traffic. The versatility of these different types of ANNs allows for a more comprehensive approach to securing Cyber-Physical Systems (CPS) and other critical systems, as highlighted in recent studies that emphasize the importance of deep learning methods in enhancing cybersecurity measures [82].

Neural networks and deep learning techniques have shown great promise in cyber-attack detection, particularly in their ability to automatically learn and extract features from complex datasets. CNNs have been effectively utilized to analyze spatial patterns in network traffic data, making them well-suited for detecting attacks that exhibit distinct patterns. RNNs, particularly LSTM networks, are effective for analyzing sequential data, such as time-series sensor and actuator data in CPS, enabling them to capture temporal dependencies and detect subtle attack behaviors. Additionally, hybrid models that combine CNNs and RNNs, such as the LSTM-CNN approach, have been proposed to leverage the strengths of both architectures for improved attack detection accuracy. The ongoing evolution of neural networks and

deep learning in cyber-attack detection continues to be a focal point of research, with novel architectures and techniques being explored to enhance detection capabilities [83].

Recurrent Neural Networks (RNNs) have evolved to address various challenges in sequential data analysis, including cyber-attack detection. One popular variant is the Long Short-Term Memory (LSTM) network, designed to mitigate the vanishing gradient problem in traditional RNNs. LSTMs are well-suited for capturing long-term dependencies in time-series data, making them effective in identifying subtle patterns indicative of cyber-attacks. Another variant, the Gated Recurrent Unit (GRU), offers a simplified architecture compared to LSTM while maintaining similar performance, making it a more computationally efficient choice for some applications. Additionally, Bidirectional RNNs combine information from both past and future time steps, allowing for a more comprehensive understanding of temporal patterns in data. These advancements in RNN architectures provide valuable tools for cyber-attack detection in CPS and other critical systems, as they enhance the ability to model complex sequential data [84].

Among various approaches, Long Short-Term Memory (LSTM) networks have emerged as a prominent choice for cyber-attack detection in CPS. LSTM networks are specifically designed to model sequential data with long-range dependencies, making them well-suited for analyzing time-series sensor and actuator data common in CPS. The ability of LSTMs to retain and selectively update information over time enables them to capture complex patterns indicative of cyber-attacks, such as anomalies in system behavior or unexpected sensor readings. Additionally, LSTMs can handle variable-length sequences, allowing them to adapt to different scenarios and effectively model the dynamic nature of cyber threats. Their capability to learn from historical data and identify evolving attack patterns makes LSTM networks a powerful tool for enhancing the security of CPS against cyber threats [85].

3.1.3 Problem Formulation

The aim of this work is the detection, identification, and isolation of cyber-attacks on a quadcopter. Sensor and actuator data from the quadcopter's movement across various trajectories are utilized to develop a model for detecting, identifying, and isolating cyber-attacks.

The first step involves modeling the quadcopter and designing a controller. The quadcopter is then flown along different trajectories while recording data packets from the communication network between the controller and the quadcopter. This data is subsequently used to create the dataset.

A common approach to addressing this type of problem is training a machine learning model on the data, which is then applied to detect, identify, and isolate cyber-attacks. Since the recorded data is in time-series format, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are well-suited for this task.

Two LSTM-based approaches can be employed for cyber-attack detection on a quadcopter:

1) **Signature-based detection:** In this approach, cyber-attacks are introduced during the quadcopter's flight, and data packets are labeled as normal or under attack. An LSTM model is then trained to classify the packets, learning to distinguish between normal and attack data. This trained model is used to classify new data packets.

2) **Anomaly-based detection:** In this approach, the quadcopter moves along its trajectories without any cyber-attacks. The recorded data is used to train an LSTM model to predict the next data packet. If the difference between the predicted and actual packet exceeds a certain threshold, an attack is flagged.

This work adopts the first approach.

LSTM models require sequences of data as input, rather than individual packets. Therefore, data packets need to be reformatted into sequences, and the sequence is labeled based on the labels of its constituent packets. As a result, this problem is framed as a sequence classification task for time-series data.

Key characteristics of this machine learning problem include:

1) **Classification:** The objective is to classify new data packets as either normal or under attack, making this a binary classification problem.

2) **Imbalanced dataset:** The majority of the data consists of normal packets, with only a small proportion representing attacks, leading to an imbalance between the classes.

3) **Time-series data:** The dataset is derived from the recorded communication network data during the quadcopter's flight. As the data follows a temporal sequence, methods that account for this time-series nature are essential for optimal performance.

The problem of cyber-attack identification closely resembles that of detection, except that the output is a label specifying the type of attack: DoS, FDI, or replay, thus making it a multi-class classification task. Cyber-attack isolation, on the other hand, is a more challenging extension of the identification problem, where the number of classification outputs increases from 3 to 10, further complicating the task.

This study introduces an enhanced Long Short-Term Memory (LSTM) network specifically tailored for the unique cybersecurity challenges associated with quadcopters. Although LSTM networks are widely recognized for their ability to model sequential dependencies, this work goes beyond conventional applications by focusing on three interconnected tasks: detection, identification, and isolation of cyber-attacks within a unified multi-output framework. This approach addresses gaps in current methodologies that typically target only detection, often disregarding the critical steps of identifying the attack type and isolating the impacted component, such as sensors or actuators.

The proposed LSTM-based architecture leverages a shared backbone with dedicated output heads for each of the three tasks, enabling a more efficient and cohesive response to cyber threats compared to traditional single-output models. Unlike prior works, which often limit sequence processing to basic detection, this study emphasizes the importance of sequence generation and optimization, treating it as a foundational pre-processing step. By fine-tuning sequence length, overlap, and labeling techniques, the model achieves a deeper understanding of temporal dependencies, critical for robust cyber-attack detection in complex quadcopter systems. This holistic approach not only enhances detection accuracy but also provides precise identification of attack types and isolation of affected areas, thereby setting a new benchmark for comprehensive UAV cybersecurity.

3.2 Methodology

In this section the proposed methodology is explained in detail.

3.2.1 Data Preprocessing

The generated dataset is loaded at this step and used for machine learning model training and test.

Removing the time column in a time-series dataset when using LSTM can be a strategy to prevent the model from learning patterns that are based solely on the timestamp and not on the actual features of the data. Including the time column could lead the model to learn time-related patterns that might not be relevant for the task at hand, potentially introducing bias or "cheating" in the model. By removing the time column, the model is forced to focus on the features that are more directly related to the problem, which can lead to a more accurate and generalizable model. Therefore, in this work time column is removed from features.

StandardScaler is used to standardize features by removing the mean and scaling them to unit variance. This means that each feature will have a mean of 0 and a standard deviation of 1, making the data more suitable for machine learning models that are sensitive to the scale of input features. Standardization helps improve model performance and ensures that no single feature dominates due to differences in scale.

After converting the data into sequences, it is partitioned into training and testing sets, with 75% of the data allocated for training and the remaining 25% reserved for testing. This split ensures that the model has sufficient data for learning while leaving a portion for performance evaluation.

3.2.2 Proposed Sequence Generation Preprocessing

Focusing on appropriate sequence generation preprocessing step is one of the contributions of this work.

There are three main questions to answer about sequence generation:

1) How to split time series data into sequences appropriately?
2) How to label the sequence properly?
3) What is the optimal sequence size?

Each of the questions will be answered using appropriate explanations or experiments.

There are two main approaches to split time series data:

- No overlap: Sequences are entirely distinct, ensuring that there is no shared data between consecutive sequences.
- Full overlap: The full overlap approach allows for the generation of sequences that fully utilize available data by creating overlapping segments.

These approaches are demonstrated in Figure 3.1.

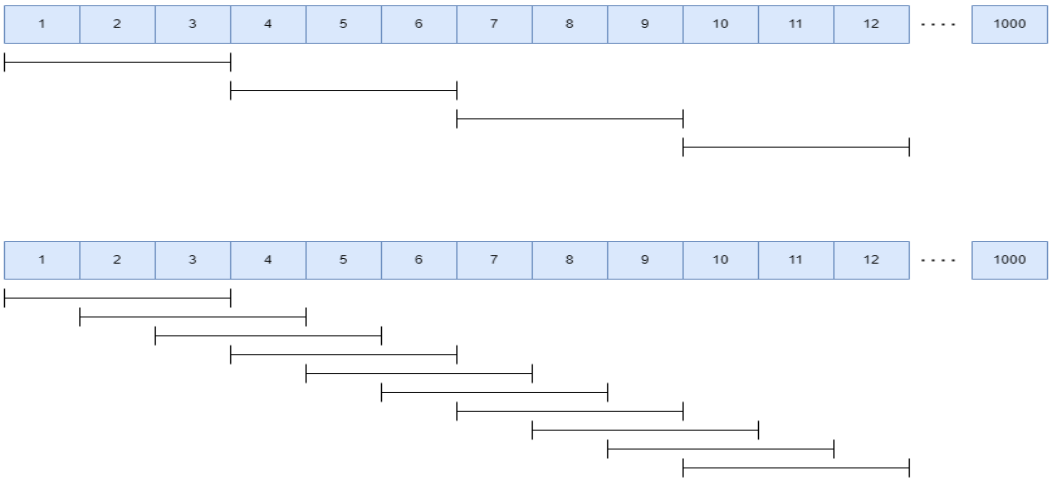


Figure 3.1: Two approaches to split time series data into sequences.

In this work, the second approach, full overlap, is adopted to leverage the richness of the available dataset. By generating overlapping sequences, the model can learn from a more extensive array of temporal patterns and relationships within the data. This is particularly beneficial in the context of LSTM networks, which excel at capturing dependencies across time. The full overlap strategy not only enhances the amount of training data but also facilitates improved model performance by allowing the LSTM to recognize and learn from repeated patterns that may occur in different contexts throughout the dataset.

There are different approaches to label sequences:

1) **Last Packet Label**: The label for a sequence S is assigned based on the label of the last packet in the sequence:

$$Label(S) = Label(P_n), \tag{3.1}$$

where P_n is the label of the last packet in sequence $S = \{P_1, P_2, \ \dots, P_n\}$.

2) **Threshold-based**: The sequence S is labeled as an attack if the number of packets labeled as an attack exceeds a predefined threshold T:

$$Label(S) = \begin{cases} Attack & if \sum_{i=1}^{n} 1[Label(P_i) = Attack] > T, \\ Normal & otherwise, \end{cases} \tag{3.2}$$

where T is the threshold value and 1[·] is an indicator function that equals 1 if the condition is true, and 0 otherwise.

3) **Majority Voting**: The label for a sequence S is determined by the most frequent label among its packets:

$$Label(S) = argmax_l \sum_{i=1}^{n} 1[Label(P_i) = l] \tag{3.3}$$

where l is a possible label (e.g., normal or attack).

4) **Weighted Voting**: The label for sequence S is determined by a weighted sum of the labels of individual packets, with weights W_i considering their position or importance:

$$Label(S) = argmax_l \sum_{i=1}^{n} W_i . 1[Label(P_i) = l] \tag{3.4}$$

where W_i represents the weight assigned to packet P_i.

In this work, the label of the last packet is used as the sequence label. The goal is to utilize information from previous samples, along with the current sample, to classify it as normal or an attack. This approach allows the model to incorporate past information, enabling it to detect changes in the time-series data that may indicate an attack, thereby improving detection performance.

Optimal sequence size will be determined based on experiments in the next section. It can have different values such as 10, 20, 40, 100. Its candidate values should set based on the specific problem and dataset.

Subsequent data packets are grouped to form sequences, with each scenario containing 1000 data packets. These packets are combined to serve as input to the LSTM network based on the specified sequence length. For instance, with a sequence length of 5, the packets are organized as [0, 4], [1, 5], and so on, continuing until [996, 1000]. Finally, the sequences from all scenarios are aggregated to form the complete dataset of data sequences.

3.2.3 LSTM Architecture

In this section, the architecture of the proposed LSTM-based multi output model is outlined, beginning with a detailed explanation of the initial LSTM block. The overall architecture of the model is illustrated in Figure 3.2, providing a high-level overview. A more in-depth discussion of the internal architecture of the LSTM will follow in subsequent sections.

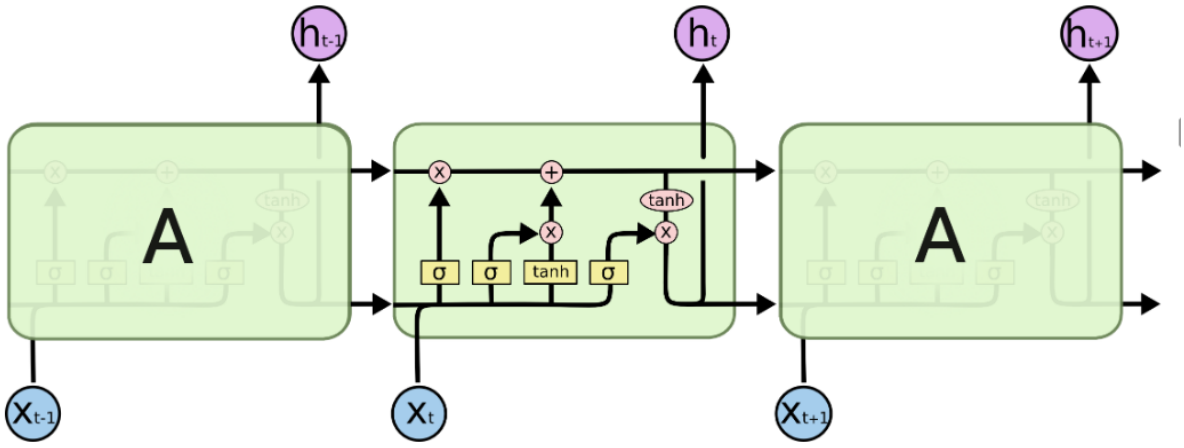


Figure 3.2: LSTM network [86].

The core component of Long Short-Term Memory (LSTM) networks is the cell state, which is represented by the horizontal line running through the top of the network diagram. The cell state propagates throughout the entire sequence with minimal linear modifications, allowing information to be transferred with limited alteration. This architecture enables the LSTM to selectively retain, remove, or incorporate information into the cell state, facilitating the network's ability to manage long-term dependencies in sequential data [86]. Cell state is demonstrated in Figure 3.3.

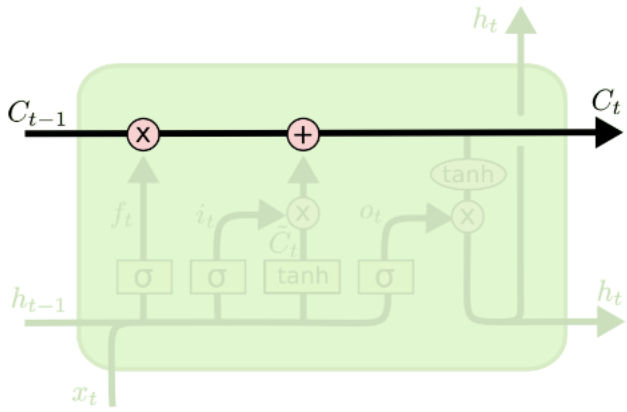


Figure 3.3: LSTM block Cell state [86].

The initial step in an LSTM network involves determining which information should be discarded from the cell state. This decision is governed by a sigmoid activation layer known as the "forget gate." The forget gate evaluates both the previous block's output and the current input, producing a value between 0 and 1 for each element in the cell state. An output value of 1 indicates that the information should be fully retained, while a value of 0 signifies that the information should be entirely discarded. This

mechanism allows the LSTM to regulate the flow of information effectively across time steps [86]. Forget gate is demonstrated in Figure 3.4.

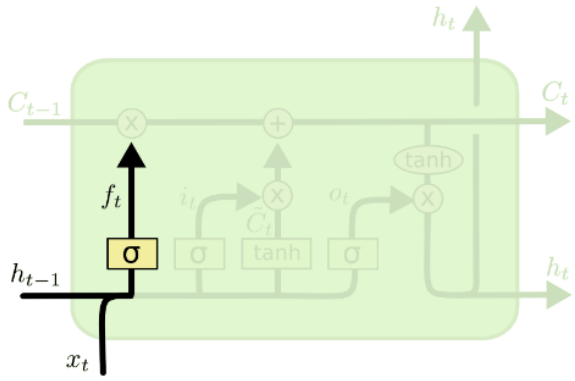


Figure 3.4: LSTM block forget gate [86].

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big), \tag{3.5}$$

The next step in the LSTM process is to determine which new information will be stored in the cell state. This process has two parts. First, a sigmoid activation layer, referred to as the "input gate," selects which values should be updated. Following this, a tanh activation layer generates a vector of potential candidate values, which represent the new information that may be incorporated into the cell state. In the final step, these two outputs are combined to update the cell state, allowing the LSTM to integrate relevant new information while maintaining long-term dependencies [86]. Input gate is demonstrated in Figure 3.5.

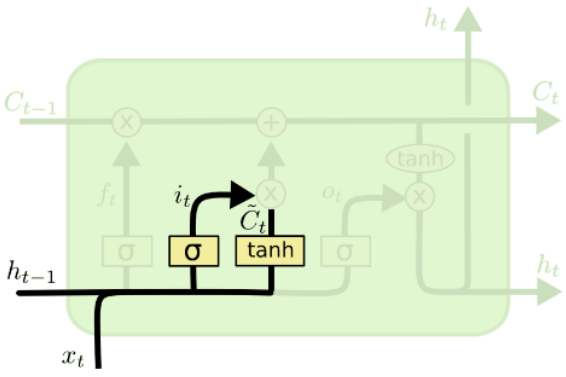


Figure 3.5: LSTM block input gate [86].

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{3.6}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{3.7}$$

At this stage, the previous cell state is updated to form the new cell state, based on the decisions made in earlier steps. The process begins by multiplying the previous cell state by the forget gate output, effectively removing the information deemed unnecessary. Following this, the candidate values generated by the current cell block are added to the state, scaled according to the input gate's decision on how much to update each state value. This results in the formation of the updated cell state, which integrates both preserved and new information [86]. The cell state update process is demonstrated in Figure 3.6.

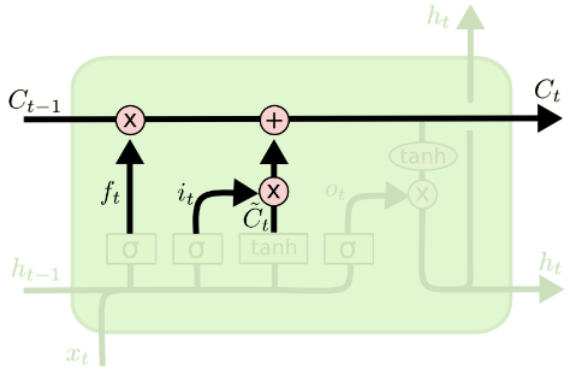


Figure 3.6: LSTM block cell state update [86].

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t, \tag{3.8}$$

The final step involves determining the output, which is a filtered version of the cell state. This is accomplished by first applying a sigmoid layer, which selects the relevant parts of the cell state to output. Subsequently, the cell state is passed through a tanh activation function, which compresses the values to a range between -1 and 1. The result is then multiplied by the output of the sigmoid gate, ensuring that only the selected portions of the cell state are output. This process allows the LSTM to produce a controlled and refined output based on the current cell state [86]. Output gate is demonstrated in Figure 3.7.

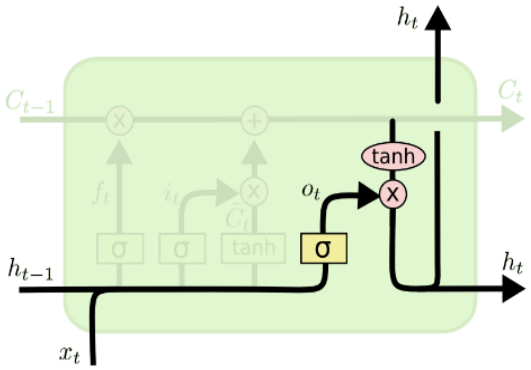


Figure 3.7: LSTM block output gate [86].

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{3.9}$$

$$h_t = o_t \times \tanh(C_t) \tag{3.10}$$

3.2.4 Proposed MO LSTM-based Architecture

This section describes the architecture of the multi-output LSTM model, highlighting its shared backbone and dedicated output heads for detection, identification, and isolation tasks.

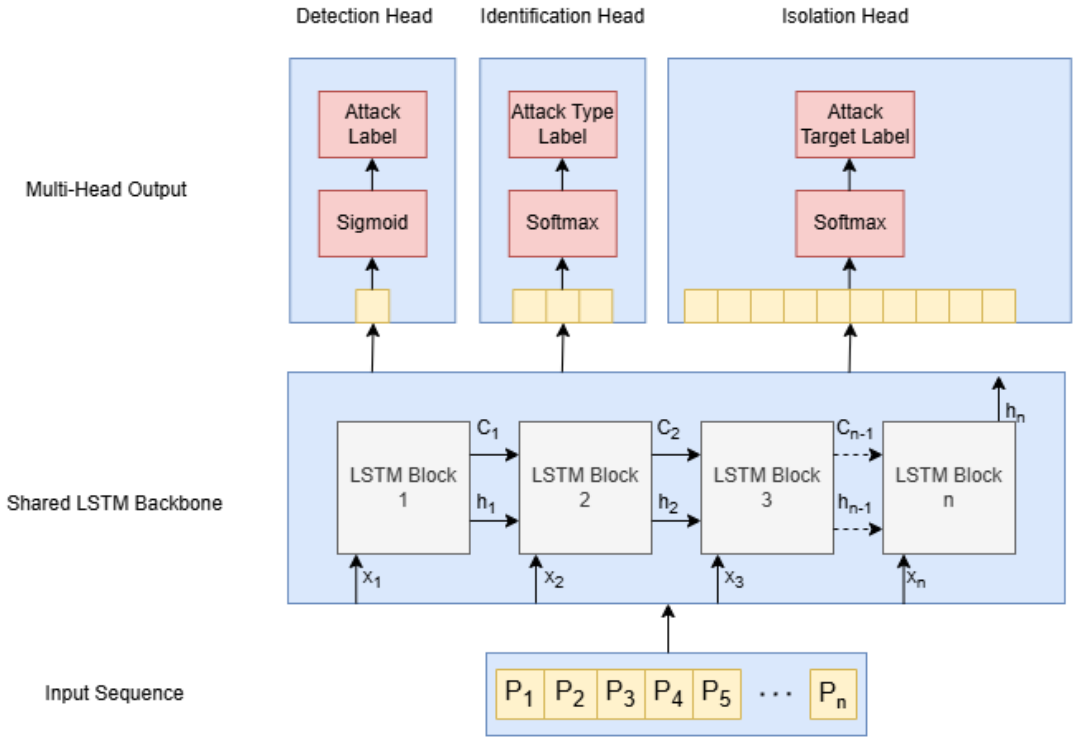


Figure 3.8: Proposed MO LSTM-based architecture.

The data flow in the model is described as follows. The model begins with an input sequence $P = [P_1, P_2, \dots, P_n]$ and this process is identical for all subsequent sequences. The input sequence has a length of n and m features. Each time step of the sequence is fed into the shared LSTM network, with each time step processed by an LSTM block according to its index. Specifically, each time step is fed into the LSTM network as $P_1 = X_1, P_2 = X_2, \dots P_n = X_n$, where X_i represents the input at each time step. The data moves through the LSTM network step-by-step, from left to right, at each time step, as explained in Section 3.2.3 (LSTM Architecture).

At the final time step, the output of the last LSTM block contains information about all previously learned patterns, as the data has flowed through the network using the cell state. The output from the shared LSTM network, represented by h_n is fully connected to each of the three separate heads. Each head is dedicated to a different task, producing values between 0 and 1 for the output neurons, which are used for detection, identification, and isolation.

The architecture presented in Figure 3.8 highlights a novel adaptation of LSTM within a multi-output (MO) framework, structured to handle the tasks of detection, identification, and isolation simultaneously. This design employs a shared LSTM backbone, enabling the system to efficiently process temporal dependencies in time-series data and leverage shared weights across tasks. This integration is

distinct from existing single-task or binary detection models, enhancing both computational efficiency and predictive accuracy in a unified model.

1) **Input Configuration**: The model processes sequential sensor and actuator data, structured into pre-processed input sequences to retain critical temporal patterns. By designing the input in this way, the model captures temporal dependencies that reflect the dynamic behaviors of quadcopters under both normal and attack conditions, crucial for robust detection, identification, and isolation.

2) **Shared LSTM Backbone with Shared Weights**: The backbone consists of LSTM blocks with shared weights that process features from the input data, forming a single vector that serves all three tasks: detection, identification, and isolation. This design is computationally efficient and enables cohesive learning across tasks by generalizing common patterns. Since the problem is formulated to generate a single label or set of labels per sequence, only the output from the last LSTM block is required. Outputs from intermediate blocks are unnecessary, as they are used in hierarchical LSTM structures, which are not applicable here. The backbone efficiently processes time-series data, emphasizing long-term dependencies and sequential patterns critical for multi-task cyber-attack analysis.

3) **Three Distinct Output Heads for Detection, Identification, and Isolation**:

    - **Detection Head**: Positioned as the first output, the detection head is a binary classifier using a single neuron with sigmoid activation, allowing the model to classify sequences as either normal (0) or under attack (1). This approach simplifies the initial decision-making layer, making it quick and effective for attack presence identification.

    - **Identification Head**: The identification head comprises three neurons with softmax activation, classifying the attack type into one of three categories: Denial of Service (DoS), False Data Injection (FDI), or replay attacks. This head is critical for specifying the type of cyber-attack and serves as a secondary analysis.

    - **Isolation Head**: The third output layer contains ten neurons, each linked to a target within the system, with softmax activation to isolate the specific component under attack (e.g., sensor or actuator). By linking neurons to individual targets, the isolation head enhances the precision of threat isolation.

In contrast to prior LSTM-based models referenced in literature, which tend to focus on a single detection task, the proposed architecture integrates multi-task learning, effectively managing multiple outputs from a shared LSTM backbone. This structure supports comprehensive threat assessment in a single model, advancing cybersecurity in drone systems by achieving high accuracy in detection, specific attack identification, and precise target isolation, functionality that is typically achieved only through separate models or with limited output versatility.

3.2.5 Training and Optimization

Loss Function

In this architecture, three distinct task detection, identification, and isolation are handled by specialized output heads that leverage tailored loss functions, each designed to optimize performance for its respective objective.

1) **Detection Output (Binary Classification):** The detection head functions as a binary classifier aimed at distinguishing between normal and attack conditions. Binary Cross-Entropy (Equation 3.11) is used as it aligns with the binary nature of the detection task, where each prediction is either normal (0) or attack (1). The function effectively captures the deviation between predicted and true labels, optimizing the model's ability to discern an attack presence at a high-level classification without additional specifics about type or location.

$$L_{detection} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \tag{3.11}$$

where y_i is the true label (0 or 1) and $\hat{y}_i$ is the predicted probability. i is the index of sample in batch and N indicates the number of samples in each batch [87].

2) **Identification Output (Multi-Class Classification):** For identification, the model aims to classify attack types among multi-class categories such as Denial of Service (DoS), False Data Injection (FDI), and Replay attacks. To address this multi-class classification requirement, Sparse Categorical Cross-Entropy is applied, which is designed to handle integer-encoded labels effectively and to encourage precise attack-type classification through minimized categorical prediction errors. Unlike detection, which requires only binary accuracy, identification demands differentiation across multiple classes.

$$L_{identification} = -\frac{1}{N \times M}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} \cdot \log(\hat{y}_{i,j}), \tag{3.12}$$

where $y_{i,j}$ denotes the true label (0 or 1), and $\hat{y}_{i,j}$ represents the predicted for output neuron j. M refers to the number of output neurons, which corresponds to the number of attack types that is 3. i is the index of sample in batch and N indicates the number of samples in each batch [87].

3) **Isolation Output (Multi-Class Classification):** The isolation objective is to locate the specific component affected by the attack, operating as a multi-class classifier with each output neuron corresponding to a different system component, such as a specific sensor or actuator. Sparse Categorical Cross-Entropy is again employed here to support this multi-class setup, ensuring efficient and accurate isolation by optimizing across categorical probabilities.

$$L_{isolation} = -\frac{1}{N \times P}\sum_{i=1}^{N}\sum_{j=1}^{P} y_{i,j} \cdot \log(\hat{y}_{i,j}), \quad (3.13)$$

where $y_{i,j}$ denotes the true label (0 or 1), and $\hat{y}_{i,j}$ represents the predicted for output neuron j. P refers to the number of output neurons, which corresponds to the number of attack targets that is 10. i is the index of sample in batch and N indicates the number of samples in each batch [87].

Weights Update

The Adam approach is employed for model optimization. This approach calculates derivative of loss function at each step and update weight based on its rules. The total loss function is calculated as follows:

$$L_{total} = \alpha \cdot L_{detection} + \beta \cdot L_{identification} + \gamma \cdot L_{isolation}, \quad (3.14)$$

α, β and γ are weights assigned to each task's loss, controlling the contribution of each task to the overall loss. Currently α, β and γ are set to 1 to contribute equally for model training.

The shared LSTM layer in the model has weights that are used across all three tasks. Therefore, with only one LSTM backbone three different tasks are accomplished.

The gradient for use in Adam optimizer is calculated as follows.

$$\frac{\partial L_{total}}{\partial W} = \alpha \cdot \frac{\partial L_{detection}}{\partial W} + \beta \cdot \frac{\partial L_{identification}}{\partial W} + \gamma . \frac{\partial L_{isolation}}{\partial W}, \quad (3.15)$$

Adam Optimizer

The Adam optimizer is an adaptive optimization algorithm. In Adam, two moving averages are used for each weight's gradient: the first moment (mean) and the second moment (variance), allowing for more efficient updates. Here are the steps of the algorithm [88].

Step 1: Given Gradient

The gradient calculated above will be denoted as g_t and used in the subsequent calculations for the Adam optimizer. This gradient indicates the direction and magnitude of change needed for each weight to reduce the loss.

Step 2: First Moment Estimate (m)

Adam calculates an exponential moving average of the gradients, known as the first moment estimate. This average captures the direction and the speed of the gradient, providing a smoother update over time. The formula for updating the first moment m_t at time step t is given by:

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . g_t, \tag{3.16}$$

where β_1 is a decay rate which is set to 0.9, that controls the impact of previous gradients. This creates a moving average of the gradient values.

Step 3: Second Moment Estimate (v)

The second moment estimate v_t is an exponential moving average of the squared gradients, which helps to regulate updates based on gradient variability. The second moment estimate is calculated as:

$$v_t = \beta_2 . v_{t-1} + (1 - \beta_2) . g_t^2, \tag{3.17}$$

where β_2 is another decay rate which is set to 0.999. This estimate stabilizes the update by adapting to the magnitude of the gradient.

Step 4: Bias Correction for m and v

To correct for initialization bias, especially in early training steps, Adam introduces bias-corrected versions of the first and second moment estimates:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \ , \tag{3.18}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \ , \tag{3.19}$$

Step 5: Weight Update

Finally, each weight w is updated using the corrected moment estimates. The formula for updating each weight is:

$$w_{new} = w_{old} - \alpha . \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \ , \tag{3.20}$$

Here, α is the learning rate, and ε is a small constant (often 10^{-8}) to prevent division by zero. This update rule allows each weight to adaptively adjust based on the historical gradient information and the gradient magnitude.

3.2.6 Implementation

The implementation was carried out using the Python programming language within the Google Colab environment. Python is widely regarded as the most popular language for developing machine learning algorithms due to its versatility and robust libraries. Google Colab, a widely used online platform, offers free access to powerful computational resources such as GPUs and TPUs, making it highly suitable for machine learning tasks. Additionally, the platform simplifies development by minimizing challenges related to library dependencies and delays, thereby facilitating a more efficient implementation process.

Various machine learning libraries were employed for the implementation of the proposed model, each serving specific purposes:

- **Pandas:** Pandas is a robust Python library for data manipulation and analysis, offering versatile data structures and tools for working with structured datasets. In this work, it was utilized for reading the dataset files in CSV format.
- **NumPy:** NumPy is a foundational package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a wide range of high-level mathematical functions. In this study, NumPy was used for reformatting arrays and converting data types, such as transforming Pandas DataFrames into NumPy arrays, which can serve as inputs for other libraries.
- **Scikit-learn:** Scikit-learn is a comprehensive machine learning library that offers tools for classification, regression, clustering, dimensionality reduction, and more, built on NumPy, SciPy, and Matplotlib. In this work, it was used for randomly splitting the data into training and test sets, as well as calculating evaluation metrics such as accuracy, precision, recall, and F1-score.
- **Keras (TensorFlow):** Keras is a high-level neural network API, written in Python, and capable of running on top of TensorFlow, CNTK, or Theano. It is designed to facilitate easy and fast experimentation with deep learning models. In this work, Keras was employed to construct the neural network architecture, compile it with appropriate loss functions and optimizers, and to conduct model training and testing using its built-in tools.

3.3 Dataset Creation

3.3.1 Cyber-Security Datasets

Cybersecurity datasets play a crucial role in the development and evaluation of intrusion detection systems (IDS) and other security mechanisms. These datasets provide researchers and practitioners with the necessary data to train and test machine learning models, enabling the detection of various cyber threats. The effectiveness of these models heavily relies on the quality and diversity of the datasets used, as they must encompass a wide range of attack types and normal behavior patterns to ensure

robust performance in real-world scenarios [89]. Among the most notable cybersecurity datasets are SWaT, CICIDS 2017, and UNSW-NB15, each contributing unique characteristics and challenges for cybersecurity research.

- The SWaT (Secure Water Treatment) dataset is designed to simulate a water treatment facility's operational data, including both normal and attack scenarios. It contains time-series data that reflects the system's behavior under various conditions, making it particularly valuable for testing anomaly detection algorithms in critical infrastructure settings. The dataset includes multiple attack types, such as denial of service and data manipulation, providing a comprehensive environment for evaluating IDS performance [89].
- CICIDS 2017, developed by the Canadian Institute for Cybersecurity, is another significant dataset that encompasses a wide variety of network traffic scenarios. It includes both benign and malicious traffic, with a focus on modern attack vectors. The dataset is notable for its realistic simulation of network environments, capturing a diverse array of attack types, including DDoS, brute force, and web attacks. This variety allows researchers to assess the effectiveness of different detection techniques across multiple attack scenarios [90].
- The UNSW-NB15 dataset, created by the Australian Cyber Security Centre, is distinguished by its comprehensive feature set and the inclusion of nine different attack types. It was generated using the IXIA tool to simulate normal behavior and various modern attacks, resulting in a dataset that includes 49 features. This dataset is particularly valuable for evaluating machine learning models, as it provides a rich source of data for training and testing intrusion detection systems [91]. The UNSW-NB15 dataset has been widely used in research to benchmark the performance of various classification algorithms, making it a cornerstone in the field of cybersecurity research [92].

A review of the literature reveals no widely recognized dataset specifically for UAVs. The datasets commonly referenced are primarily those used in research on cybersecurity for cyber-physical systems, networks, or UAVs. While some studies have generated UAV-related data from real-world implementations, these datasets have not been made publicly available.

In the field of data science and machine learning, datasets can be broadly categorized into real-world and simulated datasets:

- Real-world datasets are derived from actual observations and measurements, reflecting the complexities and variabilities present in real-life scenarios. These datasets are invaluable for training models that need to perform under realistic conditions, as they encapsulate the noise and unpredictability inherent in real-world data [93].
- simulated datasets are generated through computational models and algorithms, allowing researchers to create controlled environments where specific variables can be manipulated. While simulated datasets can provide insights into theoretical scenarios and allow for extensive experimentation without the constraints of real-world data collection, they may not always capture the full complexity of real-world situations [94].

Since the necessary equipment for this study was unavailable, it was not possible to collect a real-world dataset. Instead, simulated datasets offer a practical alternative. Simulation tools, such as MATLAB and Simulink, are widely recognized for their effectiveness in generating these datasets. Their flexibility and

ease of use make them ideal for academic research, including this study, where they enabled the creation of detailed and controlled datasets, supporting reliable research results.

Features of a good dataset include:

- Diversity: A good dataset should encompass a wide range of scenarios and conditions to ensure that models trained on it can generalize well to unseen data.
- Quality: Data should be accurate, consistent, and free from errors or biases that could skew results.
- Size: A sufficiently large dataset is essential to provide enough examples for training, validation, and testing of models, which helps in achieving statistical significance.
- Relevance: The dataset should be relevant to the specific problem domain, containing features and labels that directly pertain to the task at hand.
- Accessibility: Datasets should be easily accessible to researchers and practitioners, ideally with clear documentation on how to use them effectively.
- Balance: A good dataset should have a balanced representation of different classes or categories to prevent model bias towards more prevalent classes [95].

3.3.2 Dataset Design

The dataset for this study was designed with the following key criteria:

- **Diversity**: Multiple scenarios with varying conditions were included.
- **Size**: Enough samples were generated to support the deep learning model.
- **Relevance**: The data relates entirely to quadcopter behavior, with proper labels for normal and attack data.
- **Balance**: Approximately 10-20% of the data consists of attack samples, ensuring a reasonable class balance.

The dataset consists of multiple scenarios representing the quadcopter's movements. In each scenario, the quadcopter follows a trajectory, during which cyber-attacks are applied.
Features of the dataset are demonstrated in Table 3.1.

Table 3.1: Features of dataset.

Feature	Description
Number of samples	50,000 (including both normal and attack data)
Number of scenarios	50 (simulation-based quadcopter movement scenarios)
Number of features	10 (actuator and sensor data)
Data type	Time series
Features type	Numerical (sensor readings and actuator commands)
Class distribution	15% attack data / 85% normal data
Labels	3 labels: detection (binary), identification (multiclass), isolation (multiclass)

Scenarios are generated using the Monte Carlo approach by randomly selecting variables, as defined in Table 3.2, for each scenario.

Table 3.2: Monte Carlo variables.

Variable	Description
Trajectory points	Reference points that define the quadcopter's path during its movement
Attack targets	Set of specific sensors and actuators targeted during the scenario
Number of targets	5-10 (Number of individual attacks applied in the scenario)
DoS start	Start time of the Denial of Service (DoS) attack
DoS duration	1-3 seconds (Duration of DoS attack)
FDI start	Start time of the False Data Injection (FDI) attack
FDI duration	1-3 seconds (Duration of FDI attack)
FDI value	The false value injected into the system during the FDI attack
Record start	Start time of data recording in the simulation
Replay start	Start time of the replay attack
Replay duration	1-3 seconds (Duration of replay attack)

Variables of the simulation for each scenario are defined in Table 3.3.

Table 3.3: Simulation variables.

Variable	Value
Simulation time (seconds)	100
Step value (seconds)	0.1
Number of trajectory points	10
Time interval between setpoints (seconds)	10

The attack column labels are demonstrated in Table 3.4.

Table 3.4: Attack column labels.

Variable	Values
Attack label	0 (Normal), 1 (Attack)
Attack Types	1 (DoS), 2 (FDI), 3 (Replay)
Attack Targets	$x, y, z, \phi, \theta, \psi, T, \tau_\phi, \tau_\theta, \tau_\psi$

Overview of Dataset:

The dataset comprises sensor and actuator readings from a quadcopter system, designed to detect various types of cyber-attacks. Key features include spatial coordinates (x, y, z), orientation angles (ϕ, θ, ψ), and control actuator inputs $(T, \tau_\phi, \tau_\theta, \tau_\psi)$. These measurements capture both the sensros of the quadcopter and the applied control signals over time, allowing for comprehensive monitoring of the system's behavior.

The dataset includes three label columns: label, type, and target. The label column indicates whether the data instance corresponds to normal operation or an attack, supporting the detection objective. The type column specifies the type of the cyber-attack, aiding in identifying which attack is applied. Lastly,

the target column designates the particular target of the attack within the quadcopter system, facilitating precise localization of the compromised component.

3.3.3 Implementation

Data generated from the Simulink simulations were transferred to MATLAB using the Data Inspector tool. This dataset consists of 4 actuator variables and 6 sensor variables. In MATLAB, additional columns, including time, attack labels (label, type, and target), were incorporated. The time variable, also sourced from the Simulink Data Inspector, underwent further refinement within MATLAB. Due to the simulation's time range (0 to 100 seconds with a 0.1 second interval), an extra data point (at time 100) was included, resulting in 1001 data points per scenario. To prevent inconsistencies during the attack detection phase, the last row of each simulation was removed, reducing the dataset to 1000 data points per scenario. Finally, the data was formatted and exported as a CSV file.

3.4 Simulation Results

3.4.1 Evaluation Metrics for Model Performance

In the context of classification models, evaluating performance is crucial to understanding their effectiveness in predicting outcomes. Different metrics provide insights into various aspects of model behavior, particularly how well the model handles both positive and negative cases. Each metric emphasizes a unique facet of performance, ensuring that the model is evaluated from multiple angles. Accuracy measures overall correctness, while precision focuses on the correctness of positive predictions. Recall assesses the model's ability to identify all positive instances, and the F1-score provides a harmonic balance between precision and recall. Additionally, macro and weighted averages offer a way to evaluate performance across multiple classes, considering both class balance and the distribution of instances. A comprehensive evaluation using these metrics allows for a deeper understanding of the model's strengths and weaknesses in different contexts, which is essential for model optimization and deployment.

Definitions of Terms:

- **True Positives (TP)**: The number of instances correctly predicted as positive (i.e., the model correctly identifies a positive class).
- **True Negatives (TN)**: The number of instances correctly predicted as negative (i.e., the model correctly identifies a negative class).
- **False Positives (FP)**: The number of instances incorrectly predicted as positive (i.e., the model incorrectly classifies a negative class as positive).
- **False Negatives (FN)**: The number of instances incorrectly predicted as negative (i.e., the model incorrectly classifies a positive class as negative) [96].

Performance Metrics:

- **Accuracy**: Accuracy measures the overall correctness of the model, representing the proportion of total predictions that are correct.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.21}$$

- **Precision**: Precision quantifies the model's ability to correctly identify positive instances, considering how many predicted positives are actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{3.22}$$

- **Recall**: Recall measures the model's ability to find all actual positive instances in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{3.23}$$

- **F1-Score**: The F1-score is the harmonic mean of precision and recall, providing a balanced measure when precision and recall need to be equally weighted [96].

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.24}$$

Average Metrics:

- **Macro Average**: The macro average computes the metric for each class independently and then averages the results. It treats all classes equally, regardless of the number of instances in each class.

$$Macro\ Avg\ Precision = \frac{1}{N}\sum_{i=1}^{N} Metric_i \tag{3.25}$$

where N is the number of classes and $Metric_i$ is precision, recall or f1-score for class i.

- **Weighted Average**: The weighted average takes the support (the number of true instances) of each class into account, giving more weight to classes with more samples.

$$Weighted\ Avg\ Precision = \frac{\sum_{i=1}^{N}(Metric_i \cdot Support_i)}{\sum_{i=1}^{N} Support_i} \tag{3.26}$$

where $Support_i$ is the number of true instances for class i and $Metric_i$ is precision, recall or f1-score for class i [96]

3.4.2 Model Preprocessing and Configuration

In this section, the primary focus is on optimizing the performance of the model through careful preprocessing and tuning. Three key elements are explored: Normalization and Standardization, Optimizer Selection, and Loss Function Configuration. First, normalization and standardization approaches are applied to ensure the input features are scaled consistently, which is critical for enhancing the convergence speed and stability of the neural network. This involves experimenting with techniques such as Min-Max scaling and Z-score standardization to evaluate their impact on model performance.

Next, multiple optimizers, including popular options like Adam, RMSprop, and Nadam are tested to find the most effective method for minimizing loss during training. Each optimizer is evaluated based on how well it handles gradient descent and improves model learning. Finally, different loss functions, such as binary cross-entropy and binary focal cross-entropy, are considered for binary classification and different loss functions, such as sparse categorical cross-entropy, categorical cross-entropy, and Kullback-Leibler Divergence are considered for categorical classification to ensure the most appropriate metric is used for the classification task. By evaluating these configurations, this section aims to establish the most suitable preprocessing and optimization setup for the subsequent model training stages. Table 3.5 demonstrates other parameters of the model during these experiments.

Table 3.5: Parameters for preprocessing and configuration experiments.

Parameter	Value
Number scenario	20
Sequence length	20
Sequence overlap	Full
LSTM block	64
Epoch	50
Batch	128

1) **Normalization and Standardization:** In this section, three distinct preprocessing techniques were applied to the data to evaluate their impact on model performance. The first experiment utilized raw, unprocessed data, serving as a baseline for comparison. In the second experiment, data

normalization was implemented to scale the features within a specific range, improving comparability across features. The third experiment employed standardization, specifically Z-score transformation, to rescale the data to have a mean of zero and a standard deviation of one. Table 3.6 provides detailed parameters related to these experiments, highlighting the configurations used in this case. Figure 3.9 illustrates the results of experiments.

The results of the three data preprocessing approaches reveal a notable impact on model performance. Using raw data yielded an accuracy of 94%, with a precision of 87.4%, but recall remained low at 60.5%, indicating that the model struggled to detect all positive instances. Normalization, while intended to improve feature scaling, performed poorly in this case, with an accuracy drop to 87.7% and zero values for precision, recall, and F1-score, suggesting a complete failure in classification under this transformation. In contrast, standardization outperformed the other approaches, achieving the highest accuracy of 95.3%, with improved precision 90.2% and recall 69.9%, leading to an F1-score of 78.8%. This suggests that standardization effectively enhanced the model's ability to generalize and balance precision and recall better than raw data and normalization.

Table 3.6: Parameters for normalization and standardization experiments.

Parameter	Value
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Sparse categorical crossentropy
Loss isolation	Sparse categorical crossentropy

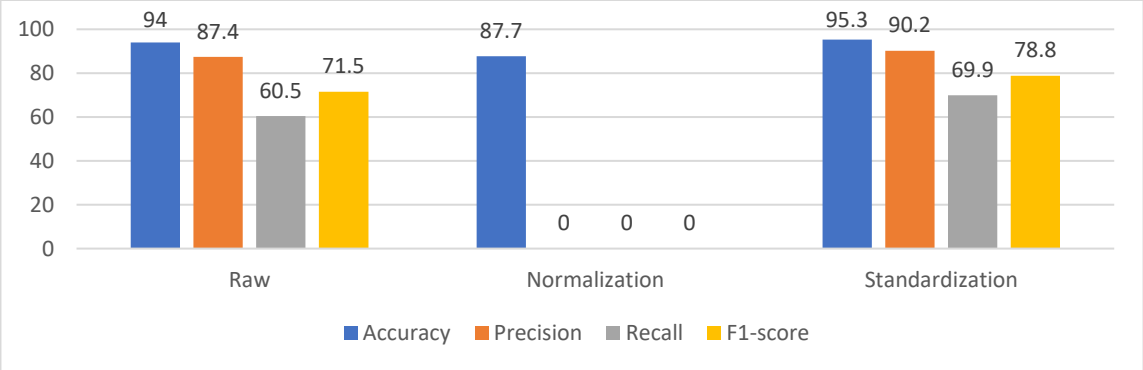


Figure 3.9: Normalization and standardization experiment results.

2) **Optimizer:** In this section, three distinct optimization algorithms were employed to assess their influence on model performance. The first experiment utilized the Adam optimizer, known for its adaptive learning rate and effectiveness in handling sparse gradients, making it a popular choice for various neural network architectures. The second experiment incorporated the RMSprop optimizer, which adjusts the learning rate based on the moving average of squared gradients, thus addressing

the vanishing learning rate issue commonly encountered in deep learning. The final experiment employed Nadam, a combination of Adam and RMSprop that incorporates Nesterov momentum, providing a more responsive and robust optimization strategy. Table 3.7 outlines the detailed parameters associated with these optimization techniques, while Figure 3.10 illustrates the results of these experiments.

The results from the experiments using different optimizers: Adam, RMSprop, and Nadam, indicate strong performance in the context of intrusion detection. Adam achieved the highest detection accuracy at 95.6%, alongside impressive precision 91.5% and F1-score 79.8%, demonstrating its effectiveness in optimizing model training and managing complex gradients. RMSprop followed closely, with a detection accuracy of 95.1% and a notable precision of 86.2%, while maintaining a solid recall of 71.4% and an F1-score of 78%. Nadam also performed competitively with a detection accuracy of 95.5%, precision of 90.5%, and a similar recall to RMSprop at 70.7%, resulting in an F1-score of 79.4%. The results suggest that while all three optimizers yield high accuracy and precision, Adam's performance stands out in terms of overall effectiveness, likely due to its adaptive learning rate mechanism and momentum incorporation, which helps navigate the optimization landscape effectively. These findings underline the importance of selecting appropriate optimization techniques to enhance the performance of machine learning models in cybersecurity applications.

Table 3.7: Parameters for optimizer experiments.

Parameter	Value
Normalization and standardization	Standardization
Loss detection	Binary crossentropy
Loss identification	Sparse categorical crossentropy
Loss isolation	Sparse categorical crossentropy

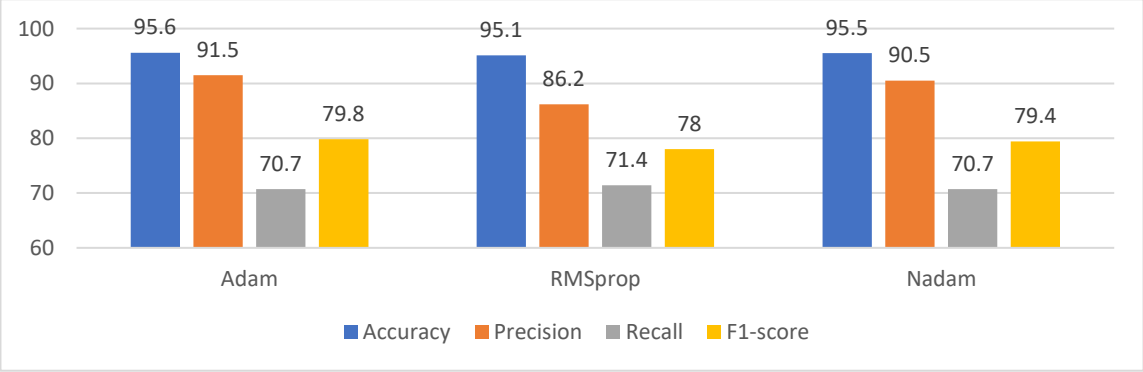


Figure 3.10: Optimizer experiment results.

3) **Loss function:** This section evaluates the influence of various loss functions on model performance across three heads: detection, identification, and isolation. For binary classification tasks associated with detection, both binary cross-entropy and binary focal cross-entropy are utilized. The binary cross-entropy loss effectively measures the disparity between predicted probabilities and actual binary labels, while binary focal loss places greater emphasis on hard-to-classify instances, helping to mitigate class imbalance issues. In the case of the identification and isolation heads, which deal with multi-class classification, sparse categorical cross-entropy and categorical cross-entropy are employed. Sparse categorical cross-entropy is advantageous for handling integer-encoded labels efficiently, whereas categorical cross-entropy is suited for one-hot encoded labels. Additionally, Kullback-Leibler Divergence is utilized to assess the divergence between predicted and true probability distributions, aiding in the model's capacity to distinguish between different categories. Table 3.8 summarizes the parameters associated with these experiments.

Table 3.8: Parameters for loss function experiments.

Parameter	Value
Normalization and standardization	Standardization
Optimizer	Adam

a) **Detection head:** The experimental results comparing binary cross-entropy and binary focal cross-entropy for the detection head are illustrated in Figure 3.11. The comparison of binary cross-entropy and binary focal cross-entropy in the detection head reveals notable performance metrics. The model utilizing binary cross-entropy achieved an accuracy of 95.1%, with precision, recall, and F1-score values of 83.2%, 76%, and 79.4%, respectively. In contrast, the binary focal cross-entropy approach yielded an accuracy of 94.8%, with precision at 84.1%, recall at 71.7%, and an F1-score of 77.4%. These results indicate that while both loss functions produced comparable accuracies, the binary cross-entropy exhibited slightly better overall performance in terms of recall and F1-score. However, binary focal cross-entropy provided improved precision, suggesting its potential for scenarios where false positives are particularly critical. This analysis underscores the importance of selecting the appropriate loss function based on specific model objectives and the trade-offs between precision and recall in detection tasks.

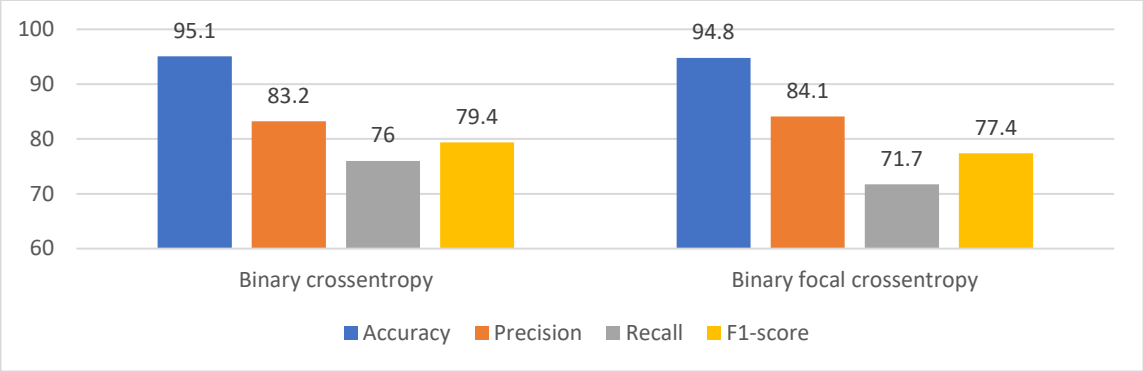


Figure 3.11: Detection head Loss function experiment results.

b) Identification head: The experimental results comparing categorical cross-entropy, Kullback-Leibler Divergence, and sparse categorical cross-entropy for the identification head are illustrated in Figure 3.12. The comparison of loss functions: categorical cross-entropy, Kullback-Leibler Divergence, and sparse categorical cross-entropy, demonstrates comparable performance in the identification head. The identification accuracy was 94.5% for both categorical and sparse categorical cross-entropy, while Kullback-Leibler Divergence achieved a slightly higher accuracy of 94.8%. These findings suggest that while Kullback-Leibler Divergence may have a slight edge in modeling probability distributions, all three loss functions effectively support the identification task. The choice among them can depend on specific model requirements and dataset characteristics, underscoring the importance of selecting an appropriate loss function for optimal classification performance.

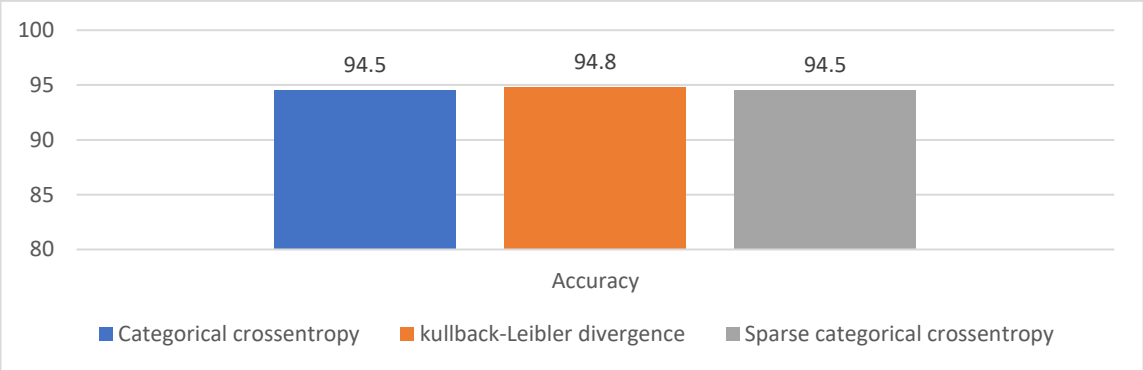


Figure 3.12: Identification head Loss function experiment results.

c) **Isolation head:** The experimental results comparing categorical cross-entropy, Kullback-Leibler Divergence, and sparse categorical cross-entropy for the isolation head are illustrated in Figure 3.13. The evaluation of loss functions for the isolation head revealed varying levels of accuracy among categorical cross-entropy, Kullback-Leibler Divergence, and sparse categorical cross-entropy. Specifically, Kullback-Leibler Divergence yielded the highest accuracy at 94.6%, while categorical cross-entropy and sparse categorical cross-entropy achieved accuracies of 94.0% and 93.7%, respectively. These results indicate that Kullback-Leibler Divergence may provide a slight advantage in effectively capturing probability distributions for isolation tasks, though all three loss functions demonstrated considerable efficacy in classification accuracy.

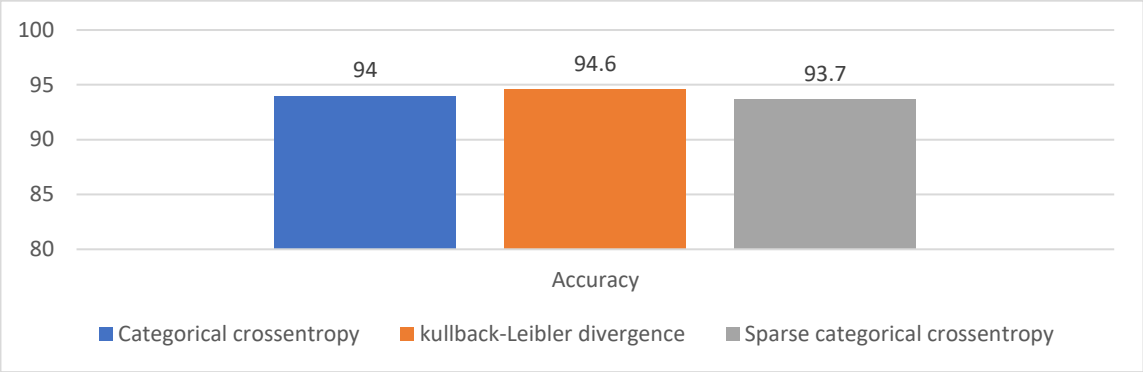


Figure 3.13: Isolation head Loss function experiment results.

Key Findings and Analysis: Based on the results from these experiments, the optimal approach for data preprocessing and model configuration has been identified. Data standardization proves to be the most effective method for adjusting the input data, ensuring consistent scaling and improved model performance. Among optimizers, the Adam optimizer emerged as the best choice, offering superior performance in terms of convergence speed and accuracy compared to other optimization algorithms.

For the binary classification task of cyber-attack detection, the binary cross-entropy loss function provided the most favorable results. This loss function is well-suited for binary classification, effectively distinguishing between normal and attack conditions. In multi-class classification problems, which are attack identification and isolation, the Kullback-Leibler Divergence loss function demonstrated optimal performance. This function is particularly useful when dealing with probabilistic models and multi-class scenarios, leading to more accurate identification and isolation of different types of attacks.

3.4.3 Comparison between Sequence Labeling Approaches

In this section, two sequence labeling approaches are compared within the second set of experiments. The first approach assigns the label of a sequence based on the last packet in the sequence; if the last packet is an attack, the entire sequence is labeled as an attack, otherwise, it is labeled as normal. The second approach introduces a threshold mechanism, where the sequence is labeled as an attack if the number of attack packets within the sequence exceeds a predefined threshold; otherwise, it is labeled as normal. In this experiment, the threshold value for labeling the sequence as an attack is determined based on percentages of the sequence length, with thresholds set at 10%, 25%, and 50%. The sequence length itself varies across different experiments, being set at 10, 20, and 40. This variation allows for a comprehensive evaluation of how different threshold percentages and sequence lengths influence the labeling process and subsequent model performance. Table 3.9 outlines the parameters used in this experiment, and Figure 3.14 presents the experimental results.

Table 3.9: Parameters for sequence labeling experiments.

Parameter	Value
Number scenario	50
Sequence overlap	Full
LSTM block	128
Epoch	100
Batch	128
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Kullback-Leibler Divergence
Loss isolation	Kullback-Leibler Divergence

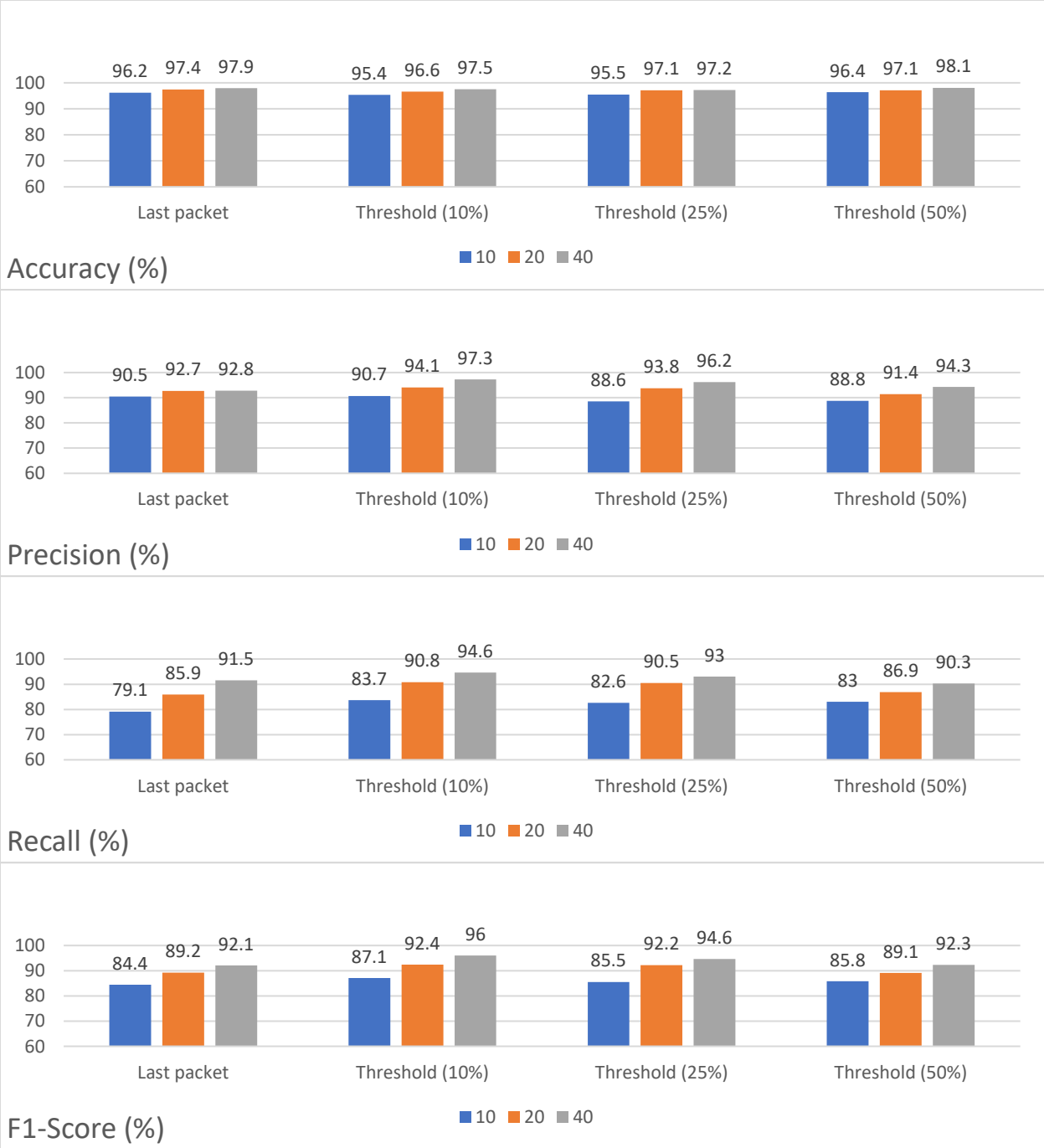


Figure 3.14: Sequence labeling approaches comparison experiments results.

Observations: This section provides a comprehensive analysis of the results from comparing different sequence labeling approaches. The first part examines the impact of altering sequence labeling methods, while the second part explores the effects of varying sequence lengths on model performance.

1) Effect of Changing Sequence Labeling Approaches:

a) **Last Packet:** When using the "Last Packet" approach, the model's decision depends entirely on the label of the last packet within each sequence. This method showed reasonable performance, with accuracy improving as the sequence length increased. At a sequence length of 10, the detection accuracy was 96.2% and the F1-score was 0.844, but with longer sequences, these values rose to 97.9% and 0.921, respectively. While effective, this approach may miss early attack signs and is more sensitive to the final packet in each sequence.

b) **Threshold (10%):** The "Threshold (10%)" method performed better than the "Last Packet" approach across all sequence lengths, as it considers multiple packets in a sequence. For a sequence length of 10, it achieved an F1-score of 0.871, which increased to 0.960 at a length of 40. This approach enhances the model's sensitivity by labeling sequences based on 10% of packets being classified as attacks, significantly improving recall without sacrificing precision.

c) **Threshold (25%):** The "Threshold (25%)" approach also outperformed the "Last Packet" method but lagged slightly behind the 10% threshold. For example, at a sequence length of 40, the accuracy was 97.2% and the F1-score reached 0.946. This method strikes a balance, considering a higher percentage of attack packets within the sequence, which tends to result in more conservative labeling, thus slightly lowering recall compared to the 10% threshold.

d) **Threshold (50%):** With the "Threshold (50%)" approach, the model showed a similar trend, though performance metrics such as recall slightly decreased. At a sequence length of 40, accuracy was the highest at 98.1%, and the F1-score remained strong at 0.923. This approach increases precision by requiring a higher threshold of attack packets, though it may occasionally overlook attacks that occur earlier in a sequence.

2) **Effect of Changing Sequence Length:**

a) **Sequence length 10**: For a sequence length of 10, the last packet labeling approach yielded an accuracy of 96.2%, a precision of 90.5%, a recall of 79.1%, and an F1-score of 84.4%. With the threshold approach, results improved incrementally as the threshold increased. At 10%, the threshold achieved a 95.4% accuracy and an F1-score of 87.1%, while the 25% and 50% thresholds resulted in similar levels of performance, with accuracy at 95.5% and 96.4%, respectively. The precision, recall, and F1-scores showed a stable increase, indicating the potential benefit of higher thresholds for shorter sequence lengths.

b) **Sequence length 20**: Increasing the sequence length to 20 packets, the detection performance improved across all approaches. The last packet approach showed a higher accuracy of 97.4%, along with improved precision and recall (92.7% and 85.9%, respectively), ending in an F1-score of 89.2%. The threshold-based labeling approaches performed better, particularly at 10%, where accuracy increased to 96.6%, and the F1-score reached 92.4%. The 25% and 50% thresholds maintained competitive performance, with accuracies of 97.1% and 97.1%, and F1-scores surpassing 92%.

c) **Sequence length 40:** At a sequence length of 40 packets, the models demonstrated the highest detection accuracy across the board. The last packet method reached 97.9% accuracy with an F1-score of 92.1%. For the threshold-based labeling, the accuracy peaked at 98.1% for the 50% threshold, which also resulted in the highest F1-score of 92.3%. Similarly, the 10% and 25% thresholds performed exceptionally well with accuracies of 97.5% and 97.2%, respectively, indicating that longer sequence lengths combined with higher thresholds can enhance detection accuracy and precision.

Key Findings and Analysis: Based on the experimental results, the threshold-based labeling approach demonstrates overall better performance when compared to the last-packet labeling approach. However, it is important to highlight that the threshold approach differs significantly from similar works in the literature, as it addresses the labeling problem in a fundamentally different way. While the last-packet labeling method leverages the temporal information from the entire sequence to determine the final label, aligning with more conventional methods the threshold approach operates by assessing the number of attack packets within a sequence, thus tackling a different classification challenge. Although the threshold method yields superior results in this context, it is not an ideal approach for the rest of the experiments due to its distinct methodology, which does not align with the typical problem framework in related studies. Therefore, it is used here primarily to illustrate that when sequences are input into an LSTM model, multiple labeling strategies can be considered, each with its own implications for model behavior and performance.

3.4.4 Comparison between Different Sequence Lengths

In this section, in the third set of experiments the focus is on comparing different configurations of LSTM blocks (64, 128, and 256) in relation to varying sequence lengths (20, 40, and 100). The objective is to evaluate how the number of LSTM units impacts model performance when applied to sequences of different lengths. The LSTM block size directly influences the model's capacity to capture temporal dependencies within the data, while the sequence length affects the amount of temporal information available for learning. The combination of these factors provides a comprehensive understanding of the trade-offs between model complexity and input sequence granularity. Table 3.10 lists the parameters employed in this experiment, and Figure 3.15 illustrates the experimental outcomes. This analysis highlights the relationship between LSTM block size and sequence length in optimizing model performance for time series data. In Figure 3.16, the training times for these models are compared to illustrate the computational cost incurred with increasing model complexity.

Table 3.10: Parameters for sequence lengths experiments.

Parameter	Value
Num scenario	50
Sequence overlap	Full
Epoch	100
Batch	128
Sequence labeling	Last packet
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Kullback-Leibler Divergence
Loss isolation	Kullback-Leibler Divergence

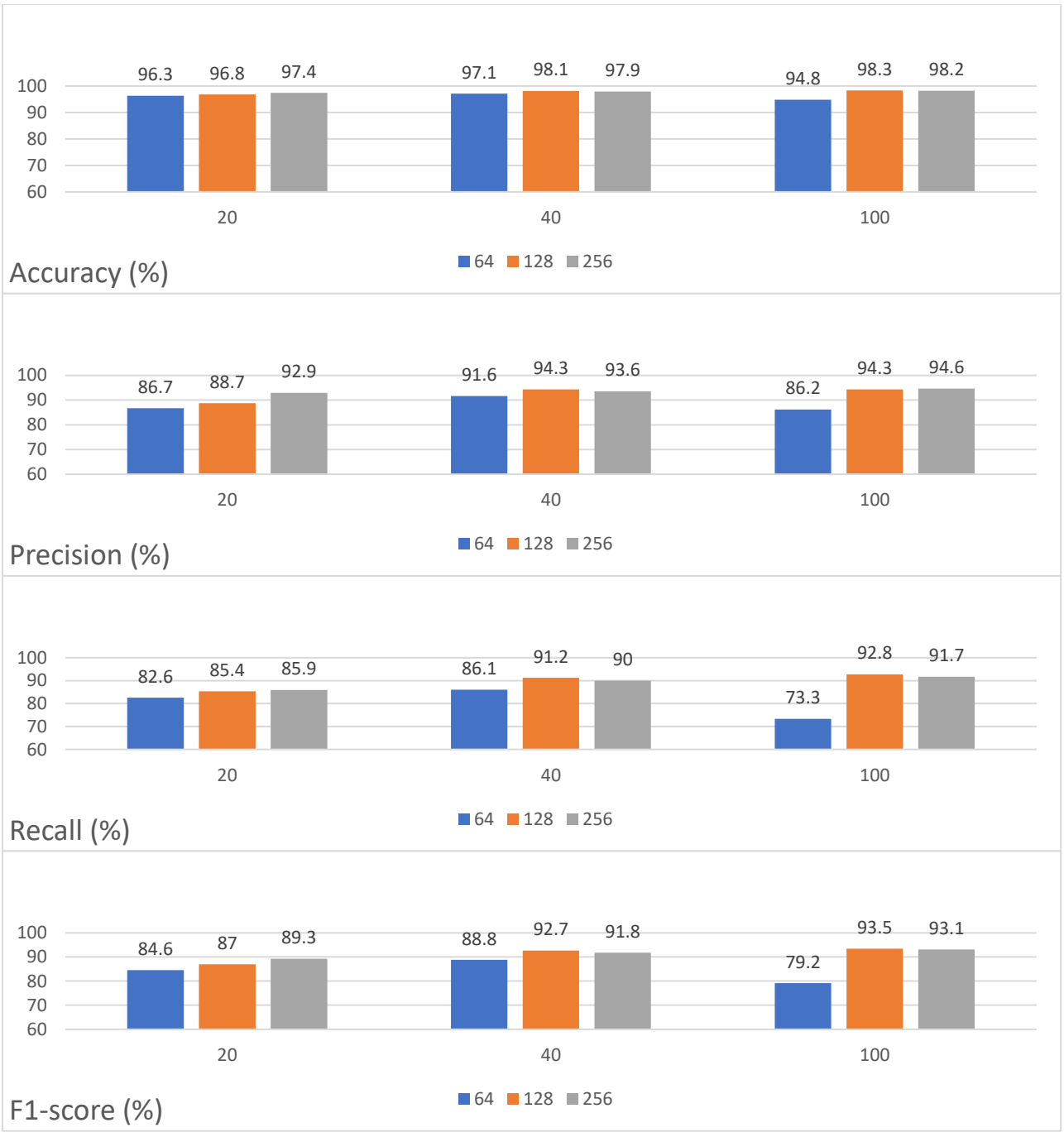


Figure 3.15: Sequence length and LSTM blocks experiments results.

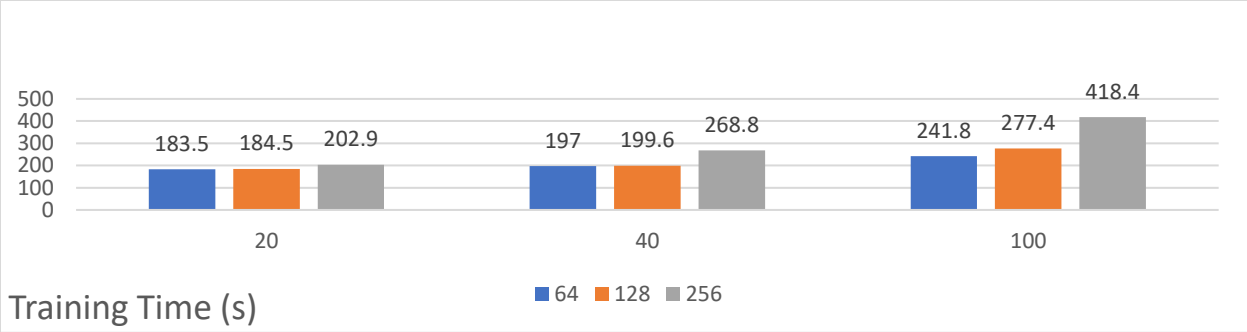


Figure 3.16: Sequence length and LSTM blocks experiments training time.

Observations: This section provides a comprehensive analysis of the results from comparing different sequence length and LSTM blocks. The first part examines the impact of altering sequence lengths, while the second part explores the effects of varying LSTM blocks on model performance and training time.

1) Effect of Changing Sequence Length:

a) Sequence length 20:

- **Performance**: With sequence length 20, the performance improves as LSTM block size increases. For 64 blocks, the detection accuracy is 0.963, with an F1-score of 0.846. The performance increases with 128 blocks (accuracy of 0.968, F1-score of 0.870) and peaks with 256 blocks, achieving 0.974 accuracy and an F1-score of 0.893. The recall and precision metrics follow a similar trend.
- **Training time:** As expected, training time increases with the number of LSTM blocks. For 64 blocks, training time is 183.5s, while for 128 blocks, it rises slightly to 184.5s. The training time increases more significantly with 256 blocks, reaching 202.9s.

b) Sequence length 40:

- **Performance**: Sequence length 40 yields higher performance across the board. For 64 blocks, detection accuracy is 0.971 with an F1-score of 0.888. The performance with 128 blocks is even better, reaching 0.981 accuracy and an F1-score of 0.927. Performance peaks with 256 blocks, providing accuracy of 0.979 and an F1-score of 0.918.
- **Training time:** The training time continues to rise with increased LSTM blocks and sequence length. The model with 64 blocks takes 197.0s, and 128 blocks increase this time to 199.6s. For 256 blocks, training time grows to 268.8s, reflecting the added complexity.

c) Sequence length 100:

- **Performance**: At sequence length 100, the model performance plateaus. Both 128 and 256 blocks achieve very similar detection accuracy (0.983 and 0.982, respectively), with F1-scores of 0.935 and 0.931. However, despite good performance, the gains from increasing LSTM blocks become marginal compared to the shorter sequence lengths.

- **Training Time**: Training time increases considerably for sequence length 100. For 64 blocks, it is 241.8s, while for 128 blocks, it reaches 277.4s. The highest training time, 418.4s, occurs with 256 blocks, showing the computational cost of handling longer sequences with a more complex model.

2) Effect of Changing Number of LSTM Blocks:

a) 64 Blocks:

- **Performance**: The model with 64 blocks performs adequately, with the best performance achieved at sequence length 40 (accuracy of 0.971, F1-score of 0.888). However, performance decreases at sequence length 100 (accuracy of 0.948, F1-score of 0.792), indicating that 64 blocks struggle with longer sequences.
- **Training Time**: Training time for 64 blocks ranges from 183.5s (sequence length 20) to 241.8s (sequence length 100), highlighting a moderate increase in computation time as sequence length increases.

b) 128 Blocks:

- **Performance**: The 128-block model shows improved performance across all sequence lengths. For sequence length 100, it achieves its highest accuracy (0.983) and F1-score (0.935). This block size offers a strong balance of performance across various sequence lengths.
- **Training Time**: Training time for 128 blocks increases with sequence length, ranging from 184.5s for sequence length 20 to 277.4s for sequence length 100. Although the performance gains are significant, the computational cost rises accordingly.

c) 256 Blocks:

- **Performance**: The 256-block model consistently delivers high performance, with its best F1-score (0.931) achieved for sequence length 100 and strong performance across the board. Despite these gains, the model's advantages over 128 blocks become less pronounced at longer sequence lengths.
- **Training Time**: Training time grows steeply with 256 blocks, particularly for sequence length 100, where it reaches 418.4s. This highlights the computational burden of using a more complex model, particularly when handling longer sequences.

Key Findings and Analysis: After analyzing the performance metrics from the latest experiments, the model with 128 LSTM blocks and a sequence length of 40 stands out as the optimal configuration. This model achieves a detection accuracy of 0.981, precision of 0.943, recall of 0.912, and an F1-score of 0.927, striking a solid balance between high performance and computational efficiency.

The training time for this setup is 199.6 seconds, which is reasonable given its consistently high detection capabilities. Although configurations with longer sequence lengths (e.g., 100) or more LSTM blocks (e.g.,

256) offer marginal improvements in performance, the corresponding increase in computational cost does not justify these gains. For instance, the model with 256 blocks and a sequence length of 100 achieves a slightly higher accuracy of 0.982, but its training time jumps to 418.4 seconds more than double that of the optimal configuration, with only minimal performance improvement.

In contrast, the 128-block model with a sequence length of 40 provides a significant step up from smaller models, such as those with 64 blocks, in both performance and efficiency. This setup delivers an ideal trade-off between high precision, recall, and F1-score while keeping training time relatively low, making it the optimal choice for this cyber-attack detection problem.

3.4.5 Model Performance Analysis and Comparison with Existing Approaches

The fourth set of experiments focuses on evaluating the final model configured with optimal parameters and comparing its performance to similar works in the existing literature, thereby demonstrating its superiority. This section details the results achieved by the proposed model in the detection, identification, and isolation of cyber-attacks. While the detection performance will be compared to other existing studies in the following section, it is noteworthy that the identification and isolation results are unique to this research, as it represents the first work addressing these specific aspects. The comparative analysis reveals that the proposed detection method outperforms two analogous studies in literature, confirming its superiority in this domain. This section also includes an experiment to measure the real-time detection capability of the model. The detection time will be evaluated to assess the model's efficiency and suitability for real-time applications. Table 3.11 provides an overview of the model parameters utilized in these experiments. Additionally, the results for detection, identification, and isolation are presented in a heatmap format, as depicted in Figures 3.17, 3.18, and 3.19, respectively.

Table 3.11: Parameters for comparative analysis experiments.

Parameter	Value
Num scenario	50
Sequence overlap	Full
Epoch	100
Batch	128
Sequence labeling	Last packet
Sequence size	40
LSTM blocks	128
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Kullback-Leibler Divergence
Loss isolation	Kullback-Leibler Divergence

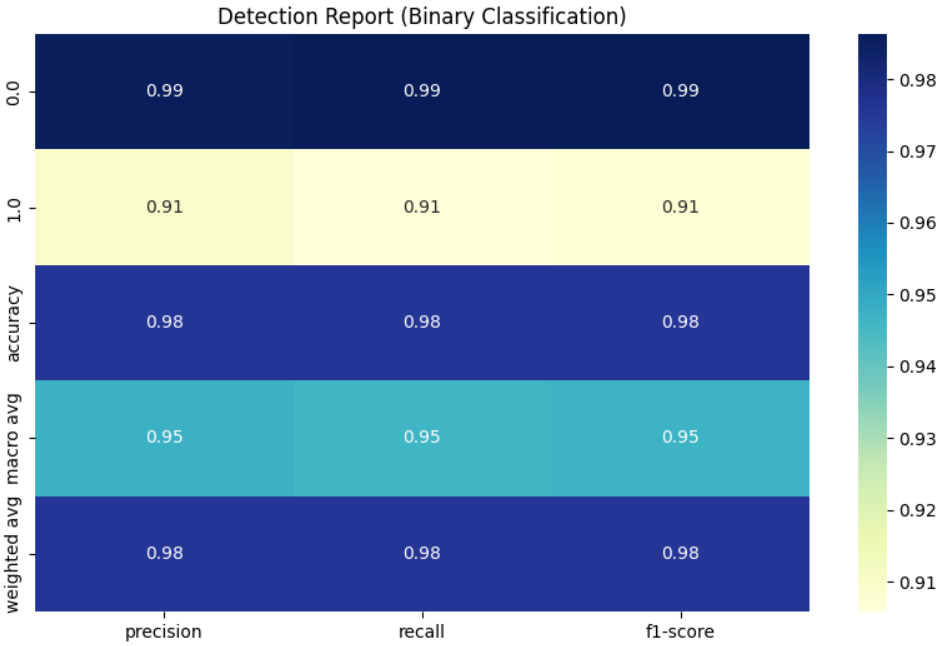


Figure 3.17: Proposed model detection results.

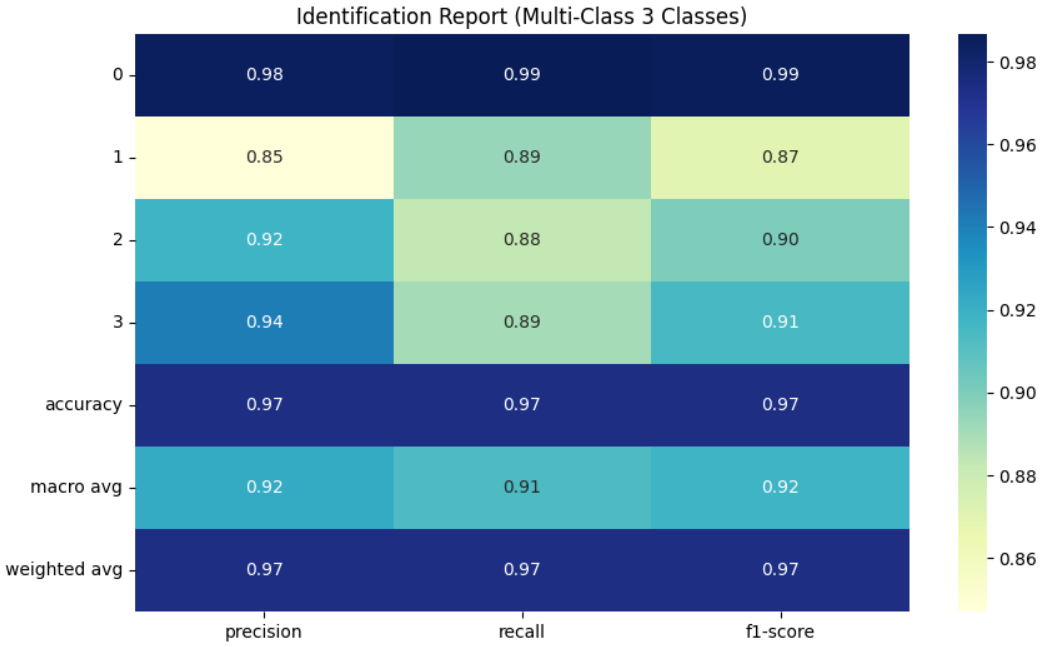


Figure 3.18: Proposed model identification results.

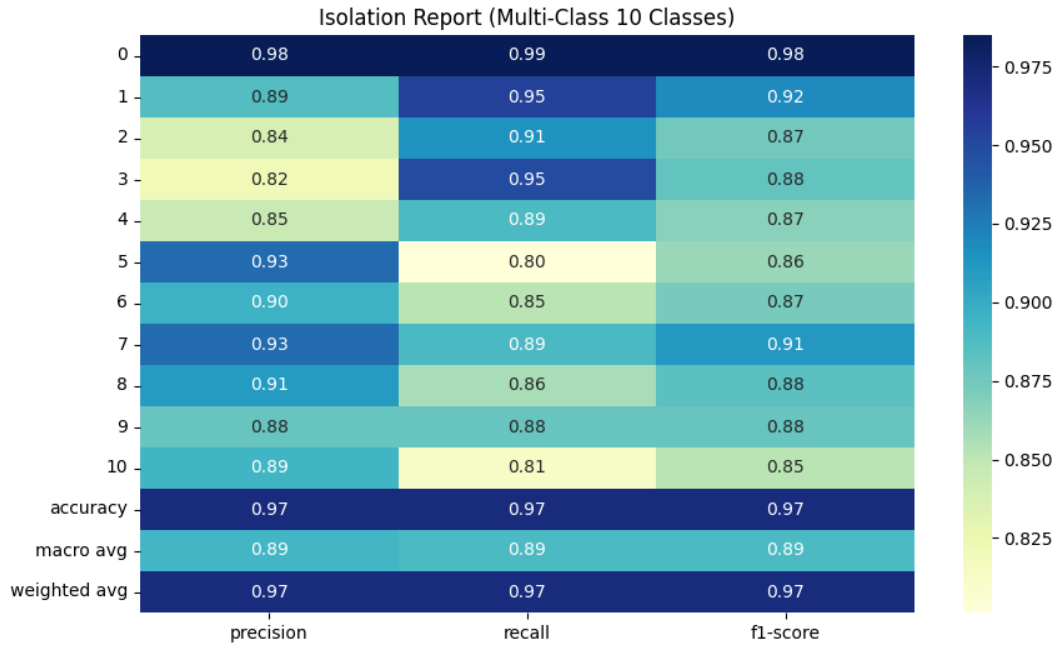


Figure 3.19: Proposed model isolation results.

Observations:

1) **Detection**: The heatmap showcases the detection results of the proposed model, revealing a commendable performance in classifying cyber-attacks. Specifically, the model achieved a precision of 0.99 for class 0 (normal traffic) and 0.91 for class 1 (attack), indicating a high accuracy in correctly identifying normal data while maintaining a strong capability in detecting attacks. The recall rates of 0.98 for class 0 and 0.91 for class 1 further highlight the model's effectiveness, suggesting that it successfully identifies a significant portion of actual attacks. With an overall accuracy of 0.98, the proposed model demonstrates a robust performance in distinguishing between normal and attack classes, affirming its potential for practical applications in cyber-attack detection.

2) **Identification**: The heatmap showcases the exceptional performance of the proposed cyber-attack identification model. Specifically, the model achieved precision scores of 0.99 for normal traffic, 0.91 for DoS attacks, 0.92 for FDI attacks, and 0.94 for replay attacks, respectively. These results demonstrate the model's ability to accurately identify each type of attack, highlighting its effectiveness in detecting various cyber threats. Additionally, the recall rates for each class were consistently high, indicating that the model successfully identifies a significant portion of actual attacks. The recall scores were 0.98 for normal traffic, 0.91 for DoS attacks, 0.92 for FDI attacks, and 0.94 for replay attacks. Finally, the F1-scores, which combine precision and recall into a single metric,

further reinforce the model's overall performance. The F1-scores for each class were 0.98 for normal traffic, 0.91 for DoS attacks, 0.92 for FDI attacks, and 0.94 for replay attacks.

The comparative analysis of attack identification reveals notable variations in precision across different types of attacks. Specifically, the precision for Denial of Service (DoS) attacks is significantly lower than that of other attack types, while the precision for Replay attacks is marginally higher compared to False Data Injection (FDI) attacks. In terms of recall, all attack types exhibit similar performance levels, indicating that the model maintains consistent detection capabilities across the board. Lastly, the F1 scores highlight that the model performs better in identifying Replay and FDI attacks compared to DoS attacks.

3) **Isolation**: The results of the proposed model for the isolation of cyber-attacks demonstrate impressive performance across all metrics, showcasing its effectiveness in accurately identifying various attack classes. The model excels in precision for the "No Attack" class, achieving a remarkable score of 0.98, which highlights its capability to minimize false positives effectively. Other classes also show strong results, particularly Class 4 and Class 7, with F1-scores of 0.94 and 0.91, respectively, indicating that the model can reliably identify these specific attacks. Recall scores are generally robust, suggesting the model effectively captures true positives across most attack types, with all classes performing admirably around the 0.85 mark. The overall accuracy of 0.89 signifies the model's reliability and its contribution to the field of cyber-attack detection and isolation.

   The results for the isolation of cyber-attacks indicate that the overall F1-scores of the models are comparable, averaging around 0.87 across most targets. However, two specific targets, namely 1 and 7, achieved higher F1-scores exceeding 0.90, demonstrating their superior identification capabilities. Analyzing precision and recall reveals a nuanced performance; many targets exhibit a trade-off where models excel in either precision or recall but not both. For instance, targets 5, 6, 8, and 10 show better precision than recall, indicating a propensity to minimize false positives, while targets 2, 3, and 4 achieve higher recall than precision, suggesting a focus on capturing true positives. Notably, targets 1, 7, and 9 excel in both precision and recall, highlighting the effectiveness of the proposed model in accurately identifying these specific attack targets. Overall, these findings underscore the robustness of the model in addressing the challenges of cyber-attack isolation and emphasize its contribution to advancing the understanding of performance metrics in this domain.

Comparative analysis of detection performance: This section presents a comparative analysis of the detection performance of the proposed model against two similar works that employed preprocessing techniques alongside LSTM. The objective is to evaluate the efficacy of the proposed model in relation to these established approaches. By implementing a specialized preprocessing strategy, the proposed LSTM-based approach demonstrates superior detection capabilities compared to the referenced literature. It is important to note that the results pertaining to identification and isolation cannot be directly compared to existing works, as no studies have focused on these specific tasks in the current literature. This emphasizes the novelty of the proposed model in addressing these aspects of cyber-attack detection. Figure 3.20 illustrates the results of comparative analysis experiments.

The comparative results presented in the Table demonstrate the significant superiority of the proposed model over two existing works in literature. The proposed model achieved an impressive accuracy of 98%, with a precision, recall, and F1-score of 91%, indicating a robust capability for effective cyber-attack detection. In contrast, the first model reported an accuracy of 88.5%, with a lower precision of 54.2%, and an F1-score of 64.9%, suggesting limitations in its ability to identify true positive instances. The second model showed improved performance, achieving an accuracy of 94% and an F1-score of 79%, yet still fell short of the proposed model's metrics. These results not only affirm the efficacy of the proposed approach but also highlight its enhanced ability to balance precision and recall effectively. This advancement is crucial in practical applications where accurate detection of cyber threats is paramount, reinforcing the value of the proposed methodology in the field of cyber-attack detection.

The results obtained from this section's experiments indicate that the proposed model demonstrates a high level of performance, achieving metrics around 90% or above across all tasks, including detection, identification, and isolation. This consistent level of excellence in performance metrics signifies that the proposed model not only excels in its specific tasks but also surpasses existing methods in the literature, affirming its superiority in cyber-attack detection capabilities. Thus, the proposed model represents a valuable contribution to the advancement of knowledge in this field, highlighting its potential to enhance existing methodologies and provide more effective solutions for cybersecurity challenges.

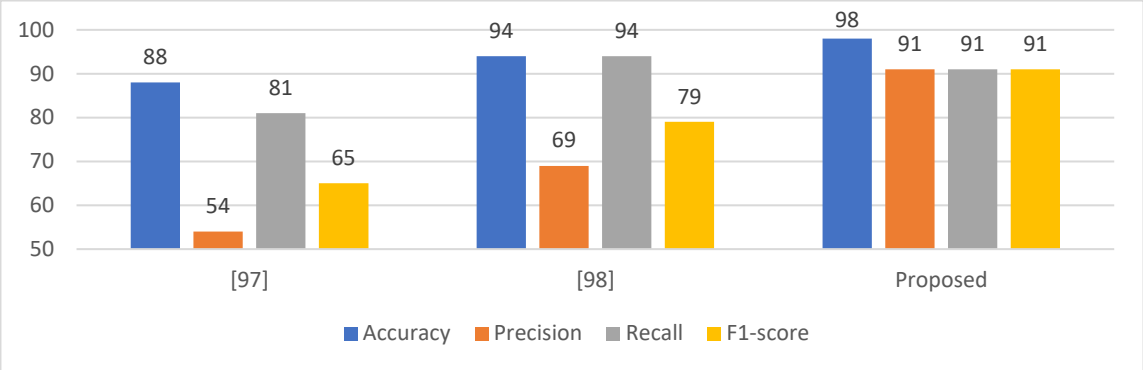


Figure 3.20: Comparative analysis experiments results.

Detection Time: To evaluate the feasibility of the proposed model for real-time applications, an experiment was conducted to measure its detection time. Detection time refers to the total time required by the model to process an input sequence and perform the tasks of detection, identification, and isolation. The real-time detection capability of the model was assessed by measuring the time required to generate predictions for 50 randomly selected test samples. Each prediction was performed on a CPU, as this closely approximates the computational limitations of embedded processors used in quadcopters. The detection time was measured for each sample individually, and the results were averaged to obtain a reliable estimate of the model's performance in a real-time setting. The measured average detection time was 0.053 seconds (53 ms), demonstrating the computational efficiency of the model and its suitability for deployment in real-time scenarios of single quadcopter.

3.5 Conclusion and Contributions

This chapter addressed the critical tasks of cyber-attack detection, identification, and isolation within a single quadcopter system, framing it as a machine learning problem. The methodology introduced here leveraged Long Short-Term Memory (LSTM) networks to handle the temporal nature of sensor and actuator data. A primary contribution of this work is the integration of a shared LSTM backbone within a multi-output framework, which allows simultaneous handling of detection, identification, and isolation tasks. This approach contrasts with traditional models that typically address only one of these tasks, enhancing both computational efficiency and predictive accuracy.

By incorporating a novel preprocessing step focused on sequence generation and labeling, the proposed model was able to outperform existing approaches in the literature, particularly in the detection task. The ability to generate sequences from the quadcopter's time-series data allowed the model to capture long-term dependencies, a key factor in detecting subtle attack behaviors that might otherwise go unnoticed.

In addition to advancing detection techniques, this work is the first to introduce identification and isolation tasks for cyber-attacks in the context of quadcopters using machine learning. The identification head of the model accurately classified attack types, including Denial of Service (DoS), False Data Injection (FDI), and replay attacks, while the isolation head effectively identified the specific components (sensors or actuators) under attack. This multi-task capability is significant, as it demonstrates the model's potential to not only detect an attack but also provide valuable information for mitigating its effects.

Overall, the results presented in this chapter show that the proposed LSTM-based model, with its shared-weight architecture and multiple output heads, offers a robust solution for cyber-attack detection, identification, and isolation. By achieving high performance across these tasks, this work sets the stage for future research aimed at improving the security of autonomous systems. The findings underscore the importance of integrating machine learning techniques for enhancing the resilience of cyber-physical systems against a wide range of cyber-attacks.

Chapter 4

4. Detection, Identification, and Isolation of Cyber-Attacks on Centralized and Decentralized Network of Quadcopters

4.1 Introduction

4.1.1 Problem Formulation

The field of cyber-attack detection in networks of quadcopters using machine learning is currently underexplored. To the best of the author's knowledge, a few prior works have fully addressed the unique aspects of networked quadcopter systems, including clear topology design and data collection specific to such networks. Most existing studies rely on general datasets like UNSW-NB15 or CICIDS-2017, which are not specifically designed for quadcopters and only approximate single-drone attack scenarios. For studies generating their own data, the focus remains on single quadcopters rather than networked systems. Although some studies suggest expanding to networked quadcopters as a potential future direction, none have pursued this yet. Moreover, while many works reference cyber-attack detection in the context of an "Internet of Drones," they often utilize general-purpose datasets not tailored to networked quadcopter configurations. Given the lack of prior research on cyber-attack detection, identification, and isolation in multi-quadcopter networks, this work represents a valuable contribution to the literature.

This chapter addresses the problem of cyber-attack detection, identification, and isolation in a network of quadcopters, focusing on attacks such as Denial of Service (DoS), False Data Injection (FDI), and Replay attacks. These attacks have the potential to compromise both sensor and actuator data across all quadcopters in the network, thereby impacting the network's stability and functionality.

The primary objectives within this problem scope are threefold:

1) **Detection**: Identifying the presence of any cyber-attack within the network.
2) **Identification**: Classifying the specific types of detected attack.
3) **Isolation**: Pinpointing the exact quadcopter(s) affected within the network.

To tackle these objectives, this work introduces a Multi-Input, Multi-Output (MIMO) LSTM-based architecture designed specifically for scalability in networked quadcopter systems. Unlike single-purpose

or single-quadcopter models, this MIMO framework accommodates centralized and decentralized configurations, extending the flexibility and resilience of the system across various network topologies.

By employing a shared LSTM backbone with adaptable input heads for each quadcopter, the proposed model scales across networks of multiple quadcopters, from two to five in current tests. This multi-task architecture allows for simultaneous detection, identification, and isolation, consolidating three traditionally separate models into a single, unified framework in Network Topology: The architecture is adaptable to both centralized and decentralized topologies, where the choice of topology impacts the model. In centralized configurations, the model centralizes data from multiple quadcopters, enabling efficient processing at a central node. In contrast, decentralized setups distribute detection tasks across nodes, enhancing network-wide resilience to attacks on specific quadcopters.

4.1.2 Network of Quadcopters

Table 4.1 presents a summary of key aspects related to the network of quadcopters, including various options for network topology, cyber-attack detection locations, movement strategies, network configurations, data transmission methods, and inter-quadcopter communication.

Table 4.1: Different aspects for a network of quadcopters.

Aspect	Options
Network Topology	Centralized, Decentralized, Distributed
Cyber-attack Detection Location	Single Quadcopter, Subset of Quadcopters, All Quadcopters
Movement Strategy	Stochastic, Consensus, Formation, Coverage, Hovering
Network Configuration	Independent Quadcopters, Leader-Follower
Data Transmission	Sensors and Actuators, Sensors Only
Inter-quadcopter Communication	Between Different Quadcopters, With Leader Only

Network Topology: The network topology of quadcopters significantly influences their operational effectiveness and resilience to cyber-attacks. Three primary configurations are typically considered: centralized, decentralized, and distributed. In a centralized topology, all quadcopters transmit their data to a central control point, such as a ground control station (GCS) or a designated quadcopter. While this model simplifies management and coordination, it introduces vulnerabilities by making the central node a potential target for cyber-attacks. In a decentralized topology, a subset of the quadcopters possesses detection capabilities, with the remaining quadcopters sending their data to these nodes and relying on them for detection. In contrast, in a distributed topology, every quadcopter has independent detection capabilities. Each topology offers distinct advantages and challenges, particularly in terms of scalability, resilience, and response time in the face of cyber-attacks.

Cyber-attack Detection Location: The location of cyber-attack detection plays a vital role in the overall security of the quadcopter network. Detection can be performed at a single quadcopter, a subset of quadcopters, or all quadcopters within the network. Detecting attacks at a single quadcopter allows for quicker response times but may leave the system vulnerable if that quadcopter fails or is compromised. Detection at a subset of quadcopters can provide a balanced approach, leveraging the strengths of

certain quadcopters while maintaining a decentralized structure. Finally, implementing detection across all quadcopters enhances overall system security and provides comprehensive monitoring but may introduce significant computational overhead and communication challenges.

Movement Strategy: The movement strategy of quadcopters significantly impacts their ability to operate effectively in dynamic environments. Several strategies exist, including stochastic, consensus, formation, coverage, and hovering. The stochastic approach introduces randomness into movement patterns, which can help avoid predictable behaviors that attackers might exploit. The consensus strategy requires quadcopters to reach a common agreement on movement goals, fostering collaboration but potentially leading to delays in decision-making. Formation strategies enable quadcopters to maintain specific spatial arrangements for enhanced functionality or aesthetics. The coverage strategy aims to monitor a designated area effectively, while hovering allows quadcopters to maintain a fixed position, facilitating detailed data collection. Each strategy has implications for mission success, energy efficiency, and vulnerability to attacks.

Network Configuration: The configuration of the quadcopter network can be characterized as either independent or leader-follower. In an independent configuration, quadcopters operate autonomously, making decisions based on local information and reducing reliance on a central authority. This approach increases resilience, as the failure of one unit does not compromise the entire network. In contrast, a leader-follower configuration designates specific quadcopters as leaders, which guide the movement and actions of their followers. This can enhance coordination and efficiency but may create vulnerabilities if the leader is targeted by an attack. The choice of configuration influences the system's overall flexibility, robustness, and ability to respond to cyber threats.

Data Transmission: Data transmission in a quadcopter network can occur via two primary methods: through sensors and actuators or sensors only. Utilizing both sensors and actuators allows quadcopters to transmit comprehensive information, enabling better decision-making and operational responses. However, this approach may increase the complexity of data management and processing. Conversely, sensors only transmission focuses on collecting environmental and operational data, potentially streamlining communication and reducing processing overhead. The choice of data transmission method can significantly affect the network's responsiveness and resilience to cyber-attacks, necessitating a careful balance between data richness and system efficiency.

Inter-quadcopter Communication: Inter-quadcopter communication can be structured either between different quadcopters or with the leader only. Communication between different quadcopters fosters a collaborative environment where multiple units can share information and support each other's operations, enhancing situational awareness and collective decision-making. On the other hand, communication with the leader only centralizes the information flow, simplifying coordination but potentially creating a bottleneck and single point of failure. The chosen communication strategy influences the network's resilience to attacks, the efficiency of operations, and the ability to adapt to changing conditions in real-time.

4.1.3 Selected Network Cases

Two network configurations are selected for further analysis in this study, each representing distinct topological and operational characteristics:

Case 1: Centralized Network with Single Quadcopter Detection

In this configuration, a centralized topology is employed where a single quadcopter is responsible for cyber-attack detection. The quadcopters follow a stochastic movement strategy, operating independently from each other. Data transmission is considered in two approaches: sensor-only data and combined sensor and actuator data, enabling a comparison of their impact on detection performance. Inter-quadcopter communication occurs between different quadcopters, allowing them to relay their data to the central node. This configuration emphasizes simplicity in network management but introduces potential vulnerabilities at the central detection point, which, if compromised, could impact the entire network's integrity.

Case 2: Decentralized Network with Subset of Quadcopters Detection

In the second configuration, a decentralized topology is utilized, where a subset of the quadcopters has detection capabilities, while the remaining quadcopters send their data to these nodes for analysis. Like Case 1, the quadcopters employ a stochastic movement strategy and operate independently. Data transmission is considered in two approaches: sensor-only data and combined sensor and actuator data, enabling a comparison of their impact on detection performance. Communication occurs between different quadcopters, enabling data-sharing for cyber-attack detection. This decentralized model provides more resilience compared to the centralized approach, as the reliance on multiple detection nodes reduces the risk of total system failure in the event of an attack.

4.2 Methodology

4.2.1 Data Preprocessing

The dataset generated for the network of quadcopters is loaded at this stage and used for training and testing the machine learning models. As a pre-processing step, the time column is removed from the features to prevent the model from learning patterns based on time rather than the actual feature data. The features are then standardized using the StandardScaler, which removes the mean and scales the features to unit variance. This process ensures that each feature has a mean of 0 and a standard deviation of 1, making the data more suitable for machine learning models that are sensitive to feature scaling. Standardization is crucial for improving model performance and ensuring that no single feature dominates due to differences in scale.

Sequences are generated for all quadcopters using the same approach. The method involves creating sequences from time-series data with full overlap to maximize the use of available data. For labeling the sequences, the "last packet label" approach is applied, where the label of the last packet in each sequence is assigned as the label for the entire sequence. This method leverages the temporal information from previous packets in the sequence to assist in determining whether the last packet represents an attack or not.

Once the data has been converted into sequences, it is split into training and testing sets. 75% of the data is allocated for training, with the remaining 25% reserved for testing, ensuring that the model has

enough data to learn from while maintaining a separate set for performance evaluation. Finally, each sequence's packets are separated into independent components corresponding to the features of each quadcopter, allowing them to be used in the appropriate input head of the multi-input deep learning model.

4.2.2 Output Heads Reduction

The dataset for the network of 5 quadcopters provides 50 features, with 10 features corresponding to each quadcopter, and 15 labels, 3 for each quadcopter. In total, there are 15 distinct labels that the machine learning model must predict. However, solving 15 separate machine learning tasks using a single model with shared weights is impractical and inefficient. Therefore, a process is required to reduce the number of outputs from 15 to a more manageable number that the model can handle effectively.

The primary objectives of this problem are the detection, identification, and isolation of cyber-attacks. To achieve these goals, three labels are sufficient, with each label specifically tuned to its corresponding purpose. The structure of each newly created label is outlined as follows:

- **Detection label**: This output is a binary label (0 or 1) representing the presence or absence of an attack. It is derived from the detection labels of individual quadcopters. If any quadcopter's detection label equals 1, indicating an attack, the overall detection label is set to 1.
- **Identification label**: This output identifies the type of attack, distinguishing between Denial of Service (DoS), False Data Injection (FDI), and Replay attacks. The label has three components, each corresponding to one type of attack, and each can be either 0 or 1. For example, if a DoS attack is detected, the first component is set to 1, and similarly for FDI and Replay attacks. This label is constructed by iterating over the identification labels of each quadcopter and assigning the appropriate values based on the attacks present.
- **Isolation label**: This output isolates the attack by identifying which quadcopter(s) is under attack. It has five components, one for each quadcopter, and each component can be either 0 or 1. For instance, if quadcopters 2 and 4 are under attack, the isolation label will be [0, 1, 0, 1, 0]. This label is directly derived from the detection labels of the individual quadcopters, with a value of 1 indicating that a specific quadcopter is compromised.

Once the new labels are created, the original labels are removed and replaced with the newly defined ones. The following Figure demonstrates the newly created labels for the extended problem in the network of quadcopters. Figure 4.1 demonstrates how new network labels are created from individual quadcopters labels for a network with 5 quadcopters.

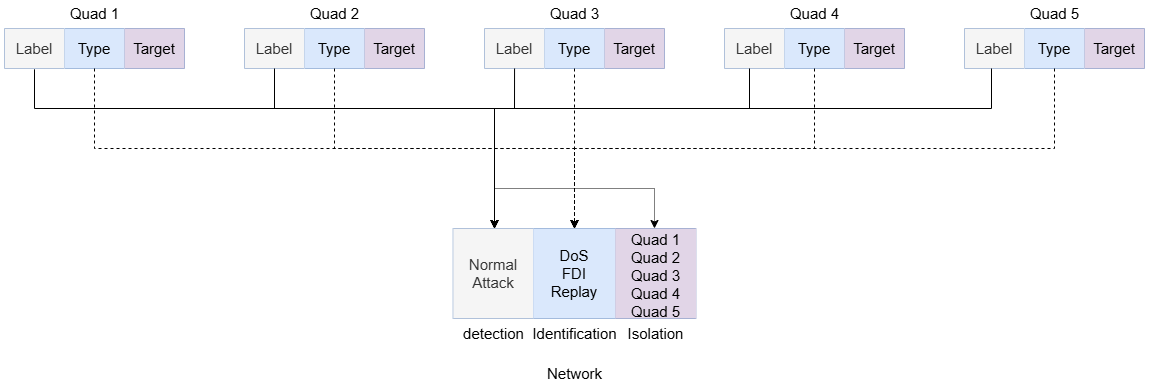


Figure 4.1: Output heads reduction process for 5 quadcopters.

4.2.3 Proposed MIMO LSTM-based Architecture

In this section, the architecture of the proposed Multi-Input Multi-Output (MIMO) model is presented. This model is designed to accommodate both centralized and decentralized topologies, demonstrating its versatility in various network configurations. Furthermore, the MIMO model is adaptable and can be effectively utilized in networks consisting of two to five quadcopters. While it is capable of handling varying quantities of quadcopters, careful tuning of the model enhances its performance while minimizing computational complexity across different scenarios. The MIMO model architecture for network with 5 quadcopters is depicted in Figure 4.2.

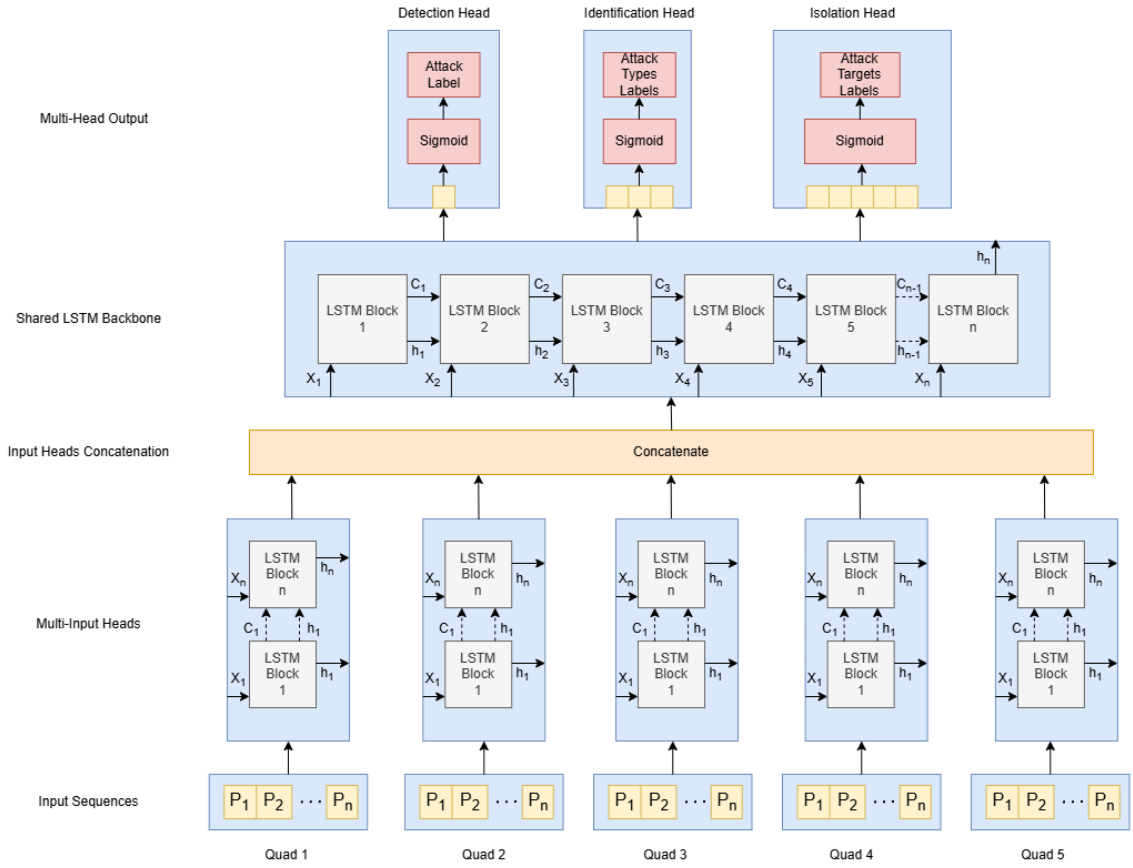


Figure 4.2: MIMO model architecture for network with 5 quadcopters.

The data flow in the model is described as follows. The model begins with five input sequence $P = [P_1, P_2, \dots, P_n]$ which are different for each quadcopter 1 to 5. The input sequence for each quadcopter has a length of n and m features. Each time step of the sequence is fed into the input LSTM network, with each time step processed by an LSTM block according to its index. Specifically, each time step is fed into the LSTM network as $P_1 = X_1, P_2 = X_2, \dots P_n = X_n$, where X_i represents the input at each time step. The data moves through the LSTM network step-by-step, from bottom to up, at each time step, as explained in Section 3.2.3 (LSTM Architecture).

The outputs of all LSTM blocks, represented by h_1 to h_n for each input head are inputs to concatenation layer. This concatenation preserves the sequence length and combines the outputs of each input head to form the input for the shared LSTM backbone. The shared LSTM network has a significantly higher capacity compared to the input heads, as it contains more neurons within each LSTM block. The output from the shared LSTM network, represented by h_n, is fully connected to each of the three separate heads. Each head is responsible for a different task, producing values between 0 and 1 for the output neurons, which are used for detection, identification, and isolation.

The adaptive multi-input multi-output (MIMO) network comprises three principal components:

Adaptive Input Heads: This component consists of an adaptable number of input heads, which are configured according to the number of quadcopters for which data is available for cyber-attack detection. Each input sequence, formed from packets, consists of concatenated features from the respective quadcopter. Each packet's features are separated based on the specific quadcopter they belong to and are then directed to their respective input head. Each input head incorporates a single-layer LSTM network with a predetermined number of blocks, which will be established through experimental trials.

Shared LSTM Backbone: This section consists of a specified number of LSTM blocks arranged in a single layer, the quantity of which will be determined during the experimental phase. It receives outputs from the input heads, concatenates them, and forwards this data into the LSTM backbone, which then transmits its output to three distinct output heads responsible for detection, identification, and isolation. The input LSTM network processes the input sequence and outputs a sequence of the same length, preserving temporal information at each time step. In contrast, the shared LSTM network only outputs the final hidden state, which is then used for detection, identification, and isolation, capturing the overall sequence's learned features.

Static Output Heads: This component includes three distinct output heads designed for the detection, identification, and isolation of cyber-attacks. The detection head comprises one neuron utilizing a sigmoid activation function, with its binary output indicating whether an attack has occurred within the system. The identification head contains three neurons, also employing sigmoid activation functions. Each neuron outputs a binary signal (0 or 1) that signifies the type of attack present in the system: if the first neuron outputs 1, a Denial of Service (DoS) attack is identified; if the second neuron outputs 1, it indicates a False Data Injection (FDI) attack; and if the third neuron outputs 1, it denotes a replay attack. Finally, the isolation head features five neurons, each with a sigmoid activation function. The output of these neurons indicates whether the corresponding quadcopter (based on its index) is under attack, producing a binary output for each quadcopter.

4.2.4 Training and Optimization

Loss Function

The proposed MIMO model architecture incorporates three distinct output heads, each necessitating a separate loss function. Since all output heads are performing binary classification tasks, binary cross-entropy is used for the detection, identification, and isolation heads. However, the loss functions for each head differ based on the number of neurons involved in their respective outputs. The mathematical formulations of these loss functions are provided in the following section.

1) **Detection Head (Single Binary Output):** The detection head in the MIMO model is structured to perform a binary classification that discerns whether any quadcopter system in the network is under attack or operating normally. Leveraging a single output neuron with binary cross-entropy, this head provides a direct indication of abnormal network behavior, allowing for rapid detection of compromised states.

$$L_{detection} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \quad (4.1)$$

where y_i is the true label (0 or 1) and $\hat{y}_i$ is the predicted probability. i is the index of sample in batch and N indicates the number of samples in each batch [87].

2) **Identification Head (Multiple Binary Outputs):** The identification head is tasked with determining the specific types of cyber-attacks impacting the network. It uses multiple binary outputs, each representing a distinct attack type, and applies a binary cross-entropy loss function per output neuron. This setup allows the head to simultaneously identify various attacks such as Denial of Service (DoS) and False Data Injection (FDI). The overall identification loss value is calculated as the average of the loss values from each of the three neurons.

$$L_{identification} = -\frac{1}{N \times M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[y_{i,j}\log\left(\hat{y}_{i,j}\right) + \left(1 - y_{i,j}\right)\log\left(1 - \hat{y}_{i,j}\right)\right], \quad (4.2)$$

where $y_{i,j}$ denotes the true label (0 or 1), and $\hat{y}_{i,j}$ represents the predicted for output neuron j. M refers to the number of output neurons with Sigmoid activation function, which corresponds to the number of attack types that is 3. i is the index of sample in batch and N indicates the number of samples in each batch [87].

3) **Isolation Head (Multiple Binary Outputs):** The isolation head focuses on pinpointing the specific quadcopter(s) under attack within a networked environment. Its multi-output design, tailored for binary classification per unit, enables it to isolate affected quadcopters accurately by signaling which specific nodes within the network are compromised. This head's binary cross-entropy approach per output supports multi-target localization. The overall isolation loss value is calculated as the average of the loss values from each of the three neurons.

$$L_{isolation} = -\frac{1}{N \times P}\sum_{i=1}^{N}\sum_{j=1}^{P}\left[y_{i,j}\log\left(\hat{y}_{i,j}\right) + \left(1 - y_{i,j}\right)\log\left(1 - \hat{y}_{i,j}\right)\right], \quad (4.3)$$

where $y_{i,j}$ denotes the true label (0 or 1), and $\hat{y}_{i,j}$ represents the predicted for output neuron j. P refers to the number of output neurons with Sigmoid activation function, which corresponds to the number of quadcopters that can be any value from 2 to 5. i is the index of sample in batch and N indicates the number of samples in each batch [87].

Weights Update

The Adam optimizer is utilized for model optimization, computing the derivative of the loss function at each step and adjusting the weights according to its algorithm. The overall loss function is determined as follows:

$$L_{total} = L_{detection} + L_{identification} + L_{isolation}, \quad (4.4)$$

The shared LSTM layer within the model utilizes a single set of weights that are applied across all three output heads. As a result, this single LSTM backbone efficiently facilitates the execution of three distinct tasks simultaneously, enabling the model to detect, identify, and isolate cyber-attacks without the need for separate LSTM networks for each task. This architectural design not only reduces computational complexity but also enhances the model's ability to learn shared patterns from the input data.

The gradient for use in Adam optimizer is calculated as follows.

$$\frac{\partial L_{total}}{\partial W} = \frac{\partial L_{detection}}{\partial W} + \frac{\partial L_{identification}}{\partial W} + \frac{\partial L_{isolation}}{\partial W}, \quad (4.5)$$

To avoid redundancy, the full explanation of the Adam optimizer provided in section 3.2.5 will not be repeated here.

4.2.5 Implementation

The implementation of the proposed multi-input multi-output (MIMO) model for the network of quadcopters was conducted using the Python programming language within the Kaggle online notebooks environment. The Kaggle platform, known for its user-friendly interface, provides access to powerful computational resources, enhancing the efficiency of machine learning tasks. Additionally, the platform alleviates challenges associated with library dependencies and execution delays, thereby streamlining the implementation process for the MIMO model in a multi-quadcopter context. TensorFlow is an excellent library utilized for implementing the multi-input multi-output (MIMO) model, providing robust tools for building and training complex neural networks.

Unlike many studies that simply apply popular libraries like TensorFlow or Keras, this work goes beyond basic usage by carefully designing models, preparing data, and formulating the problem. The specific configuration of machine learning tasks, such as detection, identification, and isolation, is tailored to the unique needs of quadcopter networks. Moreover, the precise and transparent use of these libraries distinguishes this approach from others, ensuring clarity and providing a solid foundation for future research.

4.3 Dataset Creation

4.3.1 Dataset Design

The dataset for this study is designed with the following key criteria:

- **Diversity**: Multiple scenarios of network with varying conditions such as simultaneous attacks were included.
- **Size**: Enough samples of network data were generated to support the deep learning model.
- **Relevance**: The data relates entirely to network of quadcopter behavior, with proper labels for normal and attack data.
- **Balance**: Approximately 10% of the network data consists of attack samples.

The dataset consists of multiple scenarios representing the network quadcopters movements. In each scenario, each quadcopter follows a trajectory, during which cyber-attacks are applied.
Features of the dataset are demonstrated in Table 4.2.

Table 4.2: Features of dataset.

Feature	Description
Number of samples	50,000 (including both normal and attack data)
Number of scenarios	50 (simulation-based network of quadcopters movement scenarios)
Number of quadcopters	5 (number of quadcopters in the network)
Number of features	50 (actuator and sensor data of quadcopters)
Data type	Time series
Features type	Numerical (sensor readings and actuator commands)
Class distribution	10% attack data / 90% normal data
Labels	15: detection (binary), identification (multiclass), isolation (multiclass) for each quadcopter

Scenarios are generated using Monte Carlo approach by randomly selecting variables at each scenario. These variables are defined in Table 4.3 which will be set differently for each quadcopter in the network.

Table 4.3: Monte Carlo variables.

Variable	Description
Trajectory points	Reference points that define the quadcopter's path during its movement
Attack targets	Set of specific sensors and actuators targeted during the scenario
Number of targets	3-6 (Number of individual attacks applied in the scenario)
DoS start	Start time of the Denial of Service (DoS) attack
DoS duration	1-2 seconds (Duration of DoS attack)
FDI start	Start time of the False Data Injection (FDI) attack
FDI duration	1-2 seconds (Duration of FDI attack)
FDI value	The false value injected into the system during the FDI attack
Record start	Start time of data recording in the simulation
Replay start	Start time of the replay attack
Replay duration	1-2 seconds (Duration of replay attack)

Variables of the simulation for each scenario are defined in Table 4.4.

Table 4.4: Simulation variables.

Variable	Value
Simulation time (seconds)	100
Step value (seconds)	0.1
Number of trajectory points	10
Time interval between setpoints (seconds)	10

The attack column labels for each quadcopter are demonstrated in Table 4.5.

Table 4.5: Attack column labels.

Variable	Values
Attack label	0 (Normal), 1 (Attack)
Attack Types	1 (DoS), 2 (FDI), 3 (Replay)
Attack Targets	$x, y, z, \phi, \theta, \psi, T, \tau_\phi, \tau_\theta, \tau_\psi$

4.3.2 Dataset Implementation

The data generated from the Simulink simulations were transferred to MATLAB using the Data Inspector tool. This dataset contains 4 actuator variables and 6 sensor variables per quadcopter, totaling 50 variables for the network. In MATLAB, additional columns such as time, attack labels (including label, type, and target) were added for each quadcopter. The time variable, originally from the Simulink Data Inspector, was further refined in MATLAB. Given the simulation's time range (0 to 100 seconds at 0.1-second intervals), an extra data point at time 100 was included, resulting in 1001 data points per scenario. To ensure consistency in the attack detection process, the last row of each simulation was removed, reducing the dataset to 1000 data points per scenario. The final dataset was then formatted and exported as a CSV file.

4.4 Simulation Results

4.4.1 Comparison between Different Sequence Lengths

In this section, the focus is on comparing different configurations of paired LSTM blocks (16, 64), (32, 128), and (64, 256) across varying sequence lengths (20, 40, and 100). These experiments aim to evaluate how the combination of smaller LSTM units in the initial layers, followed by larger LSTM units in the shared layer, impacts model performance when processing sequences of different lengths. The MIMO LSTM architecture allows for more granular temporal processing at the input stage, with each input being fed into its respective LSTM block before being merged into a shared LSTM layer. The paired LSTM block sizes influence the model's ability to capture both localized and more extensive temporal dependencies in the data, while the sequence length dictates how much temporal information the model can leverage. The interplay between these factors offers insights into how different LSTM configurations and sequence lengths affect the model's ability to learn and generalize temporal patterns.

The experiments and comparisons cover all three critical tasks: detection, identification, and isolation, ensuring a comprehensive evaluation of the model's performance across different attack-handling capabilities. Due to the large number of metrics in the experiments, the comparison focuses on two key metrics: F1-score and accuracy. These metrics were selected for their balanced assessment of both prediction quality and overall performance, making them ideal for evaluating the trade-off between precision and recall, as well as general correctness. Table 4.6 lists the experimental parameters, and figures 4.3, 4.4 and 4.5 present the results for detection, identification, and isolation respectively. Additionally, Figure 4.6 compares the training times across these configurations, highlighting the computational trade-offs introduced by increased model complexity.

Table 4.6: Parameters for sequence lengths experiments.

Parameter	Value
Num scenario	30
Num quadcopter	4
Sequence overlap	Full
Epoch	50
Batch	128
Sequence labeling	Last packet
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Binary crossentropy
Loss isolation	Binary crossentropy
Average approach	Macro
Data transferred	Both sensors and actuators

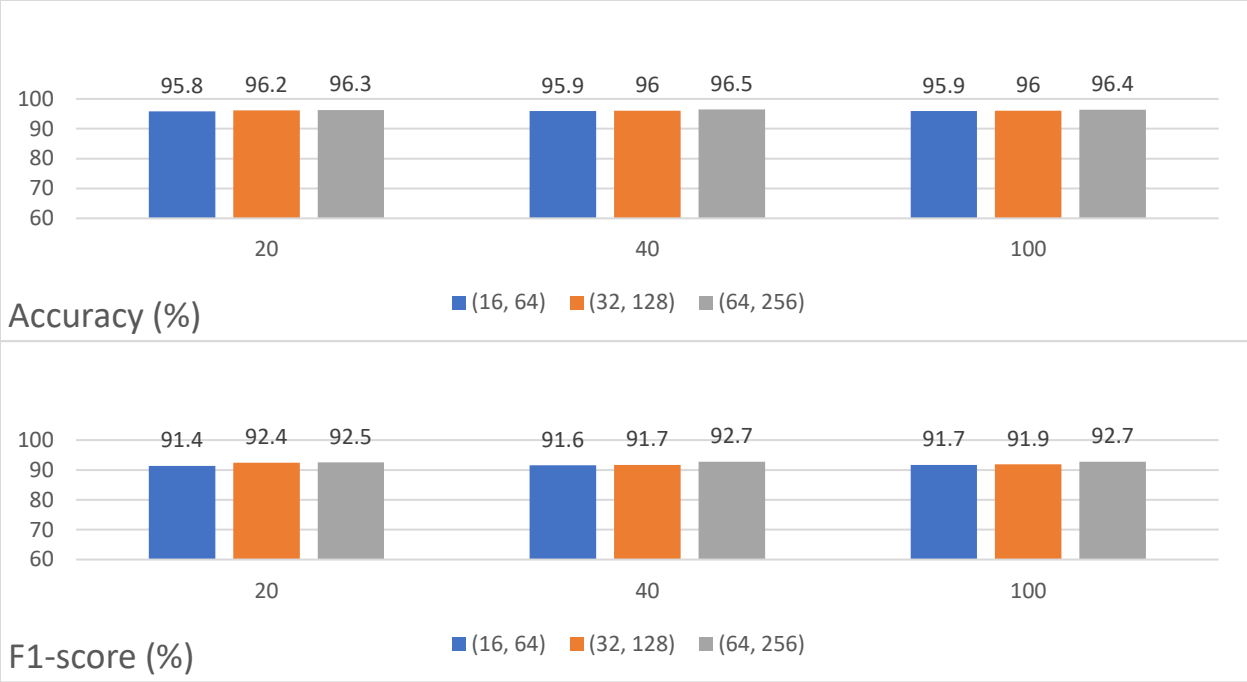


Figure 4.3: Cyber-attack detection performance in experiments with different sequence length and LSTM blocks.

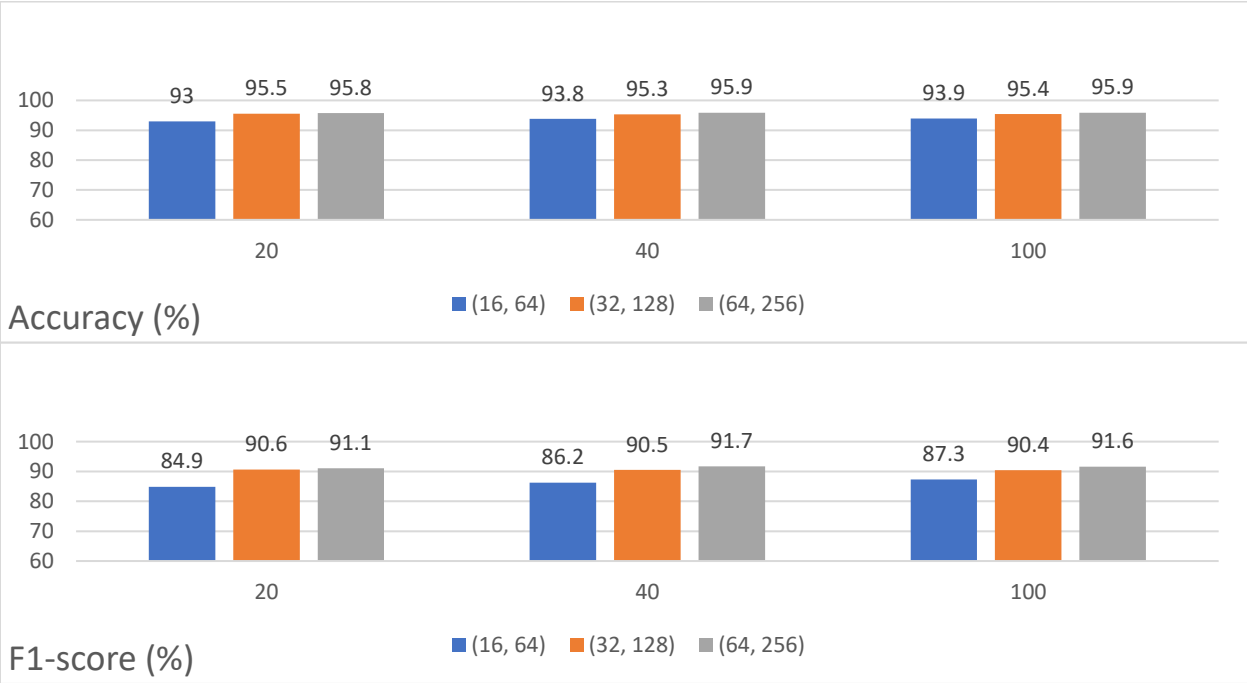
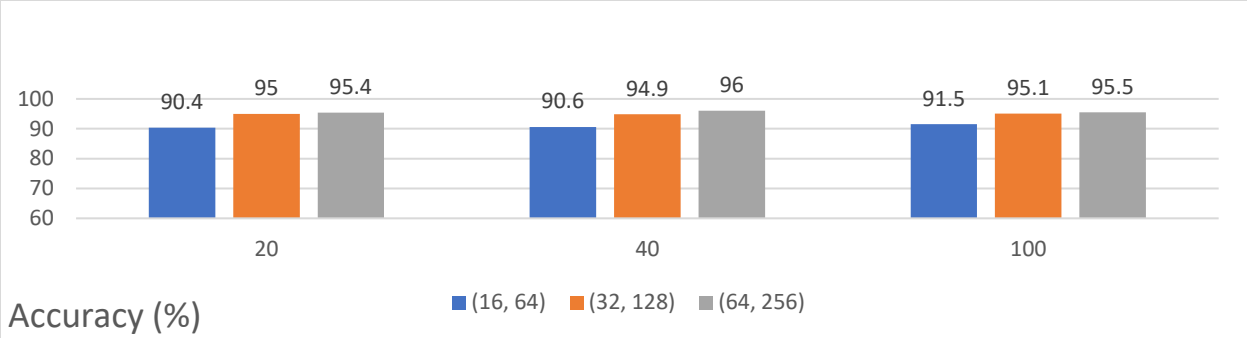


Figure 4.4: Cyber-attack Identification performance in experiments with different sequence length and LSTM blocks.



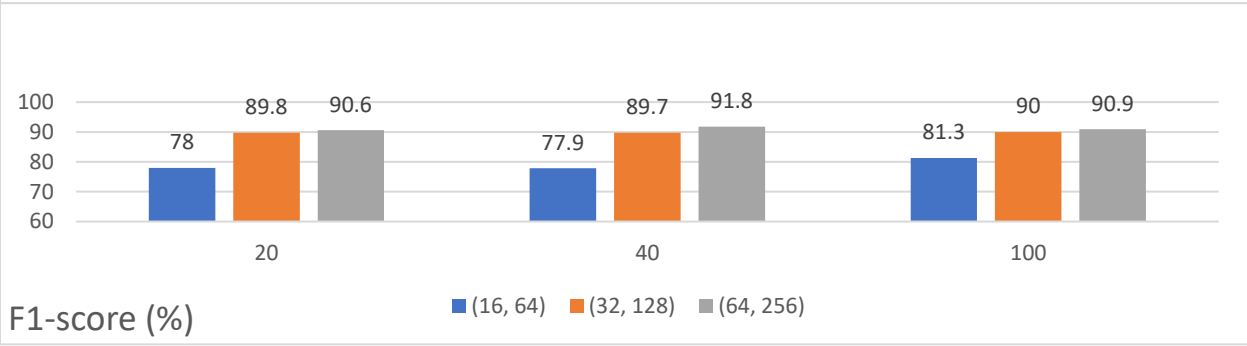


Figure 4.5: Cyber-attack isolation performance in experiments with different sequence length and LSTM blocks.

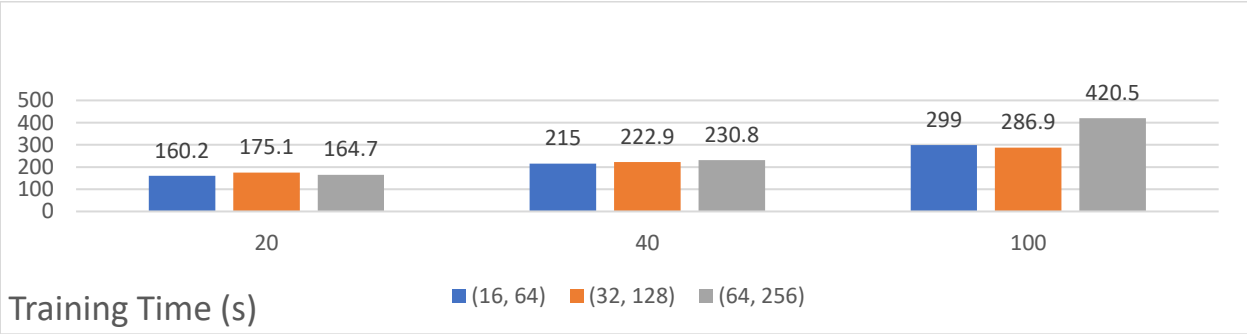


Figure 4.6: Training times in experiments with different sequence length and LSTM blocks.

Observations: This section presents a detailed analysis of the results from the comparison of different sequence lengths and LSTM configurations in the context of cyber-attack detection, identification, and isolation for a network of quadcopters.

1) **Effect of Changing Sequence Length:**
   a) **Sequence Length 20**: At a sequence length of 20, the models show varied performance, with the architecture (64, 256) demonstrating the highest detection accuracy at 0.963 and an F1-score of 0.925, closely followed by (32, 128) with 0.962 accuracy and an F1-score of 0.924. The (16, 64) model shows slightly lower performance, achieving an accuracy of 0.958 and an F1-score of 0.914. In terms of identification, the (32, 128) model leads with an accuracy of 0.955 and an F1-

score of 0.906, whereas the (16, 64) model has the lowest identification accuracy at 0.930 and an F1-score of 0.849. Isolation performance also reflects a similar trend, with the (64, 256) model achieving an accuracy of 0.954 and an F1-score of 0.906, while the (16, 64) model ranks lowest with an accuracy of 0.904 and an F1-score of 0.780. The training times for this length vary slightly, with the (16, 64) model at 160.2 seconds and (32, 128) model taking 175.1 seconds, compared to the (64, 256) model at 164.7 seconds.

b) **Sequence Length 40**: At sequence length 40, there is a noticeable consistency in performance, with the (64, 256) model achieving the highest detection accuracy of 0.965 and an F1-score of 0.927, reflecting a significant improvement from the previous length. The (32, 128) architecture follows closely with an accuracy of 0.960 and an F1-score of 0.919. Meanwhile, the (16, 64) model shows slightly lower performance with an accuracy of 0.959 and an F1-score of 0.916. In identification, the (64, 256) model again leads with an accuracy of 0.959, while the (32, 128) model scores 0.953, demonstrating competitive performance. Isolation performance remains stable across the models, with the (64, 256) architecture achieving the highest accuracy of 0.960 and an F1-score of 0.918. Training times increase across the board, with the (16, 64) model taking 215.0 seconds, the (32, 128) model at 222.9 seconds, and the (64, 256) model at 230.8 seconds.

c) **Sequence Length 100**: With a sequence length of 100, the detection accuracy and F1-scores reach their peak, with the (64, 256) architecture achieving the highest detection accuracy of 0.964 and an F1-score of 0.927, indicating a slight improvement compared to the previous length. The (32, 128) model also shows strong performance with an accuracy of 0.960 and an F1-score of 0.919. The (16, 64) model's performance remains stable, with an accuracy of 0.959. In identification, the (64, 256) model again leads with an accuracy of 0.959, while the (32, 128) architecture records 0.954, showcasing robust performance across the sequence lengths. Isolation performance sees slight variations, with the (64, 256) model achieving 0.955 accuracy, closely followed by the (32, 128) model at 0.951. However, this sequence length incurs a higher training time, with the (64, 256) model taking the longest at 420.5 seconds, reflecting the increased complexity of processing longer sequences. The (16, 64) model takes 299.0 seconds, while the (32, 128) model records 286.9 seconds.

2) **Effect of Changing LSTM Blocks:**

a) **(16, 64) Configuration**: In the (16, 64) configuration, the model exhibits varying performance metrics across different sequence lengths. At a sequence length of 20, the detection accuracy stands at 0.958, accompanied by an F1-score of 0.914. For identification, the model achieves an accuracy of 0.930 and an F1-score of 0.849, while isolation performance is slightly lower, with an accuracy of 0.904 and an F1-score of 0.780. Training time for this configuration is relatively efficient, at 160.2 seconds. With sequence lengths of 40 and 100, the performance remains stable but demonstrates slight improvements, particularly in detection and identification accuracy. The training times increase progressively, with the model taking 215.0 seconds at 40 and 299.0 seconds at 100, indicating a balance between performance gains and processing time.

b) **(32, 128) Configuration**: The (32, 128) configuration shows a notable enhancement in performance across all metrics compared to the (16, 64) setup. At a sequence length of 20, detection accuracy rises to 0.962, with an F1-score of 0.924. Identification performance also improves, achieving an accuracy of 0.955 and an F1-score of 0.906. Isolation metrics reflect similar gains, with an accuracy of 0.950 and an F1-score of 0.898. Training time for this configuration is slightly higher than (16, 64), recorded at 175.1 seconds. As sequence lengths increase to 40 and 100, the (32, 128) model continues to perform competitively, maintaining an accuracy of 0.960 at 40 and slightly increasing to 0.960 at 100. However, the training times also increase more substantially, reaching 222.9 seconds at 40 and 286.9 seconds at 100, highlighting a trade-off between model complexity and training duration.

c) **(64, 256) Configuration**: The (64, 256) configuration showcases the best performance metrics among the three architectures, particularly at longer sequence lengths. At a sequence length of 20, the model achieves a detection accuracy of 0.963 and an F1-score of 0.925, indicating a clear advantage over both (16, 64) and (32, 128). The identification performance follows suit with an accuracy of 0.958 and an F1-score of 0.911. Isolation metrics are also impressive, with an accuracy of 0.954 and an F1-score of 0.906. Training time is optimized at 164.7 seconds, making it the most efficient among the configurations for this sequence length. With a sequence length of 40, the performance further improves, with detection accuracy reaching 0.965 and an F1-score of 0.927. The training time increases slightly to 230.8 seconds. At a sequence length of 100, the model achieves peak performance with a detection accuracy of 0.964 and an F1-score of 0.927, although this comes at a significant training time of 420.5 seconds, suggesting the model's complexity leads to better outcomes but requires more computational resources.

Key Findings and Analysis: The models utilizing a sequence length of 40 demonstrated the best and most reliable performance while maintaining manageable training times. In comparison, models with a sequence length of 20 exhibited lower performance levels, while those with a sequence length of 100 offered similar results but at a higher training cost. Among the models with a sequence length of 40, the configuration (16, 64) produced relatively poor results. Conversely, both the (32, 128) and (64, 256) configurations yielded promising performance metrics. Notably, the (64, 256) configuration outperformed the (32, 128) model by approximately 1-2% in F1-score, indicating a slight yet meaningful enhancement in performance.

Consequently, both configurations (32, 128) and (64, 256) with a sequence length of 40 are deemed optimal for deployment. The smaller configuration (32, 128) is particularly suited for scenarios involving 2 to 3 quadcopters, which often operate within decentralized network topologies. In contrast, the larger configuration (64, 256) is preferred for applications involving 4 to 5 quadcopters, where the increased model complexity can enhance overall performance, typically observed in centralized network topologies.

4.4.2 Comparison between Sensor-Only and Sensor-Actuator Data Transmission

In the second set of experiments, a comparative analysis is conducted between two distinct data transmission approaches: sensor-only transmission and sensor-actuator transmission. In the sensor-only scenario, only sensor data is transmitted between the quadcopters, which is utilized for cyber-attack detection, identification, and isolation. Conversely, in the sensor-actuator configuration, both sensor and actuator data are transmitted among the quadcopters, enhancing the system's capacity for comprehensive cyber-attack detection, identification, and isolation. The experiments are designed to evaluate the performance of the models under varying configurations of quadcopters, specifically with 2-3 quadcopters using the (32, 128) LSTM block configuration, and with 4-5 quadcopters utilizing the (64, 256) LSTM block configuration. This approach enables a detailed comparison of data transmission methods across different scenarios, while also assessing the models' performance with varying numbers of quadcopters.

A comprehensive evaluation is conducted across all three critical tasks: detection, identification, and isolation, ensuring a thorough assessment of the models' capabilities in handling various types of cyber-attacks. Given the extensive range of performance metrics collected during the experiments, the analysis primarily focuses on two key metrics: F1-score and accuracy. These metrics were selected for their ability to provide a balanced evaluation of prediction quality and overall performance, making them particularly suited for assessing the trade-offs between precision and recall, as well as general correctness. Table 4.7 presents the experimental parameters, while Figures 4.7, 4.8, and 4.9 illustrate the results for detection, identification, and isolation, respectively. Additionally, Figure 4.10 compares the training times across these configurations, thereby highlighting the computational trade-offs associated with increased model complexity.

Table 4.7: Parameters for different data transmission experiments.

Parameter	Value
Num scenario	30
LSTM blocks input	32-64
LSTM blocks shared	128-256
Sequence length	40
Sequence overlap	Full
Epoch	50
Batch	128
Sequence labeling	Last packet
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Binary crossentropy
Loss isolation	Binary crossentropy
Average approach	Macro

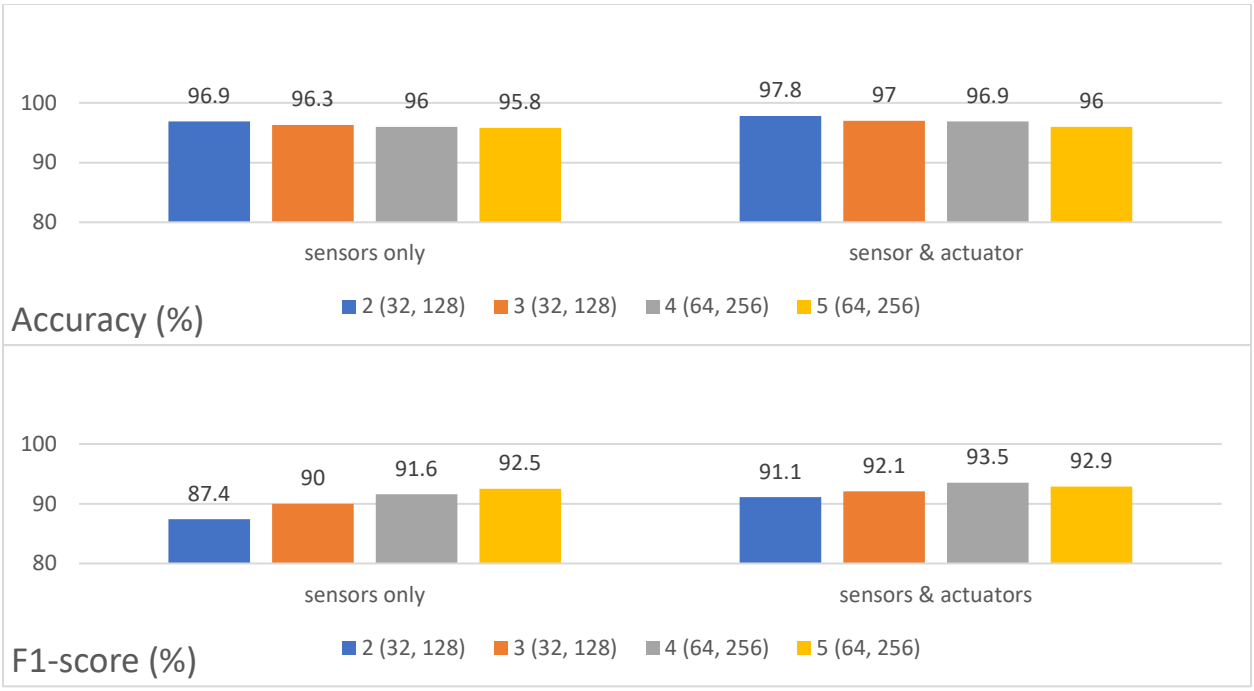


Figure 4.7: Cyber-attack detection performance in experiments with different transmitted data and number of quadcopters.

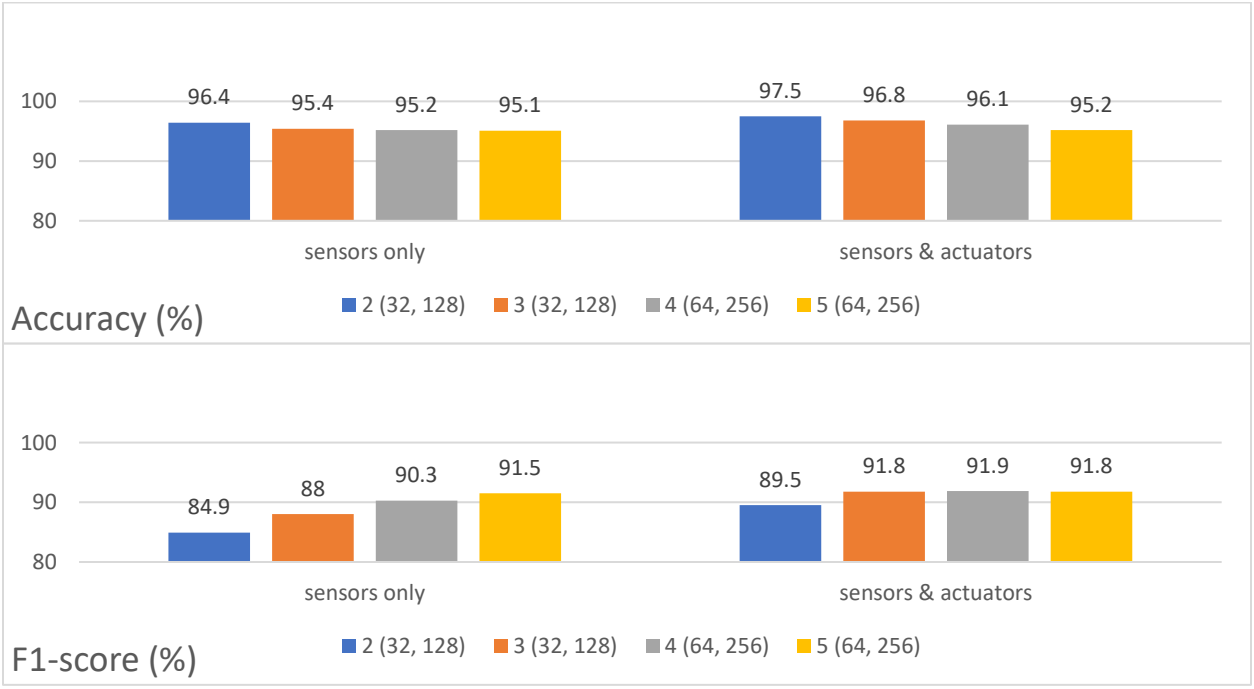


Figure 4.8: Cyber-attack identification performance in experiments with different transmitted data and number of quadcopters.

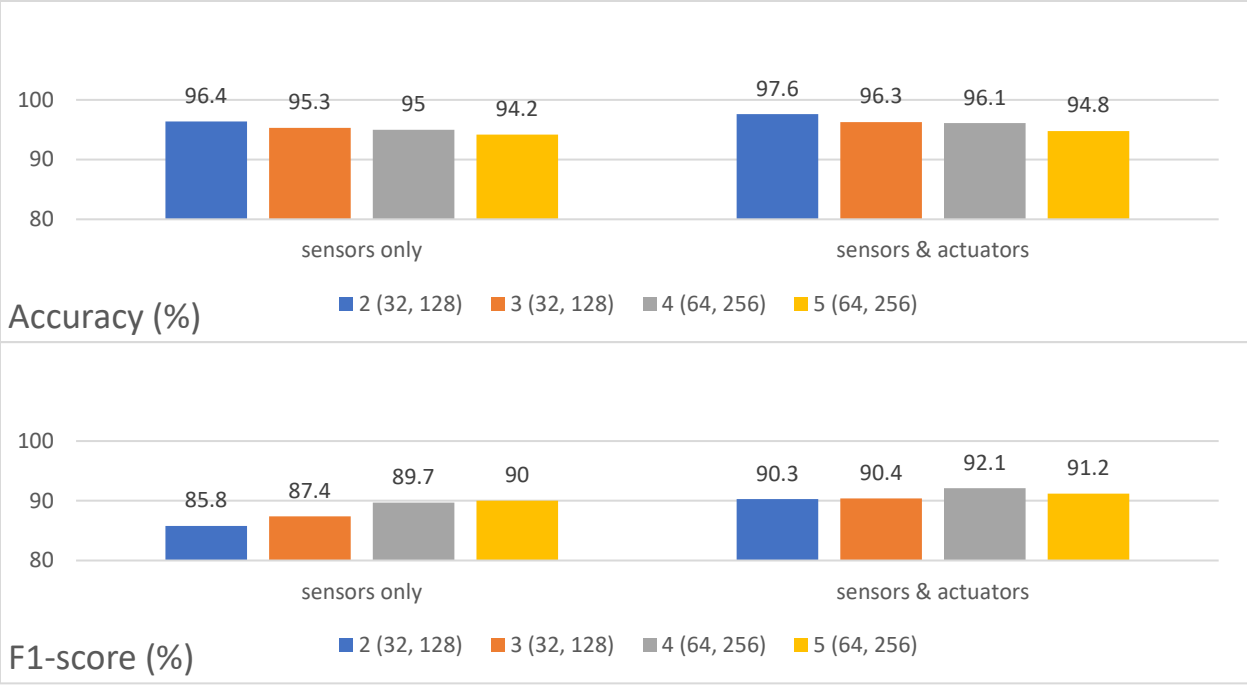


Figure 4.9: Cyber-attack isolation performance in experiments with different transmitted data and number of quadcopters.

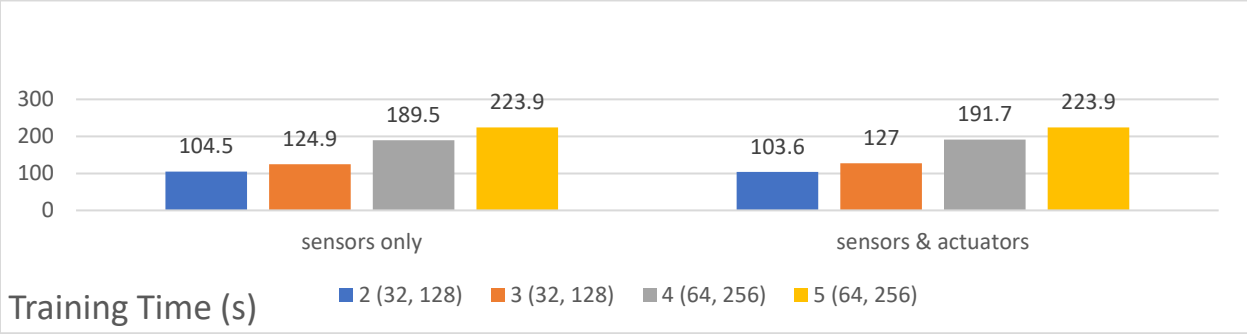


Figure 4.10: Training times in experiments with different transmitted data and number of quadcopters.

Observations:

1) **Effects of having only sensors data or sensors and actuators data:**
   a) **Only sensors data:** The experiments utilizing only sensor data reveal varying performances across detection, identification, and isolation tasks, particularly when considering the F1-score as the primary metric for evaluation. For the configuration with 2 quadcopters, the model achieved an F1-score of 0.874 for detection, 0.849 for identification, and 0.858 for isolation, with a training time of 104.5 seconds. As the number of quadcopters increased to three, the F1-score

for detection improved to 0.900, while the identification and isolation scores also increased to 0.880 and 0.874, respectively, despite the slightly longer training time of 124.9 seconds. In configurations with 4 and 5 quadcopters, the F1-scores continued to rise, reaching 0.916, 0.903, and 0.897 for detection, identification, and isolation tasks in the 4 quadcopters scenario, and 0.925, 0.915, and 0.900 in the 5 quadcopters scenario, with training times of 189.5 and 223.9 seconds, respectively. Overall, while the accuracy remained relatively stable across these configurations, the consistent increase in F1-scores indicates an improvement in the model's capability to balance precision and recall, thereby enhancing its overall effectiveness in cyber-attack detection and handling.

b) **Both sensors and actuators data:** In contrast, the experiments incorporating both sensor and actuator data demonstrated superior performance across all tasks. For the configuration with 2 quadcopters, the model achieved an F1-score of 0.911 for detection, 0.895 for identification, and 0.903 for isolation, with a training time of 103.6 seconds. As the number of quadcopters increased to three, the F1-scores remained strong at 0.921 for detection, 0.918 for identification, and 0.904 for isolation, albeit with a slight increase in training time to 127.0 seconds. In the 4 quadcopters scenario, the model maintained high performance, achieving F1-scores of 0.935, 0.919, and 0.921 for detection, identification, and isolation tasks, with a training time of 191.7 seconds. The trend continued for the 5 quadcopters configuration, where the F1-scores were 0.929, 0.918, and 0.912, with the training time reaching 223.9 seconds. Overall, the inclusion of actuator data significantly enhanced the model's performance across all evaluation metrics, as evidenced by the higher F1-scores, indicating improved precision and recall in cyber-attack detection and handling tasks. The results underscore the value of incorporating both sensor and actuator data for more effective and reliable system performance.

2) **Effects of change in number of quadcopters:**

a) **2 Quadcopters:** With two quadcopters, using sensor-only data results in F1-scores of 0.874 for detection, 0.849 for identification, and 0.858 for isolation. When actuator data is added, these scores improve to 0.911, 0.895, and 0.903, respectively. The addition of actuator data shows a clear improvement. Compared to larger quadcopter cases, this setup has slightly lower performance, especially as the F1-scores rise with more quadcopters, but still shows solid gains. Training time is almost equal for both configurations, with 104.5 seconds for sensor-only and 103.6 seconds for sensor-actuator, indicating efficiency with two quadcopters.

b) **3 Quadcopters:** For three quadcopters, F1-scores in the sensor-only case improve to 0.900 for detection, 0.880 for identification, and 0.874 for isolation. With sensor-actuator data, these rise to 0.921, 0.918, and 0.904, respectively. The improvement with actuator data remains clear across all tasks. When comparing with two quadcopters, the F1-scores are slightly better, especially in identification and isolation, showing the model benefits from more quadcopters. Training times are also close, with 124.9 seconds for sensor-only and 127.0 seconds for sensor-actuator data, slightly higher but acceptable.

c) **4 Quadcopters:** In the case of four quadcopters, sensor-only F1-scores for detection, identification, and isolation are 0.916, 0.903, and 0.897, respectively. With both sensor and actuator data, these improve to 0.935, 0.919, and 0.921. The improvement is consistent but smaller compared to the jump seen between two and three quadcopters, especially in detection. Compared to the three quadcopters, both configurations show better performance, particularly in isolation. Training time increases moderately to 189.5 seconds for sensor-only and 191.7 seconds for sensor-actuator, showing scalability with added complexity.

d) **5 Quadcopters:** With five quadcopters, the sensor-only configuration gives F1-scores of 0.925 for detection, 0.915 for identification, and 0.900 for isolation, while the sensor-actuator configuration improves to 0.929, 0.918, and 0.912. The performance here is almost equal to the four-quadcopter case, with slight improvements across all tasks. Compared to previous cases, the gains are smaller, suggesting a performance plateau with higher numbers of quadcopters. Training time remains unchanged at 223.9 seconds for both configurations, which is higher than earlier cases but stable when using more quadcopters.

Key Findings and Analysis: Based on the experimental results, the use of both actuator and sensor data consistently improves performance across all tasks: detection, identification, and isolation in every scenario. The improvement is particularly pronounced when fewer quadcopters are involved, highlighting the added value of actuator data in less complex configurations. As the number of quadcopters increases, the training time shows a clear upward trend, reflecting the increased complexity of the problem. For sensor-only data, the results show noticeable improvement with a larger number of quadcopters. In contrast, when both sensor and actuator data are used, the performance remains similar, with results remaining relatively stable across different quadcopter configurations. Notably, the performance is slightly lower with two quadcopters for both data types, indicating that more data diversity contributes to stronger model outcomes.

4.4.3 Comparison between Centralized and Decentralized Topologies

In the third set of experiments, the performance of the proposed Multiple-Input Multiple-Output (MIMO) model is evaluated under two distinct network topologies: centralized and decentralized. These experiments focus on cyber-attack detection, identification, and isolation within a network of five quadcopters. The primary goal is to assess how the different topologies influence the model's performance and computational requirements across the three key tasks. For this comparison, the centralized topology involves four quadcopters transmitting their sensor and actuator data to a single central quadcopter responsible for handling cyber-attack detection, identification, and isolation. In contrast, the decentralized topology splits the detection capabilities between two quadcopters. Specifically, quadcopters 1 and 4 are responsible for detection, while quadcopters 2 and 3 send their data to quadcopter 1, and quadcopter 5 sends its data to quadcopter 4. Figures 4.11 and 4.12 provide a visual representation of these centralized and decentralized topologies, respectively.

In the decentralized configuration, the two sets of results from quadcopters 1 and 4 are averaged for performance metrics such as accuracy, precision, recall, and F1-score. Meanwhile, the total training time

for the decentralized approach is calculated by summing the individual training times of the two models used by quadcopters 1 and 4. To ensure a comprehensive evaluation, the comparison covers all three critical tasks: detection, identification, and isolation, across both topologies. The performance metrics accuracy, precision, recall, and F1-score are used to assess the overall effectiveness of the models in handling various types of cyber-attacks. Table 4.8 provides a summary of the experimental parameters, while Figures 4.13 illustrate the performance results for detection, identification, and isolation. Furthermore, Figure 4.14 compares the training times for both centralized and decentralized approaches, shedding light on the computational trade-offs associated with increasing model complexity.

Table 4.8: Parameters for comparison centralized and decentralized topologies experiments.

Parameter	Value
Num scenario	30
LSTM blocks input	32-64
LSTM blocks shared	128-256
Sequence length	40
Sequence overlap	Full
Epoch	50
Batch	128
Sequence labeling	Last packet
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Binary crossentropy
Loss isolation	Binary crossentropy
Average approach	Macro
Data transferred	Both sensors and actuators

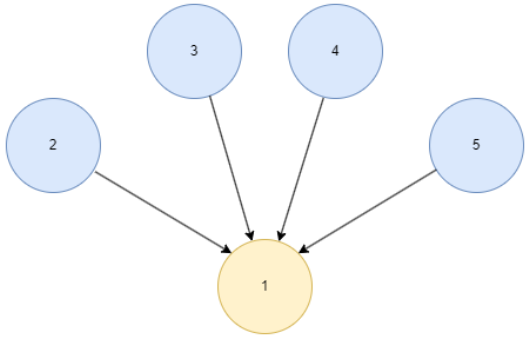


Figure 4.11: Centralized network topology with 5 quadcopters.

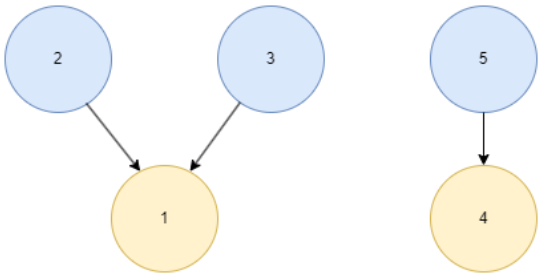


Figure 4.12: Decentralized network topology with 5 quadcopters.

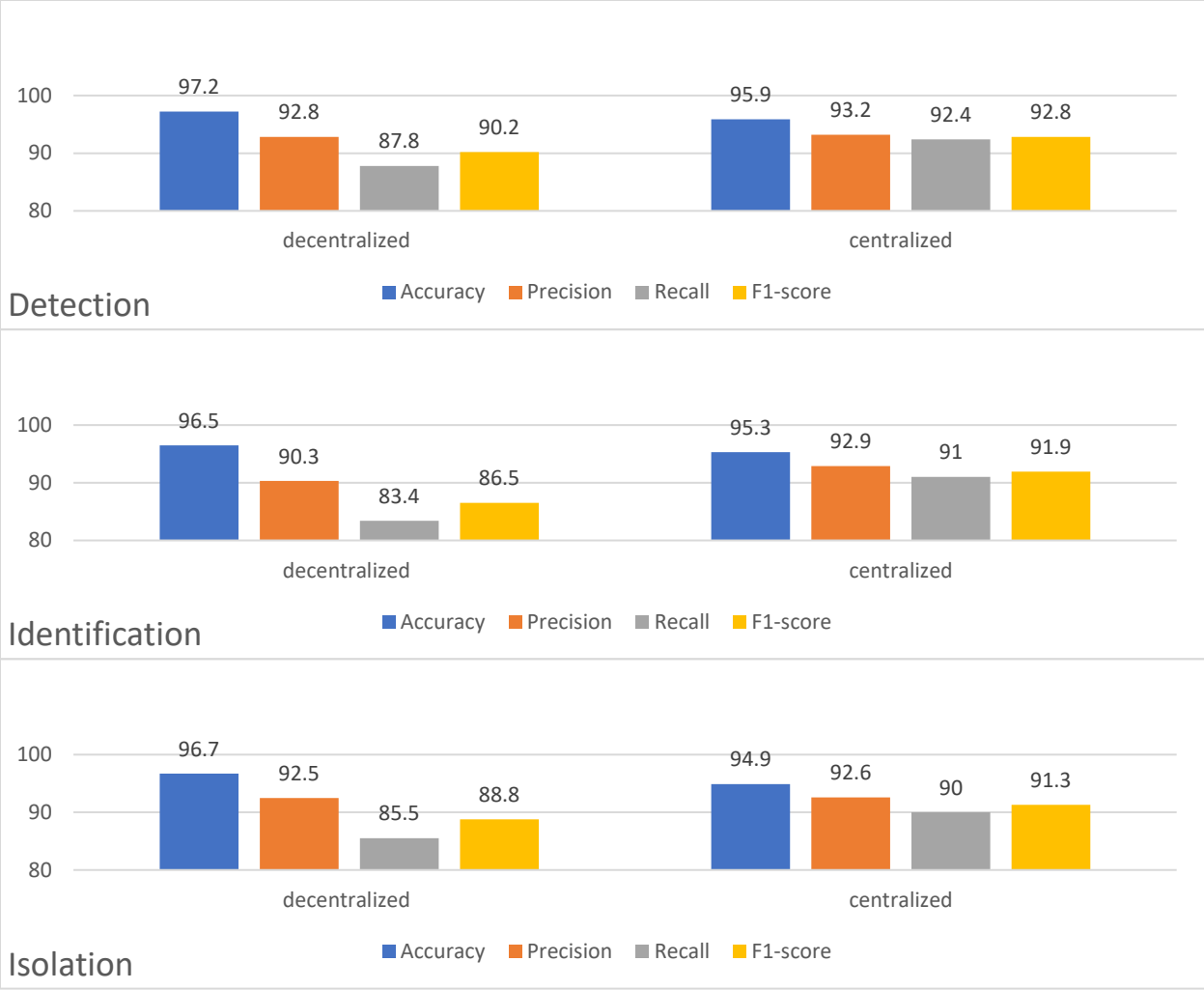


Figure 4.13: Cyber-attack detection, identification, and isolation performance in experiments between decentralized and centralized topologies.

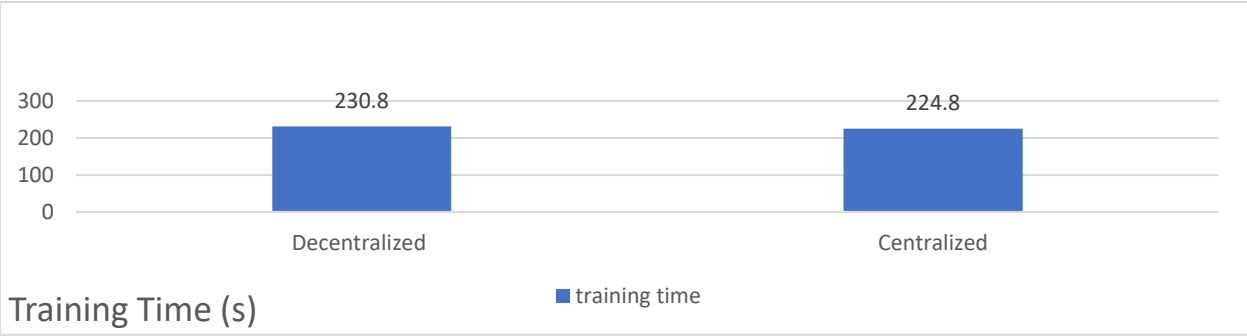


Figure 4.14: Training times in experiments between decentralized and centralized topologies.

Observations:

1) **Detection Performance:** In the detection task, the centralized approach outperformed the decentralized configuration, especially when considering the F1-score, which is the key metric for balancing precision and recall. The centralized system achieved a higher F1-score of 0.928, compared to 0.902 in the decentralized approach. Although the decentralized approach had a slightly better accuracy (0.972 vs. 0.959), the F1-score indicates that the centralized system is more reliable in handling detection, offering a better trade-off between precision (0.932) and recall (0.924). Therefore, the centralized system is clearly superior in detection performance.

2) **Identification Performance:** For identification, the centralized approach also shows better performance, with an F1-score of 0.919, significantly higher than the decentralized system's F1-score of 0.865. While the decentralized system achieved higher accuracy (0.965 vs. 0.953), the lower F1-score indicates a weaker balance between precision and recall. The centralized system also had higher precision (0.929) and recall (0.910) than the decentralized approach (precision: 0.903, recall: 0.834), making it more effective at identifying attacks with fewer false positives and false negatives.

3) **Isolation Performance:** When it comes to isolation, the centralized approach again performs better, with an F1-score of 0.913 compared to 0.888 for the decentralized configuration. Although the decentralized system had slightly higher accuracy (0.967 vs. 0.949), the F1-score once more highlights that the centralized system strikes a better balance between precision and recall. The centralized system's precision (0.926) and recall (0.900) are both slightly higher than the decentralized approach (precision: 0.925, recall: 0.855), further confirming its superiority in isolation tasks.

4) **Training Time:** In terms of training time, the centralized approach was also more efficient, taking 224.8 seconds compared to 230.8 seconds for the decentralized system. Although the difference is minimal, it still highlights the centralized system's efficiency, especially since it consistently performs better across all tasks. The slightly higher training time in the decentralized approach adds complexity without providing better performance, making the centralized topology the clear winner both in terms of computational efficiency and detection accuracy.

Key Findings and Analysis: The overall results indicate that the centralized approach consistently achieved superior F1-scores across all tasks, including detection, identification, and isolation. This improvement in F1-score highlights the centralized system's enhanced ability to maintain a balance between precision and recall, which is crucial for effectively identifying cyber threats. Additionally, the centralized approach demonstrated a slightly lower training time compared to the decentralized configuration.

Conversely, while the decentralized approach exhibited marginally higher accuracy in all tasks, this metric alone may not adequately reflect the system's overall performance. Accuracy can be misleading in scenarios where the class distribution is imbalanced or where false positives and false negatives carry different consequences. The higher F1-scores in the centralized model indicate a more reliable and

robust performance, as this metric considers both precision and recall, providing a comprehensive assessment of the model's effectiveness.

Ultimately, these findings suggest that the proposed MIMO model operates more effectively with a higher number of quadcopters, particularly in a centralized topology. The ability to optimize detection and response capabilities while minimizing training time underlines the advantages of the centralized approach in managing complex networks of quadcopters in cyber-attack scenarios.

4.4.4 Model Performance Analysis and Comparison with Existing Approaches

The fourth set of experiments is dedicated to evaluating the final model configured with optimal parameters and comparing its performance against similar works in the existing literature. This analysis aims to demonstrate the superiority of the proposed model in detecting, identifying, and isolating cyber-attacks. Notably, there is currently no research in the literature that addresses the detection of cyber-attacks within networks of quadcopters, making this study pioneering in its field. Consequently, the identification and isolation of cyber-attacks presented herein also represent unique contributions to the domain. This section also includes an experiment to measure the real-time detection capability of the model. The detection time will be evaluated to assess the model’s efficiency and suitability for real-time applications. Table 4.9 summarizes the model parameters employed during this experiment. Furthermore, the results for detection, identification, and isolation are illustrated in heatmap format, as shown in Figures 4.15, 4.16, and 4.17, respectively.

Table 4.9: Parameters for comparative analysis experiments.

Parameter	Value
Num scenario	50
Num quadcopter	5
LSTM blocks input	64
LSTM blocks shared	256
Sequence length	40
Sequence overlap	Full
Epoch	50
Batch	128
Sequence labeling	Last packet
Normalization and standardization	Standardization
Optimizer	Adam
Loss detection	Binary crossentropy
Loss identification	Binary crossentropy
Loss isolation	Binary crossentropy
Average approach	Macro
Data transferred	Both sensors and actuators

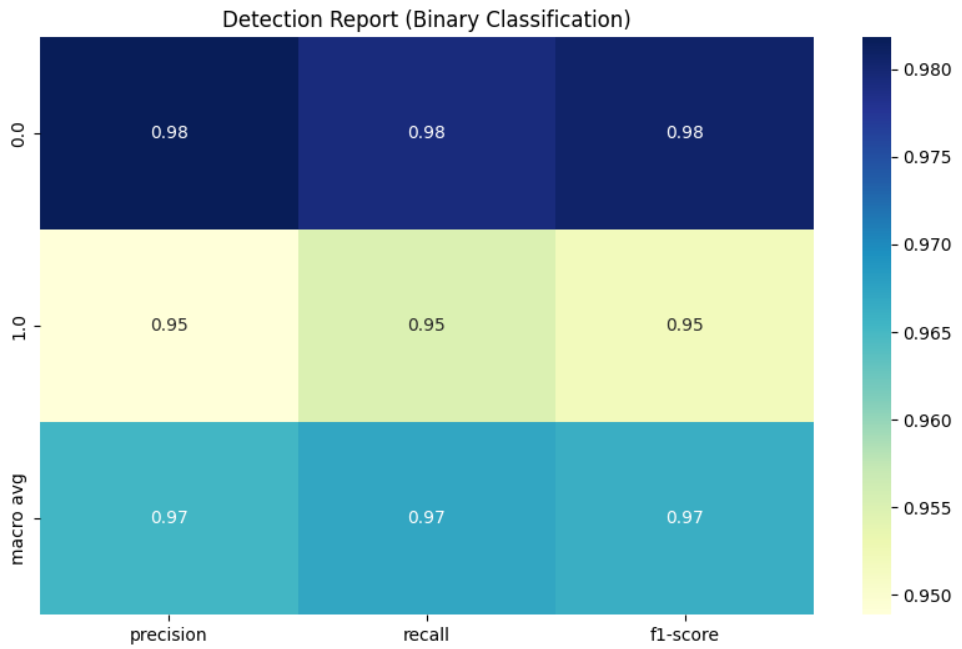


Figure 4.15: Proposed MIMO model detection results.

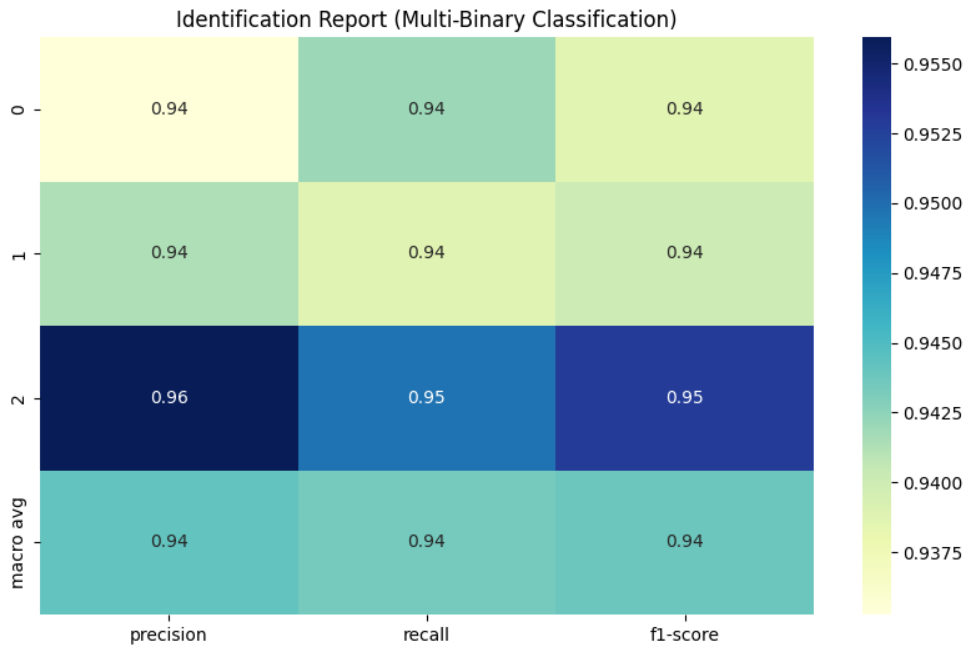


Figure 4.16: Proposed MIMO model identification results.

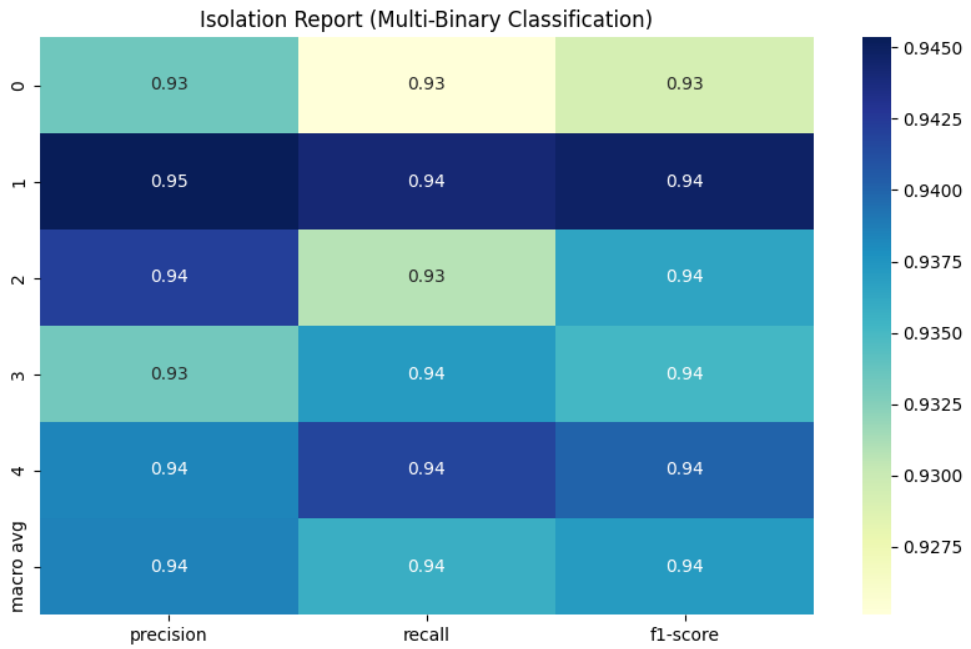


Figure 4.17: Proposed MIMO model isolation results.

Observations:

1) **Detection:** The detection performance results indicate that the proposed model demonstrates exceptional capability in detecting cyber-attacks. For the "no attack" class, the model achieves a precision and recall of 0.98, reflecting a robust ability to accurately classify non-attack scenarios with very few false positives. In contrast, the "attack" class shows precision and recall values of 0.95, suggesting that while the model is proficient in detecting actual attacks, there may be a slight increase in false negatives in this category. The macro average scores for precision, recall, and F1-score, all at 0.97, further underscore the model's balanced performance across both classes. Overall, these results highlight the model's effectiveness in maintaining high accuracy in detecting cyber-attacks, positioning it as a reliable solution for enhancing security in quadcopter networks.

2) **Identification:** In the identification task, the model demonstrated consistent performance across various attack types, with precision, recall, and F1-scores showing only minor variations. For Denial of Service (DoS) attacks, the model achieved a precision of 0.94, recall of 0.94, and F1-score of 0.94, indicating a robust capability in detecting this type of attack. Similarly, for False Data Injection (FDI) attacks, the metrics remained unchanged at 0.94 for precision, recall, and F1-score, suggesting that the model's performance is stable across these attack categories. Notably, the model performed slightly better on Replay attacks, achieving a precision of 0.96, a recall of 0.95, and an F1-score of 0.95. This trend indicates that while the model maintains a high level of performance across all

attack types, it demonstrates a marginally enhanced ability to identify Replay attacks compared to the other categories. Overall, these results reflect the model's effectiveness in identifying different cyber-attack scenarios with consistent accuracy.

3) **Isolation:** The isolation performance results reveal the model's effectiveness in identifying which specific quadcopter within the network is under attack. Each class corresponds to a different quadcopter, with Class 0 indicating an attack on the first quadcopter, Class 1 on the second, and so forth. The model achieves strong performance metrics across all classes, with precision, recall, and F1-scores averaging at 0.94. Notably, the model performs consistently well, demonstrating its ability to accurately isolate attacks across the network. While there are minor differences in performance, particularly with Class 1 achieving an F1-score of 0.94, the overall results underscore the model's robustness in ensuring effective isolation of compromised quadcopters, enhancing the overall security of the network.

Comparative analysis: This section aims to compare the performance of the proposed MIMO model in the tasks of detection, identification, and isolation of cyber-attacks with similar studies in the literature. Based on a comprehensive review of the literature, no existing study has clearly addressed cyber-attack detection in a network of quadcopters using well-defined topologies and deep learning approaches like LSTM. Most of the existing research has predominantly concentrated on the detection of cyber-attacks, leaving a gap in the literature regarding the identification and isolation of such threats, particularly within a network of quadcopters employing a specific topology. Consequently, the absence of work on detection implies that there is also a lack of studies addressing identification and isolation. As such, this study represents an advancement in the field, establishing itself as the first to utilize a novel methodology for all three tasks.

The experimental results underscore the exceptional performance of the proposed MIMO model in detecting, identifying, and isolating cyber-attacks. Specifically, the model achieved an average f1-score of approximately 97% in detection, 94% in identification, and 94% in isolation. These results highlight the model's efficacy in safeguarding networked quadcopters against cyber threats.

Detection Time: To assess the practicality of the proposed model for real-time scenarios, an experiment was conducted to determine its detection time. This metric represents the duration required by the model to process an input sequence for each quadcopter and complete the tasks of detection, identification, and isolation for the network of quadcopters. The experiment evaluated the real-time capability of the model by calculating the time needed to make predictions on 50 randomly selected test samples. All predictions were performed on a CPU to closely simulate the computational constraints typically encountered in embedded processors within quadcopters. The detection time for each sample was measured individually, and the average was computed to provide an accurate estimation of the model's real-time performance. The measured average detection time was 0.083 seconds (83 ms), highlighting the model's efficiency and its suitability for real-time applications involving a network of quadcopters.

4.5 Practical Implementation

4.5.1 Experimental Implementation for Academic Purposes

The paper [39] offers a comprehensive framework for implementing cyber-physical intrusion detection systems in UAVs. This work stands out as a valuable resource due to its detailed methodology, including the setup of a UAV testbed, application of realistic cyber-attacks, and creation of a correlated cyber-physical dataset.

The paper's systematic approach to data collection, attack simulation, and machine learning-based intrusion detection provides a practical template for addressing challenges in cyber-physical system security. The methodologies and tools outlined align closely with the requirements of this research, making it an excellent reference for guiding implementation strategies.

Type of Quadcopter: The implementation described in the paper uses the DJI Tello EDU drone, a lightweight (80g), programmable UAV. The drone's features include a front-facing 720p camera, vision positioning system, 3D infrared sensors, a ToF distance sensor, IMU, and barometer. Communication with the drone is established via a 2.4GHz 802.11n WiFi connection using the UDP protocol. These features make it particularly suitable for research, as they provide ample data points for both cyber and physical analysis [39].

Control Mechanism: The UAV is controlled using Python scripts developed with the Tello SDK. The scripts enable precise maneuverability, allowing the drone to follow a predefined path with randomized actions, such as flips or hovering. These movements were designed to simulate real-world tasks like smart farming (e.g., inspecting crops) or surveillance. Throughout its operation, the UAV continuously sends physical measurements (e.g., roll, pitch, yaw, speed, temperature) to the controller, demonstrating how control mechanisms can be integrated with real-time data collection [39].

Application of Attacks: Four types of cyber-attacks were simulated during the drone's operations, demonstrating practical methods for testing intrusion detection systems:

1) De-Authentication Attack: Exploits vulnerabilities in the IEEE 802.11 protocol to disconnect the UAV from its controller by sending spoofed de-authentication frames.
2) Replay Attack: Captures valid communication packets between the UAV and the controller, then replays them to manipulate the UAV's actions.
3) Evil Twin Attack: Establishes a rogue wireless access point mimicking the UAV's legitimate network to intercept and manipulate communications.
4) False Data Injection (FDI) Attack: Injects falsified sensor readings or control commands, causing deviations in the UAV's operations.

These attacks were implemented using tools like Aircrack-ng, Tcpdump, and Wireshark, combined with custom Python scripts [39].

Detection Methodology: The detection system in the paper was implemented as an offline analysis process, using machine learning models trained on a dataset collected during UAV flights. Data from the UAV's sensors and communication network were logged and analyzed post-flight to detect cyber-attacks. Real-time detection was not part of the implementation, as the focus was on evaluating the models' accuracy using pre-collected data [39].

Overall, paper [39] provides a clear explanation of the appropriate quadcopter selection and its control approach. Additionally, the implementation of DoS, FDI, and Replay attacks is well-documented. Similar to the current work in [39], the detection system operates offline, with no information provided about the implementation of a real-time detection system. This limitation is consistent with other similar papers in literature.

4.5.2 Real-World Implementation for Industrial Purposes

To adapt and extend the proposed work for practical use in real-world industrial applications, several critical questions must be considered. These questions encompass aspects such as data generation, computational feasibility, model scalability, and real-time detection capabilities, which are essential for transitioning from academic research to industrial deployment. The answers provided are based on the authors' understanding, supplemented by general knowledge, insights from reliable web searches, and academic references where applicable, ensuring a comprehensive and well-informed discussion.

How should the data/scenarios be generated? In real-world applications, data and scenarios should be generated by operating the quadcopter in diverse environments while applying various cyber-attacks during its operation. These scenarios should simulate realistic use cases, such as surveillance, delivery, or inspection tasks, while incorporating attacks like DoS, replay, or false data injection. The quadcopter's movement should include both structured paths and randomized actions to mimic industrial settings, ensuring the collected data captures the full spectrum of normal and attack conditions. This approach provides a robust foundation for training and validating intrusion detection systems.

How much data/scenarios are required for reliable detection module? For industrial usage, a reliable detection module requires a dataset significantly larger than academic setups. This should include 1,000–5,000 flights per class, translating to 500,000–5,000,000 samples, assuming 500–1,000 samples are generated per flight. The scale is crucial to capture diverse operational conditions, attack variations, and environmental factors, ensuring the model can generalize effectively and minimize false positives. For comparison, the CICIDS2017 dataset, widely used for network intrusion detection, contains over 3 million samples across various attack types and normal traffic, highlighting the necessity of large-scale data for robust intrusion detection systems in real-world applications [90].

How can the proposed MIMO model be extended for real-world applications? The proposed MIMO model is well-suited for networks with up to approximately 10 quadcopters. However, for larger networks, the computational complexity of the LSTM-based input heads could become a bottleneck. To address this, it is suggested to replace LSTM with a lighter feature extractor, such as a Temporal Convolutional Network (TCN) or a GRU (Gated Recurrent Unit), both of which are known to be computationally more efficient while maintaining the ability to process sequential data. Additionally, dimensionality reduction techniques, such as autoencoders or PCA (Principal Component Analysis), could be applied to further optimize the input features, making the model scalable for larger UAV networks.

What computational complexity is feasible for quadcopter? Quadcopters can handle lightweight computational complexity, typically requiring optimized algorithms with linear or near-linear complexity to balance performance and resource constraints like processing power, memory, and battery life. However, medium-capability quadcopters, commonly used in critical applications such as surveillance or industrial inspections, can support more demanding models like LSTM-based networks, which process sequential data effectively. These drones' enhanced hardware capabilities make them suitable for tasks requiring higher computational loads.

How should each sequence be labeled in real-time application? In real-time applications, the label is dynamically generated as the output of the deep learning model, meaning sequences do not require pre-labeling. Instead, the model processes incoming data streams and classifies each sequence in real time, identifying whether it corresponds to normal operation or an attack scenario. This approach eliminates the need for manual or pre-assigned labels during runtime, as the detection system itself generates the classifications.

How can the detection system detect attacks beyond the three current types? The deep learning model excels at detecting attacks it has been trained on with high accuracy. To extend its capabilities for real-world usage, the model must be trained on a comprehensive dataset that includes all known attack types at the time of training. Additionally, to detect novel or evolving attacks, the system can incorporate techniques like anomaly detection, where the model identifies deviations from normal behavior as potential threats.

What is the process for generating sequences in real-world applications? In real-world applications, data samples are continuously sent to the detection system. These samples are grouped into sequences based on a predefined sequence length, where overlapping or non-overlapping windows of data are stacked to form input sequences for the model. The detection model processes these sequences in real time, with its output indicating whether the system is operating normally or under attack. This process ensures efficient and continuous monitoring of the quadcopter’s network and physical state.

4.6 Conclusion and Contributions

This chapter introduces valuable contributions to cyber-attack detection, identification, and isolation within centralized and decentralized quadcopter networks. The proposed MIMO model supports adaptable topologies, effectively handling various network configurations. By simulating key cyber-attacks: Denial of Service (DoS), False Data Injection (FDI), and Replay, the model demonstrates robust performance across single and networked quadcopters. This work sets a foundation for future research on optimizing LSTM-based sequence generation in networked environments and expands MIMO’s applicability to broader cyber-physical systems.

Extensive experiments reveal substantial improvements in model accuracy and robustness, particularly in complex multi-quadcopter setups. Results indicate that this framework achieves high detection rates and effectively isolates attacks, with slightly better performance in centralized topologies, underscoring the model’s adaptability and resilience across configurations.

To achieve the primary objective of detecting, identifying, and isolating cyber-attacks in a network of quadcopters, a novel MIMO (Multi-Input, Multi-Output) model was developed. This model is highly

adaptable and scalable, utilizing a shared LSTM backbone to process concatenated data from multiple input heads, allowing it to accommodate varying numbers of quadcopters. The model outputs three results: detection, identification, and isolation of cyber-attacks. The detection head provides a binary output, the identification head specifies the attack type (DoS, FDI, Replay), and the isolation head identifies which quadcopters are under attack.

A dataset of 50,000 data points, representing five quadcopters, was used, with features organized for efficient data management through output reduction, simplifying the label columns from 15 to 3. Experiments optimized sequence length and LSTM blocks, with two configurations identified as ideal: one tailored for decentralized setups with 2-3 quadcopters and another for centralized setups with 4-5 quadcopters. The model was tested with sensor-only data and with combined sensor and actuator data, with the latter providing slightly better results. The MIMO model demonstrated strong performance across various network sizes and topologies.

Chapter 5

5. Conclusions and Future Work

5.1 Conclusion

This thesis presented a framework for cyber-attack detection, identification, and isolation within both single and networked quadcopter systems, addressing an area of cybersecurity in cyber-physical systems (CPS) that has gained attention with the increased use of UAVs. Using Long Short-Term Memory (LSTM) networks, this study explored a model that could perform multiple tasks essential to UAV security. Covering single-quadcopter scenarios and extending to networked configurations, the thesis aimed to provide a structured approach adaptable to real-world drone applications.

In the single-quadcopter analysis, the study tested an LSTM-based multi-output model for detecting attacks such as Denial of Service (DoS), False Data Injection (FDI), and Replay attacks. The optimized model configuration, using a 128-block LSTM with a sequence length of 40, reached a detection accuracy of 0.981 with precision at 0.943 and an F1-score of 0.927. This setup provided a balance between detection performance and computational efficiency, completing training within a practical timeframe of approximately 200 seconds. Higher configurations showed minimal performance gains but increased training times significantly, indicating that the 128-block model was suitable for UAV applications that require efficient response times.

Sequence generation preprocessing played a pivotal role in ensuring the model effectively handled time-series data for cyber-attack detection. Optimal sequence lengths allowed the model to capture temporal dependencies within the data, improving the detection of subtle patterns associated with specific attack types. This preprocessing step, by segmenting time-series data into meaningful sequences, contributed to the model's ability to distinguish between normal and anomalous behavior more accurately. Additionally, threshold-based labeling during sequence generation was tailored to refine attack sensitivity, an approach that enhanced detection precision and allowed the model to respond more reliably across diverse operational scenarios. These preprocessing techniques are critical for CPS applications, as they improve model resilience and adaptability in environments where attack timing and sequence structure are variable.

For networked quadcopters, the study developed a Multi-Input, Multi-Output (MIMO) model, tested under both centralized and decentralized network topologies. Centralized configurations showed an F1-score of 0.913 for isolation and precision of 0.926, with a consistent balance between precision and

recall. The decentralized setup, while slightly lower in certain metrics, provided flexibility for applications where distributed decision-making is valuable. Including both sensor and actuator data in network configurations also improved detection, identification, and isolation scores to 0.929, 0.918, and 0.912, respectively, across a five-quadcopter setup. Training times for this setup remained manageable, even with the increased data complexity, indicating that the model could scale to larger CPS environments.

The MIMO model further used a reduction approach to simplify outputs from 15 labels to three, which streamlined the model's training and testing phases. This reduction allowed the model to process high-dimensional data without excessive computational load. By using shared LSTM layers that capture temporal dependencies across the network, this architecture facilitated effective training times and allowed for more rapid inference without sacrificing accuracy.

In addition to methodology, the thesis included a comparative analysis with existing approaches. Although similar frameworks have achieved moderate success in cyber-attack detection, this study's approach incorporated detection, identification, and isolation into one framework, allowing a coordinated response in CPS environments. Multi-task learning capabilities were demonstrated in both single and networked quadcopter setups, suggesting potential applications of this framework in fields such as autonomous vehicles and smart grids where similar security needs exist.

The thesis also produced two datasets specifically designed for cyber-attack scenarios in UAVs: a single-quadcopter dataset with 50,000 data points labeled for detection, identification, and isolation, and a network dataset with similar data size but expanded to five quadcopters. These datasets provide resources for testing UAV security protocols in realistic conditions, supporting various network configurations and labeling approaches.

In summary, this thesis offered an approach to UAV network cybersecurity that explored both centralized and decentralized network topologies, used data reduction to manage output complexity, and examined sequence preprocessing in LSTM-based models for CPS environments. The findings suggest centralized configurations are effective for high-precision tasks, while decentralized setups can maintain resilience in distributed settings. These contributions indicate that machine learning and LSTM architectures can support CPS applications requiring resilience against cyber threats, and the MIMO model's architecture offers potential for future research in adaptable security solutions.

5.2 Future Work

One area for future work is to explore the impact of multiple concurrent attacks, particularly in networked quadcopter setups where the system may face simultaneous threats to both sensors and actuators. Although concurrent attacks are incorporated in the current network experiments, most scenarios involve a single attack type at any given time. Expanding the focus to scenarios where multiple attacks occur simultaneously would provide deeper insights into the system's resilience and help evaluate how well the model can distinguish and manage multiple, overlapping threats.

Another potential research area involves incorporating feature selection techniques. Integrating these approaches could help optimize the model by focusing on the most relevant data features, possibly enhancing detection accuracy and reducing computational load. Additionally, refining the sequence generation preprocessing approach could unlock further performance gains, not only within the

quadcopter domain but also in broader CPS applications. By fine-tuning sequence length, overlap, and labeling strategies, the LSTM-based framework could adapt to various sectors such as smart grids, autonomous vehicles, and healthcare, thus expanding its utility in securing critical infrastructure.

Expanding the range of attack types applied to the system, such as zero-dynamic or covert attacks, could also be beneficial for increasing the model's robustness. These types of attacks, often more challenging to detect, would add a layer of realism to the model's evaluation, testing its response to more subtle cyber threats. In addition, introducing disturbances or variations in flight paths for the quadcopter would better simulate real-world conditions, making the model more adaptable and responsive to varied operational settings.

For networked quadcopter applications, several enhancements can be explored. Future research could assess the performance of the Multi-Input, Multi-Output (MIMO) model under network conditions where communication links are subject to cyber-attacks, as these are common points of vulnerability in multi-quadcopter systems. Additionally, applying the MIMO model to problems like consensus and formation control, where multiple units operate in a coordinated manner, could yield insights into its effectiveness within dynamic, team-based UAV operations. Investigating leader-follower topologies could further improve network management strategies, especially in scenarios requiring one quadcopter to guide others while handling complex attack patterns.

Incorporating attention mechanisms into the MIMO model is another promising direction. By weighting input from each quadcopter, attention layers could enable the system to prioritize critical data, potentially enhancing detection, identification, and isolation accuracy. This approach may also improve efficiency in larger networks or in handling complex scenarios where the model must selectively process high volumes of data across multiple units.

Lastly, addressing data imbalances through preprocessing steps tailored to the quadcopter datasets would be valuable, especially as these imbalances may affect the model's performance in detecting rare or subtle attacks. Exploring hybrid models, combining LSTM with other architectures, could also strengthen the model's adaptability. These improvements would support the model's broader applicability and help meet the evolving cybersecurity needs of increasingly interconnected CPS environments.

References

[1] A. Eslami, M. G. Kazemi, and K. Khorasani, "Event-Based Covert Cyber-Attack in Switching Cyber-Physical Systems: Design and Detection Mechanisms," in *2024 IEEE International Systems Conference (SysCon)*, pp. 1-7, April 2024.

[2] S. Dibaji, M. Pirani, D. Flamholz, A. Annaswamy, K. Johansson, and A. Chakrabortty, "A systems and control perspective of CPS security," *Annual Reviews in Control*, vol. 47, pp. 394-411, 2019.

[3] S. Gaba, "A systematic analysis of enhancing cyber security using deep learning for cyber-physical systems," *IEEE Access*, vol. 12, pp. 6017-6035, 2024.

[4] N. Chaudhry, M. Yousaf, and M. Khan, "Security assessment of data management systems for cyber-physical system applications," *Journal of Software Evolution and Process*, vol. 32, no. 2, 2019.

[5] S. Nateghi, Y. Shtessel, and C. Edwards, "Resilient control of cyber-physical systems under sensor and actuator attacks driven by adaptive sliding mode observer," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 15, pp. 7425-7443, 2021.

[6] S. Kim, Y. Eun, and K. Park, "Stealthy sensor attack detection and real-time performance recovery for resilient CPS," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7412-7422, 2021.

[7] M. Umer, S. Sadiq, H. Karamti, R. Alhebshi, K. Alnowaiser, A. Eshmawi, et al., "Deep learning-based intrusion detection methods in cyber-physical systems: challenges and future trends," *Electronics*, vol. 11, no. 20, p. 3326, 2022.

[8] M. Taheri, K. Khorasani, I. Shames, and N. Meskin, "Cyber attack and machine induced fault detection and isolation methodologies for cyber-physical systems," 2020.

[9] S. Haider et al., “A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks,” IEEE Access, vol. 8, pp. 53972–53983, 2020.

[10] A. Mousavi, K. Aryankia, and R. R. Selmic, “A distributed FDI cyber-attack detection in discrete-time nonlinear multi-agent systems using neural networks,” European Journal of Control, vol. 66, p. 100646, Jul. 2022.

[11] K. Bitirgen and Ü. B. Filik, “A hybrid deep learning model for discrimination of physical disturbance and cyber-attack detection in smart grid,” International Journal of Critical Infrastructure Protection, vol. 40, p. 100582, Mar. 2023.

[12] Bharathi. V and C. N. S. Vinoth Kumar, “A real time health care cyber attack detection using ensemble classifier,” Computers and Electrical Engineering, vol. 101, p. 108043, Jul. 2022.

[13] X. Yin, Y. Zhu, and J. Hu, “A Subgrid-Oriented Privacy-Preserving Microservice Framework Based on Deep Neural Network for False Data Injection Attack Detection in Smart Grids,” IEEE Transactions on Industrial Informatics, vol. 18, no. 3, pp. 1957–1967, Mar. 2022.

[14] Q. Li, J. Zhang, J. Zhao, J. Ye, W. Song, and F. Li, “Adaptive Hierarchical Cyber Attack Detection and Localization in Active Distribution Systems,” IEEE Transactions on Smart Grid, vol. 13, no. 3, pp. 2369–2380, May 2022.

[15] H. Sun, M. Chen, J. Weng, Z. Liu, and G. Geng, “Anomaly Detection for In-Vehicle Network Using CNN-LSTM With Attention Mechanism,” IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 10880–10893, Oct. 2021.

[16] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, “Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks,” in 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), pp. 140–145, Jan. 2017.

[17] A. Baul, G. C. Sarker, P. K. Sadhu, V. P. Yanambaka, and A. Abdelgawad, “XTM: A Novel Transformer and LSTM-Based Model for Detection and Localization of Formally Verified FDI Attack in Smart Grid,” Electronics, vol. 12, no. 4, Art. no. 4, Jan. 2023.

[18] A. N. Jahromi, H. Karimipour, A. Dehghantanha, and K.-K. R. Choo, “Toward Detection and Attribution of Cyber-Attacks in IoT-Enabled Cyber–Physical Systems,” IEEE Internet of Things Journal, vol. 8, no. 17, pp. 13712–13722, Sep. 2021.

[19] K.-D. Lu, L. Zhou, and Z.-G. Wu, “Representation-Learning-Based CNN for Intelligent Attack Localization and Recovery of Cyber-Physical Power Systems,” IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 5, pp. 6145–6155, May 2024.

[20] V. Ravi, R. Chaganti, and M. Alazab, “Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system,” Computers and Electrical Engineering, vol. 102, p. 108156, Sep. 2022.

[21] J. Sakhnini, H. Karimipour, A. Dehghantanha, and R. M. Parizi, “Physical layer attack identification and localization in cyber–physical grid: An ensemble deep learning based approach,” Physical Communication, vol. 47, p. 101394, Aug. 2021.

[22] K. Shen, W. Yan, H. Ni, and J. Chu, “Localization of False Data Injection Attack in Smart Grids Based on SSA-CNN,” Information, vol. 14, no. 3, Art. no. 3, Mar. 2023.

[23] R. Panigrahi et al., “Intrusion detection in cyber–physical environment using hybrid Naïve Bayes—Decision Table and multi-objective evolutionary feature selection,” Computer Communications, vol. 188, pp. 133–144, Apr. 2022.

[24] A. Heidari, N. Jafari Navimipour, and M. Unal, “A Secure Intrusion Detection Platform Using Blockchain and Radial Basis Function Neural Networks for Internet of Drones,” IEEE Internet of Things Journal, vol. 10, no. 10, pp. 8445–8454, May 2023.

[25] “Cyber Attack Detection and Isolation for a Quadrotor UAV With Modified Sliding Innovation Sequences | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Oct. 17, 2024.

[26] A. Alferaidi et al., “Distributed Deep CNN-LSTM Model for Intrusion Detection Method in IoT-Based Vehicles,” Mathematical Problems in Engineering, vol. 2022, no. 1, p. 3424819, 2022.

[27] A. A. Eshmawi, M. Umer, I. Ashraf, and Y. Park, “Enhanced Machine Learning Ensemble Approach for Securing Small Unmanned Aerial Vehicles From GPS Spoofing Attacks,” IEEE Access, vol. 12, pp. 27344–27355, 2024.

[28] R. A. Ramadan, A.-H. Emara, M. Al-Sarem, and M. Elhamahmy, “Internet of Drones Intrusion Detection Using Deep Learning,” Electronics, vol. 10, no. 21, Art. no. 21, Jan. 2021.

[29] Y. Li et al., “Jamming Detection and Classification in OFDM-Based UAVs via Feature- and Spectrogram-Tailored Machine Learning,” IEEE Access, vol. 10, pp. 16859–16870, 2022.

[30] A. Gasimova, T. T. Khoei, and N. Kaabouch, “A Comparative Analysis of the Ensemble Models for Detecting GPS Spoofing attacks on UAVs,” in 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0310–0315, Jan. 2022.

[31] E. Basan, A. Basan, A. Nekrasov, C. Fidge, E. Abramov, and A. Basyuk, “A Data Normalization Technique for Detecting Cyber Attacks on UAVs,” Drones, vol. 6, no. 9, Art. no. 9, Sep. 2022.

[32] A. Ossamah, “Blockchain as a solution to Drone Cybersecurity,” in 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), pp. 1–9, Jun. 2020.

[33] T. Talaei Khoei, S. Ismail, and N. Kaabouch, “Dynamic Selection Techniques for Detecting GPS Spoofing Attacks on UAVs,” Sensors, vol. 22, no. 2, Art. no. 2, Jan. 2022.

[34] G. Aissou, S. Benouadah, H. El Alami, and N. Kaabouch, “Instance-based Supervised Machine Learning Models for Detecting GPS Spoofing Attacks on UAS,” in 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0208–0214, Jan. 2022.

[35] O. Bouhamed, O. Bouachir, M. Aloqaily, and I. A. Ridhawi, “Lightweight IDS For UAV Networks: A Periodic Deep Reinforcement Learning-based Approach,” in 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1032–1037, May 2021.

[36] J. Whelan, T. Sangarapillai, O. Minawi, A. Almehmadi, and K. El-Khatib, “Novelty-based Intrusion Detection of Sensor Attacks on Unmanned Aerial Vehicles,” in Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, in Q2SWinet ’20. New York, NY, USA: Association for Computing Machinery, pp. 23–28, Nov. 2020.

[37] Z. Baig, N. Syed, and N. Mohammad, “Securing the Smart City Airspace: Drone Cyber Attack Detection through Machine Learning,” Future Internet, vol. 14, no. 7, Art. no. 7, Jul. 2022.

[38] A. Aldaej, T. A. Ahanger, M. Atiquzzaman, I. Ullah, and M. Yousufudin, “Smart Cybersecurity Framework for IoT-Empowered Drones: Machine Learning Perspective,” Sensors, vol. 22, no. 7, Art. no. 7, Jan. 2022.

[39] S. C. Hassler, U. A. Mughal, and M. Ismail, “Cyber-Physical Intrusion Detection System for Unmanned Aerial Vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 6, pp. 6106–6117, Jun. 2024.

[40] J. Viana et al., “Deep Attention Recognition for Attack Identification in 5G UAV Scenarios: Novel Architecture and End-to-End Evaluation,” IEEE Transactions on Vehicular Technology, vol. 73, no. 1, pp. 131–146, Jan. 2024.

[41] S. Miao, Q. Pan, D. Zheng, and G. Mohi-ud-din, “Unmanned aerial vehicle intrusion detection: Deep-meta-heuristic system,” Vehicular Communications, vol. 46, p. 100726, Apr. 2024.

[42] F. Tlili, S. Ayed, and L. C. Fourati, “A New Hybrid Adaptive Deep Learning-Based Framework for UAVs Faults and Attacks Detection,” IEEE Transactions on Services Computing, vol. 16, no. 6, pp. 4128–4139, Nov. 2023.

[43] T. Hickling, N. Aouf, and P. Spencer, “Robust Adversarial Attacks Detection Based on Explainable Deep Reinforcement Learning for UAV Guidance and Planning,” IEEE Transactions on Intelligent Vehicles, vol. 8, no. 10, pp. 4381–4394, Oct. 2023.

[44] S. Wu, Y. Li, Z. Wang, Z. Tan, and Q. Pan, “A Highly Interpretable Framework for Generic Low-Cost UAV Attack Detection,” IEEE Sensors Journal, vol. 23, no. 7, pp. 7288–7300, Apr. 2023.

[45] W. Ahmad, M. A. Almaiah, A. Ali, and M. A. Al-Shareeda, “Deep Learning Based Network intrusion detection for unmanned aerial vehicle (UAV),” in 2024 7th World Conference on Computing and Communication Technologies (WCCCT), pp. 31–36, Apr. 2024.

[46] S. Wang, J. Wang, C. Su, and X. Ma, “Intelligent Detection Algorithm Against UAVs’ GPS Spoofing Attack,” in 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), pp. 382–389, Dec. 2020.

[47] C. M. Paredes, D. Martínez-Castro, V. Ibarra-Junquera, and A. González-Potes, "Detection and isolation of DoS and integrity cyber attacks in cyber-physical systems with a neural network-based architecture," *Electronics*, vol. 10, no. 18, p. 2238, 2021.

[48] R. V. Yohanandhan, R. M. Elavarasan, P. Manoharan, and L. Mihet-Popa, "Cyber-physical power system (CPPS): A review on modeling, simulation, and analysis with cyber security applications," IEEE Access, vol. 8, pp. 151019-151064, 2020.

[49] O. Kapitonov, "Cyber-physical systems in electrochemical measurements," System Technologies, vol. 4, no. 129, pp. 3-7, 2020.

[50] K. Meinke, "Learning-based testing of cyber-physical systems-of-systems: a platooning study," in Computer Performance Engineering: 14th European Workshop, EPEW 2017, Berlin, Germany, September 7-8, 2017, Proceedings 14, Springer International Publishing, pp. 135-151, 2017.

[51] G. Talmele and U. Shrawankar, "Real-time cyber-physical system for healthcare monitoring in COVID-19," International Journal of Web-Based Learning and Teaching Technologies, vol. 17, no. 5, pp. 1-10, 2022.

[52] L. Zhang, "Applying system of systems engineering approach to build complex cyber physical systems," in Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering, pp. 621-628, Springer International Publishing, 2015.

[53] A. M. Deshpande, R. Kumar, A. A. Minai, and M. Kumar, "Developmental reinforcement learning of control policy of a quadcopter UAV with thrust vectoring rotors," in Dynamic Systems and Control Conference, vol. 84287, p. V002T36A011, American Society of Mechanical Engineers, Oct. 2020.

[54] M. Jacovic, M. J. Liston, V. Pano, G. Mainland, and K. R. Dandekar, "Experimentation framework for wireless communication systems under jamming scenarios," IET Cyber-Physical Systems: Theory & Applications, vol. 7, no. 2, pp. 93-111, 2022.

[55] Y. Li, S. Zahran, Y. Zhuang, Z. Gao, Y. Luo, Z. He, and N. El-Sheimy, "IMU/magnetometer/barometer/mass-flow sensor integrated indoor quadrotor UAV localization with robust velocity updates," Remote Sensing, vol. 11, no. 7, p. 838, 2019.

[56] Y. Fan, "Flight control system simulation for quadcopter unmanned aerial vehicle (UAV) based on MATLAB Simulink," in Journal of Physics: Conference Series, vol. 2283, no. 1, p. 012011, Jun. 2022. IOP Publishing.

[57] S. Lu, "Modeling, control and design of a quadrotor platform for indoor environments," M.S. thesis, Arizona State University, 2018.

[58] S. Maggo, S. Hussain, A. Deshpande, and S. L. Patil, "Comparative study of PID and FOPID control techniques for a quadcopter," in *2022 3rd International Conference on Electrical Engineering and Informatics (ICon EEI)*, pp. 59-63, Oct. 2022.

[59] H. Arrosida and M. E. Echsony, "The design of optimal PID control method for quadcopter movement control," *International Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 1-6, 2018.

[60] D. Afidah, S. Istiqphara, and N. S. Abu, "Interface design of DJI Tello quadcopter flight control," *Journal of Fuzzy Systems and Control*, vol. 1, no. 2, pp. 49-54, 2023.

[61] A. S. Romadhon, H. Budiarto, M. Fuad, and A. Dafid, "Quadcopter navigation system with waypoint method through flight controller at ground station,".

[62] C. Pi, J. Lin, and S. Cheng, "High accuracy positioning using jet thrusters for quadcopter," in *MATEC Web of Conferences*, vol. 151, p. 04004, 2018.

[63] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface," *Journal of Neural Engineering*, vol. 10, no. 4, p. 046003, 2013.

[64] M. Rabah, A. Rohan, Y. J. Han, and S. H. Kim, "Design of fuzzy-PID controller for quadcopter trajectory-tracking," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 3, pp. 204-213, 2018.

[65] B. Shaikh, A. Nighat, and B. S. Chowdhry, "Controlling the altitude dynamics of quadcopter using robust output feedback controller," *3c Tecnología: glosas de innovación aplicadas a la pyme*, vol. 8, no. 1, pp. 384-401, 2019.

[66] P. A. Darwito, L. H. Swandana, H. Nugroho, and T. R. Biyanto, "State feedback controller design by utilizing linear quadratic regulator and backstepping control for UAV quadrotor," in *First Mandalika*

International Multi-Conference on Science and Engineering 2022, MIMSE 2022 (Mechanical and Electrical)(MIMSE-MEI-2022), Atlantis Press, pp. 89-101, Dec. 2022.

[67] C. Yuan, S. Kar, and J. Moura, "Cyber-physical attacks with control objectives," IEEE Transactions on Automatic Control, vol. 63, no. 5, pp. 1418-1425, 2018.

[68] Y. Wu, Y. Ru, Z. Shi, J. Xu, J. Liu, F. Zhang et al., "A cyber-attack detection method for load control system based on cyber and physical layer crosscheck mechanism," IET Generation, Transmission & Distribution, vol. 16, no. 14, pp. 2805-2815, 2021.

[69] K. Bernsmed, C. Frøystad, P. Meland, D. Nesheim, and Ø. Rødseth, "Visualizing cyber security risks with bow-tie diagrams," in Lecture Notes in Computer Science, Springer, pp. 38-56, 2018.

[70] J. Prashanthi, "DL based IoT energy audit analytics for detecting and identifying cyber-physical attacks," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 14, no. 03, pp. 1352-1361, 2023.

[71] H. He and J. Yan, "Cyber-physical attacks and defenses in the smart grid: A survey," IET Cyber-Physical Systems: Theory & Applications, vol. 1, no. 1, pp. 13-27, 2016.

[72] Y. Wu et al., "Optimal jamming attack scheduling of interactive channels," IEEE Access, vol. 8, pp. 2995939, 2020.

[73] A. Shahid et al., "Detection and prevention of false data injection attacks in the measurement infrastructure of smart grids," Sustainability, vol. 14, no. 11, pp. 6407, 2022.

[74] A. Hoehn and P. Zhang, "Detection of replay attacks in cyber-physical systems," in *2016 American Control Conference (ACC)*, Boston, MA, USA, pp. 290-295, July 2016.

[75] Y. Liu et al., "Zero-dynamics attacks in cyber-physical systems," IEEE Transactions on Control of Network Systems, vol. 6, no. 3, pp. 1023-1034, 2019.

[76] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," IEEE Security & Privacy, vol. 9, no. 3, pp. 49-51, 2011.

[77] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, and J. Quevedo, "Bibliographical review on cyber attacks from a control-oriented perspective," Annual Reviews in Control, vol. 48, pp. 103-128, 2019.

[78] H. Karimipour, A. Dehghantanha, R. Parizi, K. Choo, and H. Leung, "A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids," IEEE Access, vol. 7, pp. 80778-80788, 2019.

[79] M. Taheri, K. Khorasani, I. Shames, and N. Meskin, "Undetectable cyber attacks on communication links in multi-agent cyber-physical systems," arXiv, 2020.

[80] R. Amin, H. Sevil, S. Kocak, G. Francia, and P. Hoover, "The spatial analysis of malicious uniform resource locators (URLs): 2016 dataset case study," Information, vol. 12, no. 1, p. 2, 2020.

[81] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," Information, vol. 10, no. 4, p. 122, 2019.

[82] O. Maher and E. Sitnikova, "A trustworthy learning technique for securing industrial internet of things systems," JISIoT, pp. 33-48, 2021.

[83] S. Chen, Z. Wu, and P. Christofides, "A cyber-secure control-detector architecture for nonlinear processes," AIChE Journal, vol. 66, no. 5, 2020.

[84] A. Albakri, "Blockchain-assisted machine learning with hybrid metaheuristics-empowered cyber attack detection and classification model," Sustainability, vol. 15, no. 18, p. 13887, 2023.

[85] X. Bampoula, G. Siaterlis, N. Nikolakis, and K. Alexopoulos, "A deep learning model for predictive maintenance in cyber-physical production systems using LSTM autoencoders," *Sensors*, vol. 21, no. 3, p. 972, 2021.

[86] J. Colah, "Understanding LSTMs," Colah's Blog, Aug. 27, 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[87] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza, "A survey and taxonomy of loss functions in machine learning," *arXiv preprint arXiv:2301.05579*, 2023.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[89] V. Ravi, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," IEEE Access, vol. 7, pp. 41525-41550, 2019.

[90] X. Hu, "An improved algorithm for network intrusion detection based on deep residual networks," IEEE Access, vol. 12, pp. 66432-66441, 2024.

[91] H. Thanh and T. Lăng, "Evaluating effectiveness of ensemble classifiers when detecting fuzzers attacks on the UNSW-NB15 dataset," Journal of Computer Science and Cybernetics, vol. 36, no. 2, pp. 173-185, 2020.

[92] F. Alrayes, "Deep neural decision forest (DNDF): A novel approach for enhancing intrusion detection systems in network traffic analysis," Sensors, vol. 23, no. 20, p. 8362, 2023.

[93] H. Sun, H. Zhang, G. Ren, and C. Zhang, "A knowledge transfer framework for general alloy materials properties prediction," Materials, vol. 15, no. 21, p. 7442, 2022.

[94] S. Salem, H. Higa, H. Kim, H. Kobayashi, K. Oki, and T. Oki, "Assessment of chlorophyll-a algorithms considering different trophic statuses and optimal bands," Sensors, vol. 17, no. 8, p. 1746, 2017.

[95] K. Nicholls and C. Wallace, "Comparison of sparse biclustering algorithms for gene expression datasets," 2020.

[96] https://scikit-learn.org/stable/modules/model_evaluation.html#classification-report.

[97] M. Gamal, M. Elhamahmy, S. Taha, and H. Elmahdy, "Improving intrusion detection using LSTM-RNN to protect drones' networks," Egyptian Informatics Journal, vol. 27, p. 100501, 2024.

[98] A. M. Abdulghani, M. M. Abdulghani, W. L. Walters, and K. H. Abed, "Improving intrusion detection in UAV communication using an LSTM-SMOTE classification method," Journal of Cybersecurity, vol. 4, no. 4, 2022.