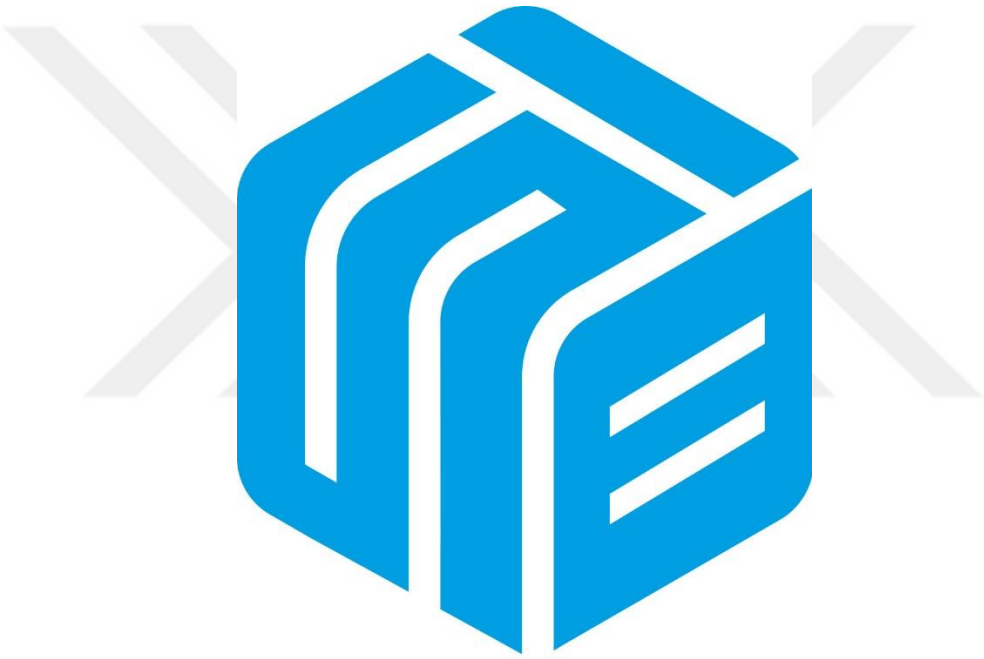


ATTACK DETECTION IN IOT SYSTEMS USING METAHEURISTIC-ENHANCED QUANTUM AND CLASSICAL MACHINE LEARNING TECHNIQUES

MUHAMMED FURKAN GÜL

MASTER'S THESIS
DEPARTMENT OF COMPUTER ENGINEERING

SIVAS UNIVERSITY OF SCIENCE AND TECHNOLOGY

INSTITUTE OF GRADUATE STUDIES

JULY 2025

ETHICAL DECLARATIONS

In this thesis, which I have prepared in accordance with the Thesis Writing Guidelines of the Graduate School of Sivas University of Science and Technology;

- I declare that the data, information, and documents presented in the thesis have been obtained in accordance with academic and ethical principles,
- That all information, documents, evaluations, and results are presented in compliance with scientific ethics and moral standards,
- That I have properly cited all the sources I benefited from in the thesis and acknowledged them accordingly,
- That I have not made any alterations to the data used,
- That the study I present in this thesis is original,

And that, otherwise, I accept in advance all legal consequences and any potential loss of rights that may arise against me.

……..
Muhammed Furkan GÜL
02/07/2025

ATTACK DETECTION IN IOT SYSTEMS USING METAHEURISTIC-ENHANCED QUANTUM AND CLASSICAL MACHINE LEARNING TECHNIQUES

(M. Sc. Thesis)

Muhammed Furkan GÜL

SİVAS UNIVERSITY OF SCIENCE AND TECHNOLOGY

INSTITUTE OF GRADUATE STUDIES

July 2025

ABSTRACT

Internet of Things (IoT) systems are composed of interconnected devices that collect and exchange data over the internet. They are widely used in domains such as healthcare, agriculture, and industry. However, their distributed nature and limited security mechanisms make them vulnerable to cyber threats. Artificial intelligence techniques, including classical machine learning (ML) and quantum machine learning (QML), are increasingly employed to enhance IoT security through real-time threat detection and prevention. This thesis proposes ML and QML models for two critical IoT domains, namely Internet of Medical Things (IoMT) and Agricultural IoT (AG-IoT). Classical ML algorithms were applied to both IoMT and AG-IoT systems. To enhance model performance, hyperparameter tuning was conducted using the Walrus Optimization Algorithm (WaOA), a modern metaheuristic designed to effectively balance exploration and exploitation in complex search spaces. Furthermore, innovative QML models were specifically developed for the AG-IoT domain to address the challenges posed by large-scale and high-dimensional data in smart agriculture environments. For feature selection, a binary version of the Starfish Optimization Algorithm (SFOA) was proposed—its first known use in this context—due to its ability to navigate complex feature spaces efficiently. To mitigate limitations of current quantum hardware, mutual information ranking was employed to reduce the feature set selected using SFOA. Subsequently, a hybrid QML architecture was developed by combining angle-encoded variational quantum circuits with classical dense layers. Experimental evaluations demonstrated that the models achieved high accuracy and strong generalization capability in both binary and multiclass intrusion detection tasks. These findings highlight the effectiveness of the proposed approach in IoT security. In the future works, the integration of federated learning and privacy-preserving techniques may be explored.

Science Code	:	92438
Key Words	:	Internet of Things, Machine Learning, Quantum Machine Learning, Optimization Algorithms, Intrusion Detection
Page Number	:	108
Advisor	:	Assoc. Prof. Dr. Halit BAKIR

METASEZGİSEL YÖNTEMLERLE GELİŞTİRİLMİŞ KUANTUM VE KLASİK MAKİNE ÖĞRENME TEKNİKLERİ KULLANILARAK IOT SİSTEMLERİNDE SALDIRI TESPİTİ

(Yüksek Lisans Tezi)

Muhammed Furkan GÜL

SİVAS BİLİM VE TEKNOLOJİ ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Temmuz 2025

ÖZET

Nesnelerin İnterneti (Internet of Things, IoT) sistemleri, internet üzerinden veri toplayan ve birbirleriyle veri alışverişi yapan bağlı cihazlardan oluşur. Bu sistemler, sağlık, tarım ve sanayi gibi birçok alanda yaygın olarak kullanılmaktadır. Ancak, dağıtık yapıları ve sınırlı güvenlik mekanizmaları, bu sistemleri siber tehditlere karşı savunmasız hale getirir. Gerçek zamanlı tehdit tespiti ve önleme amacıyla klasik makine öğrenmesi (Machine Learning, ML) ve kuantum makine öğrenmesi (Quantum Machine Learning, QML) gibi yapay zekâ teknikleri, IoT güvenliğini artırmak için giderek daha fazla kullanılmaktadır. Bu tez, IoT'nin iki kritik alanı olan Medikal Nesnelerin İnterneti (Internet of Medical Things, IoMT) ve Tarımsal IoT (Agricultural IoT, AG-IoT) için ML ve QML tabanlı modeller önermektedir. Klasik ML algoritmaları, hem IoMT hem de AG-IoT sistemlerine uygulanmıştır. Model performansını artırmak amacıyla, keşif ve sömürü arasında etkili bir denge sağlayan modern bir metasezgisel algoritma olan Walrus Optimizasyon Algoritması (Walrus Optimization Algorithm, WaOA) ile hiperparametre ayarlamaları yapılmıştır. Ayrıca, akıllı tarım ortamlarında büyük ölçekli ve yüksek boyutlu verilerin oluşturduğu zorlukları ele almak için, AG-IoT alanına özgü yenilikçi QML modelleri geliştirilmiştir. Özellik seçimi için, karmaşık özellik uzaylarında etkili gezinme yeteneği sayesinde, literatürde ilk kez kullanılan ikili bir Starfish Optimizasyon Algoritması (Starfish Optimization Algorithm, SFOA) önerilmiştir. Mevcut kuantum donanımının sınırlamalarını azaltmak için, SFOA kullanılarak seçilen özellik kümesini azaltmak amacıyla karşılıklı bilgi (mutual information) sıralaması kullanılmıştır. Sonrasında, açı kodlamalı varyasyonel kuantum devreleri ile klasik yoğun katmanların entegre edildiği hibrit bir QML mimarisi geliştirilmiştir. Deneyler, modellerin hem ikili hem de çok sınıflı saldırı tespitinde yüksek doğruluk ve güçlü genelleme yeteneği gösterdiğini ortaya koymuştur. Bulgular, önerilen yaklaşımların IoT güvenliğinde etkili olduğunu göstermektedir. Gelecekteki çalışmalarda, dağıtık öğrenme ve gizliliği koruyan tekniklerin entegrasyonu araştırılabilir.

Bilim Kodu : 92438

Anahtar Kelimeler : Nesnelerin İnterneti, Makine Öğrenmesi, Kuantum Makine Öğrenmesi, Optimizasyon Algoritmaları, Saldırı Tespiti
Sayfa Sayısı : 108

Danışman : Doç. Dr. Halit BAKIR

ACKNOWLEDGEMENTS

I would like to sincerely thank my thesis advisor, Assoc. Prof. Dr. Halit BAKIR, for his invaluable guidance, insightful feedback, and unwavering support throughout this study. His mentorship has been an exceptional learning experience and a great honour.

I am profoundly grateful to Assoc. Prof. Dr. Sibel ARSLAN, the architect of my academic journey. Her unwavering belief in my potential, inspirational guidance, and steadfast support have laid the foundation for my growth as a researcher and scholar. Nor can I forget the lasting impact of Neviye Sinem GÜRSOY, whose approach to learning fostered a sense of responsibility. I am equally thankful to my esteemed teachers, my art teacher Çağla BAŞAR and my music teacher Ahmet BAŞAR, whose invaluable guidance and encouragement enriched my artistic perspective and personal growth.

I am truly thankful to my best friend, Muharrem YÜKSEL, for his uplifting support, true friendship, and the strength he offered during difficult times. I also appreciate my colleague, Emre YÜKSEK, for his insightful input, collaborative mindset, and the motivating atmosphere he fostered throughout our study. Their contributions have been invaluable to my academic progress.

Above all, I would like to express my deepest gratitude to my beloved family—my mother, father, brothers, and sister—for their unconditional love, endless patience, and unwavering faith in me. Their constant support and sacrifices have been the silent strength behind every step of this journey.

Muhammed Furkan GÜL

Sivas, 2025

"What is point of bringing gold to a gold mine, or water to the ocean?" — Rumi

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviations	Explanation
AI	Artificial Intelligence
AG-IoT	Agriculture Internet of Things
DT	Decision Tree
DL	Deep Learning
IDS	Intrusion Detection Systems
IoMT	Internet of Medical Things
IoT	Internet of Things
LR	Logistic Regression
ML	Machine Learning
MI	Mutual Information
NB	Naive Bayes
QML	Quantum Machine Learning
RF	Random Forest
SFOA	Starfish Optimization Algorithm
WaOA	Walrus Optimization Algorithm
XGBoost	Extreme Gradient Boosting

1. INTRODUCTION

The Internet of Things (IoT) refers to physical objects integrated with sensors and software, designed to enable communication and data exchange with other systems via the internet [1], [2]. These objects, ranging from everyday household items to complex industrial machines, can collect and transmit data, enabling them to be remotely monitored, controlled, or interacted with [3]. This connectivity provides physical objects with a level of digital intelligence, making them smarter and more responsive to user needs. IoT offers significant conveniences in everyday life through smart home technologies that optimize energy consumption and adapt to user habits [4]. Wearable medical devices that support the management of chronic diseases and enable real-time monitoring and transmission of individuals' health data to healthcare professionals have significantly improved the effectiveness of medical services and enhanced the quality of life. In addition, IoT-based sensor systems in the agricultural sector facilitate the automated and efficient management of irrigation processes by monitoring soil moisture, temperature, and other environmental parameters [5]. In the industrial domain, performance data collected from production line machines can be analyzed to predict potential failures in advance, thereby enabling predictive maintenance applications that significantly reduce unplanned downtime.

In recent years, IoT has become a central component of digital transformation processes and has exhibited remarkable economic growth on a global scale. As of 2023, the global enterprise IoT market reached a value of $269 billion, marking a 15% increase compared to the previous year. Although this growth rate is slightly lower than the 18% observed in 2022, the long-term development potential of the market remains strong. For 2024, the growth rate is expected to decline to around 12%. This slowdown is primarily attributed to global economic deceleration, inflationary pressures, high interest rates, and ongoing challenges in semiconductor supply chains (see Figure 1.1) [6].

Despite this short-term slowdown, the IoT market is projected to regain momentum starting in 2025, with an expected compound annual growth rate of approximately 15% through 2030. This growth will be largely driven by software-based solutions, data analytics systems integrated with artificial intelligence (AI), and sector-specific customized IoT applications.

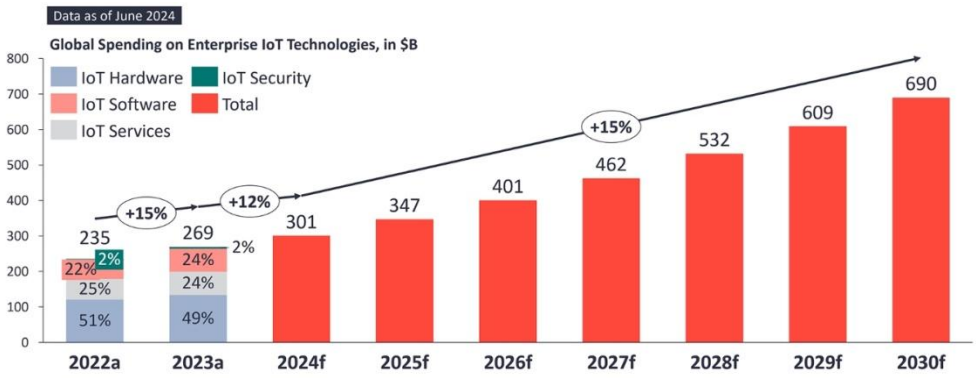


Figure 1.1. Enterprise IoT market spending forecasts (a: Actuals, f: Forecast) (This figure is retrieved from the official IoT Analytics website, based on the July 2024 market report [6].)

In addition, increasing digital infrastructure investments in emerging markets are among the other key factors supporting the global expansion of the IoT ecosystem. All these developments indicate that IoT is not merely a technological innovation, but also a strategic investment area that fosters sustainable economic growth.

The rapid and widespread adoption of IoT technologies has brought about significant security and privacy concerns. The interconnected and continuously data-exchanging nature of these systems creates new attack surfaces for malicious actors, giving rise to threats such as identity spoofing, unauthorized access, data breaches, and Denial-of-Service (DoS) attacks. According to SonicWall's 2025 Cyber Threat Report, there was a dramatic 124% increase in attacks targeting IoT devices in 2024. IP cameras, in particular, have become one of the primary targets for threat actors due to their weak security measures. In this context, the report states that more than 17 million attacks targeting IP cameras were blocked, with the monthly number of attacks ranging between 750,000 and 1.8 million [7]. These attacks are primarily carried out with the intent of manipulating surveillance systems or disrupting their functionality through operations such as Distributed Denial-of-Service (DDoS), particularly in high-value targets like government facilities, critical infrastructure, and election environments. The scale of this threat highlights that IoT-based systems should not only be viewed as tools for automation but also as potential risk factors at the level of national security.

The significant increase in attacks targeting IoT systems highlights the urgent need to prioritize security measures. In this context, AI-based approaches are playing an increasingly critical role, especially in the development of Intrusion Detection Systems

(IDS) [8]. Among these, machine learning (ML) and deep learning (DL) techniques enable modern IDS systems to analyse large-scale, heterogeneous IoT network traffic in real time, allowing them to detect both known and previously unseen attack patterns with high accuracy and adaptability [9], [10]. Within the scope of this thesis, IDS were developed by integrating AI techniques with evolutionary algorithms to address the specific security needs of various IoT extensions. The contributions of the thesis to the existing literature can be summarized as follows:

- This study utilizes two recent and domain-specific IoT datasets, namely IoMT-TrafficData and Farm-Flow. IoMT-TrafficData [11] represents the Internet of Medical Things (IoMT) domain, which involves the secure communication of medical devices and healthcare systems. Farm-Flow represents the Agriculture Internet of Things (AG-IoT) domain, focusing on smart farming technologies such as irrigation and environmental monitoring [12]. These datasets were selected to reflect realistic and sector-specific intrusion detection challenges. This ensures that the proposed models are applicable to critical and practical IoT environments.
- To the best of our knowledge, this is the first study to perform hyperparameter optimization of classical ML models using the Walrus Optimization Algorithm (WaOA), a recent nature-inspired metaheuristic designed to balance exploration and exploitation in complex search spaces. The proposed models were compared with the baseline results reported by the original authors, and they achieved superior performance across multiple evaluation metrics.
- According to the existing literature, this study presents the first known implementation of a binary version of the Starfish Optimization Algorithm (SFOA) for feature selection. This adaptation enabled efficient navigation of high-dimensional feature spaces in large-scale IoT data. The proposed method was applied in Quantum Machine Learning (QML) experiments conducted for intrusion detection in AG-IoT environments.
- To address quantum hardware limitations, Mutual Information (MI) ranking was applied after SFOA to retain only the most informative features. Using the refined subset, we proposed a novel QML architecture that integrates angle-encoded variational quantum circuits with classical dense layers in a hybrid framework. Three configurations were evaluated: a fully quantum model, a hybrid model with one dense layer, and an enhanced hybrid model with multiple dense layers.

- Extensive experiments show that our hybrid quantum–classical model consistently surpasses conventional ML baselines. The gains hold for both binary and eight-class intrusion-detection tasks, underscoring the practical advantage of QML-enhanced security solutions for AG-IoT environments.

The remainder of this thesis is organized as follows: Section 2 provides a comprehensive overview of recent studies on the IoT and its diverse application areas. It discusses the strengths and limitations of existing approaches, highlighting the research gaps this study aims to address. Section 3 outlines the materials and methods used in the study. It introduces the IoT framework, details the selected datasets, and explains the applied algorithms. Section 4 reports the experimental results and interprets them in relation to the study's goals. It compares model performances using metrics like accuracy, precision, recall, and F1-score. Section 5 presents the conclusion by summarizing the key findings and overall contributions of the study. It also outlines recommendations for future research, with a focus on improving performance and extending the applicability of the proposed approach in IoT contexts.

2. LITERATURE SURVEY

In recent years, researchers have shown growing interest in the development of IDS specifically designed for the IoT [13], [14]. This growing focus is largely driven by the increasing frequency and sophistication of cyber-attacks targeting complex and dynamic IoT environments. In parallel with general IoT frameworks, numerous studies have also focused on IDS applications in specific subdomains, including the IoMT, and the AG-IoT. Recent studies addressing these areas are as follows.

Saba et al. used a Convolutional Neural Network (CNN) based model for IDS to detect abnormal behaviours in network traffic. By training this model on NID and BoT-IoT datasets, they obtained accuracy values of 99.51% and 92.85%, respectively. In their paper, while showing that vulnerabilities in IoT networks can be effectively detected, they also draw attention to challenges such as network complexity and resource constraints [15].

Kasongo developed IDS using different DL techniques such as Recurrent Neural Networks (RNN), Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). He conducted tests on NSL-KDD and UNSW-NB15 datasets and obtained accuracy rates of 88.13% and 87.07%, respectively, thanks to feature selection with XGBoost algorithm [16].

Turukmane et al. proposed a feature selection assisted IDS called Mud Ring assisted Multilayer Support Vector Machine (M-MultiSVM) for effective detection of network-based attacks. In their paper, they used CSE-CIC-IDS 2018 and UNSW-NB15 datasets. They solved the class imbalance in the datasets with the Advanced Synthetic Minority Oversampling Technique (ASmoT). They used Modified Singular Value Decomposition (M-SvD) for feature extraction to extract basic, content and traffic features from the dataset. The selected features were optimised with Opposition-based Northern Goshawk Optimisation (ONgO). In addition to the extracted and optimized features, hyperparameter tuning was performed with the Mud Ring optimization algorithm [17].

Nandanwar and Katarya proposed a model called AttackNet by combining CNN with GRU for Industrial IoT (IIoT). The model was tested on the N_BaIoT dataset and achieved 99.75% accuracy, 99.75% precision and 99.74% recall, with an error rate of 0.0063 [18]. In another paper using the same dataset, they proposed TL-BILSTM IoT, a hybrid DL model, to detect botnet attacks in IoT environments [19]. Their proposed model combines CNN and Bidirectional Long Short-Term Memory (BiLSTM) structures with transfer learning techniques, aiming to detect BASHLITE and Mirai attacks across nine different IoT devices. According to experimental results, the model achieved 99.52% accuracy, 99.54% precision, 99.50% recall, and 99.23% F1-score, outperforming existing methods by 3.2% to 16.07%.

Hossaion and Islam proposed a reliable IDS using an ML-based ensemble model. In this paper, the feature selection process was performed by correlation analysis, Mutual Information (MI) and Principal Component Analysis (PCA) methods. The model showed an accuracy rate of over 99%, especially with the Random Forest (RF) algorithm, and high performance in other evaluation criteria [20].

Awajan proposed a novel DL-based intrusion detection system (DID) for IoT networks, designed using a six-layer fully connected neural network and implemented on a custom dataset simulating five major attack types: Blackhole, DDoS, Opportunistic Service, Sinkhole, and Wormhole. The model is protocol-independent and deployable across diverse IoT environments without requiring communication structure adjustments. Experimental results demonstrated that the proposed DID system achieved an average accuracy of 93.74%, precision of 93.71%, recall of 93.82%, and F1-score of 93.47%, highlighting its robustness and applicability for real-time intrusion detection in heterogeneous IoT networks [21].

Altulaihan et al. developed an IDS for detecting DoS attacks using the IoTID20 dataset. In their paper, they employed Correlation-Based Feature Selection (CFS) and GA for feature selection, alongside various ML methods. Among the evaluated models, Decision Tree (DT) and RF achieved the highest accuracy when trained with features selected by GA. DT was also found to be more efficient in terms of both training and testing time [22].

Racherla et al. presented a Deep-IDS for real-time attack detection on IoT nodes. Using the CIC-IDS2017 dataset, they trained an optimized LSTM network consisting of 64 hidden units. The system has been trained to detect different type of IoT attacks such as DoS, DDoS, Brute Force, Man-in-the-Middle (MITM), and Replay. A distinctive feature of their proposed method is the balance analysis between the Detection Rate (DR) and False Alarm Rate (FAR), which facilitates the real-time performance of Deep-IDS. Their method demonstrates a DR of 96.8% and an overall classification accuracy of 97.67% at a 70% threshold for the DR-FAR balance. Moreover, the Deep-IDS method exhibits superior performance with 97.67% precision, 98.17% recall, and a 97.91% F1-score [23].

Al-Ambusaidi et al. proposed ML-IDS, a system designed to ensure the security of IoT networks. They utilized the TON-IoT and UNSW-NB15 datasets to compare the performance of a modified RF algorithm with nine other ML algorithms for network attack detection. On the TON-IoT dataset, RF and other ensemble models (such as Gradient Boosting and Extreme Gradient Boosting) achieved 100% accuracy, precision, and specificity. On the UNSW-NB15 dataset, the RF algorithm demonstrated superior performance compared to many other methods, achieving an accuracy of 92.12% along with high precision and specificity values [23].

Hizal et al. suggested a two-stage DL-based IDS model for detecting DDoS attacks in IoT networks. They utilized the CICIoT2023 dataset to perform binary classification between attack traffic and normal traffic in the first stage, and then distinguished DDoS attacks from other attack types in the second stage. They identified 10 subclasses of DDoS attacks. The performance of their proposed model was evaluated using various DL methods, including Deep Neural Network (DNN), CNN, and LSTM. The best results were achieved with the LSTM-based two-stage model, which attained 94.96% accuracy and an F1-score of 92.32% [24].

Kaushik and Al-Raweshidy proposed a Teaching-Learning-Based Optimization (TLBO)-supported IDS. The system is designed to detect a wide range of attacks, including analysis, fuzzing, shellcode, worms, DoS, exploits, and backdoor intrusion attacks, while minimizing communication and device overhead. Using the UNSW-NB15 dataset, the model employs a RF classification algorithm optimized with TLBO. Experimental results revealed that the proposed TLBO-IDS outperforms state-of-the-art methods, such as the

bat algorithm and GA. Compared to the bat algorithm, TLBO-IDS achieves a 7.1% higher DR and 11.4% better accuracy, while offering 40% lower communication overhead than GA, making it an effective and robust solution for IoT security [25].

Wang et al. presented a DL-based model for detecting multi-class network attacks in IoT networks. To address the issue of data imbalance, the model uses a Conditional Tabular Generative Adversarial Network (CTGAN) to generate synthetic data for minority classes. It employs Group Convolution and Split-Attention structures to extract spatial features and BiGRU for capturing temporal features. The proposed method achieved impressive accuracy rates of 91.08%, 94.55%, and 97.47% on the UNSW-NB15, CIC-IDS2018, and CIC-IOT2023 datasets, respectively [26].

Hanafi et al. introduced a model based on Improved Binary Golden Jackal Optimization (IBGJO) and LSTM for detecting attacks in IoT networks. The proposed model was tested on the NSL-KDD and CIC-IDS2017 datasets. Feature selection was performed using the IBGJO algorithm, while LSTM networks were employed to classify attack types. To address the local optima problem of the GJO algorithm and improve convergence speed, the IBGJO was enhanced using the Opposition-Based Learning (OBL) strategy. Results demonstrated that the IBGJO-LSTM model outperformed other methods, achieving an accuracy rate of 99.6%. The model effectively analyzed network traffic, delivering a higher DR and a lower False Positive Rate (FPR) [27].

Sharma et al. developed a deep learning-based model for attack detection in IoT networks, incorporating Explainable AI (XAI) techniques to enhance the interpretability of the model. Their model was tested using the NSL-KDD and UNSW-NB15 datasets with DNN and CNN architectures. Feature selection was performed using filter-based methods, achieving high accuracy with fewer features. The model demonstrated top performance with 99.4% accuracy on the NSL-KDD dataset (using 2D-CNN) and 81% accuracy on the UNSW-NB15 dataset. XAI methods, such as LIME and SHAP, were utilized to improve the interpretability of model predictions and to analyze which features contributed most significantly to the predictions [28].

Alkhudaydi et al. presented a DL-IDS implemented on the BoT-IoT dataset. The proposed system aims to detect a broad spectrum of IoT cyberattacks—such as DDoS, DoS,

keylogging, and data theft—while addressing the issue of data imbalance using Synthetic Minority Over-sampling Technique (SMOTE). Experimental evaluations demonstrated that ensemble models like XGBoost and CatBoost significantly outperformed traditional and DL classifiers, achieving up to 98.50% accuracy [29].

Avcı and Koca proposed a hybrid IDS tailored for Building Management Systems (BMS). The system integrates the Slime Mould Optimization Algorithm (SMOA) with Artificial Neural Networks (ANN) and Support Vector Machines (SVM), and it was implemented using the CIC IoT 2022 dataset. Their model aims to accurately detect DDoS attacks in smart building environments by selecting optimal features and utilizing a two-stage detection and prediction mechanism. Experimental results showed that the proposed SMOA-based model outperformed conventional classifiers, achieving 99.19% accuracy and demonstrating superior precision, recall, and F1-score compared to traditional ML models such as K-Nearest Neighbor Classifier (KNN) and Logistic Regression (LR) [30].

Latif et al. developed a lightweight intrusion detection scheme for IIoT using a Random Neural Network (RaNN), implemented on the DS2OS dataset. The system aims to accurately detect a variety of attacks—including DoS, malicious operation, spying, scanning, and more—while maintaining low computational cost, making it suitable for resource-constrained IIoT environments. The RaNN-based model achieved a detection accuracy of 99.20%, outperforming traditional classifiers such as ANN, SVM, and DT, with significantly lower prediction time (34.51 ms) [31].

Mendonça et al. proposed a lightweight intelligent IDS for IIoT, using a Sparse Evolutionary Training (SET)-based DL model implemented on the DS2OS and CICIDS2017 datasets. The system is designed to detect various cyberattacks—including DoS, probing, malicious operations, scanning, and brute force—while significantly reducing model complexity and memory usage, making it suitable for deployment on resource-constrained devices such as Raspberry Pi. Experimental results demonstrated that the proposed SET-based model achieved up to 99.89% accuracy, with an average testing time of just 2.29 ms. It outperformed conventional models like SVM, ANN, and RF, and improved detection accuracy by 6.25% in real Industry 4.0 environments [32].

Ullah et al. introduced an HDL-IDS for the IoV, combining LSTM and GRU architectures, and implemented it using a combined DDoS dataset (CIC DoS 2016, CICIDS 2017, and CSE-CIC-IDS 2018) along with a car-hacking dataset. The proposed model addresses both inter-vehicular and intra-vehicular cyberattacks, such as DDoS, fuzzing, and spoofing, while optimizing response time and detection accuracy. Experimental results showed that HDL-IDS achieved up to 99.5% accuracy for inter-vehicular DDoS detection and 99.9% for intra-vehicular attacks. It outperformed standalone LSTM and GRU models in both accuracy and response time [33].

Ullah et al. developed a TNN-IDS tailored for MQTT-enabled IoT networks and evaluated it using the MQTT-IoT-IDS2020 dataset. The system leverages a multi-head attention mechanism to model complex patterns in sequential MQTT traffic and addresses challenges related to imbalanced data and feature selection. Experimental results revealed that TNN-IDS achieved up to 99.99% accuracy across Uni-Flow, Bi-Flow, and Packet-Flow features, outperforming traditional ML and DL models such as CNN, LSTM, GRU, and DT in terms of detection accuracy, precision, recall, and F1-score [34].

Balhareth and Ilyas proposed an optimized IDS for IoMT networks. This system combined tree-based ML algorithms and filter-based feature selection methods to classify network traffic as normal or abnormal. MI and XGBoost methods were used for feature selection, identifying the most important features. Additionally, a mathematical intersection method was applied to narrow down these features further. The model used four different tree-based ML classifiers: DT, RF, XGBoost, and CatBoost. Evaluations using the CICIDS2017 dataset showed that the model achieved an accuracy of 98.79% with a low false alarm rate (0.007 FAR) [35].

Kulshrestha and Vijay Kumar developed a ML-based IDS for IoMT networks. The system leverages several classification techniques such as Multinomial Naive Bayes (NB), LR, SVM, DT, RF, Adaptive Boosting, and others. The ToN_IoT dataset was used for training and testing, with preprocessing including imputation of missing values and feature selection using tree-based classifiers. The results showed that Adaptive Boosting performed best across multiple performance metrics like accuracy (99.18%), precision, recall, and false detection rate [36].

Alalwany et al. introduced a real-time IDS for IoMT networks using stacking ensemble DL techniques, which was evaluated using the ECU-IoHT dataset. This dataset contains data from IoMT devices subjected to different types of cyberattacks. Leveraging a Kappa Architecture framework that integrates multiple classifiers, the system processes data continuously with minimal latency. As a result, the IDS attains high detection accuracy—0.991 for binary classification and 0.993 for multi-class classification—while effectively detecting a range of cyberattacks such as DoS, Smurf, ARP spoofing, and Port Scan [37].

Berguiga et al. presented a HIDS-IoMT designed to improve security for IoMT networks, particularly against DDoS attacks. The model combines CNN for feature extraction and LSTM networks for sequence data prediction. The system was implemented using a Raspberry Pi device within a fog computing architecture, enhancing responsiveness and reducing latency. Evaluations using the IoTID20 and Edge-IIoTset datasets show the model achieving high performance, with an accuracy of 99.92%, precision of 99.91%, recall of 99.99%, and an F1-score of 99.95% [38].

Lazrek et al. proposed a hybrid anomaly IDS for securing IoMT networks, integrating Recursive Feature Elimination (RFE) with ML and DL models. The approach leverages network traffic and patient biometric data from the WUSTL-EHMS dataset for real-time anomaly detection. The RFE-based DT model achieves a training accuracy of 99% and testing accuracy of 97.85%, outperforming other models with a false acceptance rate of 0.03. The integration of Ridge regression with DL models (CNN and LSTM) further enhances detection capabilities [39].

Anjelin and Kumar developed an IDS for the IoMT using an enhanced Elephant Herding Optimization (EEHO) algorithm combined with CNN. The system focuses on reducing the dimensionality of IoMT data and optimizing feature selection for improved classification performance. Through preprocessing, the system applies one-hot encoding to transform categorical data, and the EEHO algorithm is employed to further refine features before classification. Experimental results demonstrate a 17% improvement in accuracy and a 35% reduction in time complexity compared to traditional ML models [40].

Ibrahim and Al-Wadi focused on enhancing IoMT network security by using ensemble learning-based IDS. They integrated base models such as LR, k-Nearest Neighbors (kNN),

XGBoost, RF, AdaBoost, and Multi-Layer Perceptron (MLP) within an ensemble framework. The study demonstrated that the combination of LR, AdaBoost, and XGBoost achieved outstanding results, with an accuracy of 0.9960 and a task completion time of 12.42 seconds, making it highly efficient for intrusion detection in IoMT environments [41].

Faruqui et al. introduced SafetyMed, a novel IDS for IoMT networks, which combined CNN and LSTM networks. The system was designed to detect intrusions from both image and non-image data, providing a comprehensive defense. SafetyMed achieved an average detection accuracy of 97.63% with an F1-score of 98.47%. This hybrid IDS addressed the growing security vulnerabilities in IoMT networks by offering a unique balance between FPR and DR [42].

Gupta et al. proposed a tree classifier-based network intrusion detection model for IoMT networks. The model integrates RF techniques and hyperparameter tuning to enhance the accuracy of anomaly detection. By reducing the dataset's dimensionality, the model accelerates the anomaly detection process while achieving an accuracy of 94.23%. They balanced the dataset through data augmentation to ensure sufficient representation of both normal and anomalous data. The approach effectively identifies cyber-attacks such as MITM attacks, including spoofing and data alteration [43].

Akar et al. developed the L2D2 model, a novel LSTM-based multi-class IDS for the IoMT. The L2D2 model aims to enhance security by accurately detecting a range of attacks in IoMT environments, utilizing the CICIoMT2024 dataset. The proposed system achieved a high accuracy of 98% for 19 attack classes. The model employs two LSTM layers and two dense layers, optimized using the AdamW optimizer, outperforming other ML and DL models in both binary and multi-class classification tasks. This method addresses the specific security needs of IoMT devices, offering a balance between accuracy and computational efficiency [44].

Aburasain proposed the Enhanced Black Widow Optimization with Hybrid Deep Learning Enabled Intrusion Detection (EBWO-HDLID) framework for detecting cyber threats in IoT-based smart farming. The approach incorporates Bald Eagle Search (BES) for feature selection, a hybrid DL model combining Bidirectional Gated Recurrent Unit (BiGRU) and

CNN for intrusion classification, and EBWO for hyperparameter tuning. The model was evaluated using the ToN-IoT and Edge-IIoTset datasets. It achieved an accuracy of 98.81% on the ToN-IoT dataset, while reaching 98.35% on during the testing phase [45].

Kethineni and Pradeepini presented a hybrid DL-based IDS for IoT-enabled smart farming environments. The architecture follows a three-tier design with sensing, fog, and cloud layers, using the Fused and Optimized CNN-BiGRU (FOCB) model is deployed in fog nodes for detecting DDoS attacks. The model integrates an attention mechanism within the BiGRU to enhance key feature learning and employs Wild Horse Optimization (WHO) for tuning the network's parameters. The evaluation was performed on the ToN-IoT and APA-DDoS datasets. The system achieved 99.71% accuracy on ToN-IoT and 99.35% on APA DDoS, outperforming other state-of-the-art approaches [46].

Zidi et al. developed an IDS for smart agriculture, based on a novel combination of Downsized Kernel Partial Least Squares (DKPLS) for feature extraction and reduction, and Kernel Extreme Learning Machine (KELM) for classification. The proposed system is designed to address challenges such as high feature dimensionality and multi-class classification in the AG-IoT. The model was evaluated using the X-IIoTID dataset, which includes both binary and multi-class attack scenarios. It achieved 99.92% accuracy in binary classification and up to 99.99% accuracy in multi-class classification (with 18 attack types), outperforming existing ML and DL-based IDS methods. The approach also demonstrated low computational cost, making it suitable for real-time agricultural intrusion detection [47].

Raghuvanshi et al. proposed an intrusion detection framework for IoT-enabled smart irrigation systems in agriculture. The approach utilizes the NSL-KDD dataset, which is preprocessed through symbolic-to-numeric conversion, z-score normalization, and PCA for feature extraction. Classification is performed using three ML algorithms: SVM, LR, and RF. Among these, SVM achieved the highest accuracy, exceeding 98%, whereas RF and LR achieved lower accuracies of 85% and 78%, respectively. The paper demonstrates that SVM is the most effective for detecting intrusions in smart agricultural IoT environments [48].

Zhou et al. developed CBCTL-IDS, a transfer learning-based IDS optimized with the Black Kite Algorithm (BKA) for IoT-enabled smart agriculture. The model integrates five pre-trained CNNs (MobileNet, EfficientNet, Xception, VGG19, Inception), optimized via BKA for hyperparameter tuning, and combines their outputs using a confidence averaging ensemble strategy. ToN-IoT, Edge-IIoTset, and WSN-DS datasets were used, with tabular data transformed into image inputs for CNN-based processing. The proposed system achieved high accuracy across all datasets, including 99.85% on ToN-IoT, 99.70% on WSN-DS, and 99.90% on Edge-IIoTset. Results demonstrate superior detection performance and efficiency compared to existing models, particularly in handling both balanced and imbalanced data scenarios [49].

Saadouni et al. proposed an image-based IDS for smart agriculture networks, combining transfer learning and a bio-inspired feature selection algorithm. The model uses VGG16 for feature extraction and a novel Binary Greylag Goose Optimization (BGGO) algorithm to reduce the original 25,088 image features to a more efficient 6,327, which are then classified using a RF classifier. The system is evaluated on the CICIoT2023 dataset, transformed into RGB images, and balanced using SMOTE. It achieved 99.83% accuracy for binary classification and 99.41% for multiclass, outperforming other feature selection methods like Particle Swarm Optimization (PSO), PCA, and GA. The study demonstrates the effectiveness of image-based intrusion detection in AG-IoT, with high accuracy and low FPR [50].

Li et al. introduced SELSTM, a lightweight hybrid DL-based IDS for IoT networks, integrating NSENet (a modified SENet with NonLocal, SKConv, and inverted residual modules) and LSTM to extract spatial and temporal features. The model uses Lion and Lookahead optimizers to accelerate convergence and improve generalization, while addressing class imbalance via Weighted Random Sampling. Evaluations were conducted on NSL-KDD, CIC-IDS2017, and UNSW-NB15 datasets. SELSTM achieved superior accuracy (e.g., 82.14% on UNSW-NB15) and robustness compared to models like MobileNet-LSTM, ResNet-LSTM, and ECAResNet-LSTM. Results demonstrate its effectiveness in balancing computational efficiency with high detection performance in resource-constrained IoT environments [51].

Li and Yao presented a two-stage lightweight intrusion detection model for IoT networks called CL-SKD, which integrates self-supervised contrastive learning and self-knowledge distillation to reduce reliance on labeled data and enhance generalization. In the first stage, a CNN learns feature representations via contrastive learning; in the second stage, these features are transferred to a depthwise separable CNN through distillation. The model was evaluated on five benchmark datasets: KDD CUP99, NSL-KDD, UNSW-NB15, CIC-IDS2017, and BoT-IoT, achieving up to 99.95% accuracy with significantly reduced parameters and computation. CL-SKD outperformed several state-of-the-art ML and DL models, while being highly suitable for resource-constrained IoT environments. The use of optimized loss functions and memory queue mechanisms further improved the detection accuracy and training efficiency [52].

Ghasemi and Babaie proposed a hybrid IDS for IoT networks based on SVM and Grey Wolf Optimization (GWO). In the system, SVM is used for classifying normal and anomaly patterns, while GWO optimizes the key parameters of SVM (C and gamma) and performs feature selection to improve accuracy and reduce false alarms. The model was evaluated on NSL-KDD and ToN_IoT datasets and achieved high performance across various metrics. For instance, on the NSL-KDD dataset, it reached 98% accuracy, 97% precision, 99% recall, and 98% F-score. Compared to other methods the proposed approach showed superior detection performance and lower training/testing times [53].

Hdaib et al. introduced three quantum DL-based anomaly detection frameworks for network security, leveraging quantum autoencoders (QAEs) combined with quantum one-class SVM, Quantum Random Forest (QRF), and Quantum k-Nearest Neighbors (QkNN). These frameworks aim to enhance anomaly detection performance by exploiting the synergy of quantum computing and classical DL. Implemented using benchmark datasets such as KDD99, IoT-23, and CIC IoT 23, the QAE-QkNN framework achieved the highest performance with 97.79% accuracy and 98.26% F1-score, significantly outperforming classical ML classifiers and demonstrating the potential of QML for future cybersecurity applications [54].

Kumar and Swarnkar developed QuIDS, a quantum support vector machine-based intrusion detection system (QSVC-IDS) for IoT networks, and implemented it using two benchmark datasets: Edge-IIoTset and ACI IoT. The system leverages quantum kernel

circuits and qubit-based feature encoding to classify attacks such as malware, DDoS, injection, and reconnaissance with minimal training data and high efficiency. Experimental results demonstrated that QuIDS achieved an average recall of 91.1%, precision of 84.3%, and F1-score of 86.4%, outperforming classical ML and DL methods by margins of up to 37.7% in recall and 36.9% in precision [55].

Bellante et al. proposed a theoretical evaluation framework to assess the potential advantage of QML algorithms in cybersecurity, with a specific focus on PCA-based IDS. The study analyzed how fault-tolerant quantum algorithms, such as quantum PCA and quantum clustering, can provide computational advantages over classical methods in detecting network anomalies. Using standard datasets like KDDCUP99, CICIDS2017, and DARKNET, experimental simulations demonstrated that QML methods can significantly reduce training time. At the same time, they maintain comparable detection performance, offering a promising foundation for future cybersecurity applications as quantum hardware matures [56].

Al-Hawawreh and Hossain presented a human-centered QML framework for attack detection in IoT-based Healthcare Industry 5.0, utilizing a QRF model enhanced with Local Differential Privacy (LDP) and active learning. The system integrates generative AI tools such as ChatGPT to support security analysts and improve the detection of cyber threats while preserving user data privacy. Using the WUSTL-EHMS-2020 and ICU datasets, the proposed framework achieved up to 99.90% accuracy and 99.94% F1-score. These results demonstrate high performance in detecting attacks even with privacy-preserved data, and the framework surpasses several classical ML-based intrusion detection models [57].

Kalinin and Krundyshev proposed a QML-based IDS utilizing Quantum Support Vector Machine (QSVM) and Quantum Convolutional Neural Network (QCNN), designed to process large-scale stream-based network traffic data. The authors developed a custom stream dataset with over 10 million entries and encoded it for quantum computation using the Cirq and TensorFlow Quantum frameworks. Experimental results demonstrated that both QSVM and QCNN achieved up to 98% detection accuracy, while reducing training time by more than 50% compared to traditional SVM and CNN models. These findings highlight the scalability and efficiency of QML in big data security applications [58].

Elsedimy et al. proposed a hybrid IDS for securing IoT networks, combining QSVM with an Improved Grey Wolf Optimizer (IGWO) and evaluated it using the BoT-IoT dataset. The system integrates quantum computing for enhanced classification and employs IGWO for optimizing QSVM hyperparameters, aiming to improve accuracy and reduce false positives. Experimental results demonstrated that QSVM-IGWO achieved a training accuracy of 99.11%, an F1-score of 97.78%, and outperformed traditional models like kNN, DT, LR, and RF in all evaluation metrics, including precision and recall [59].

Said developed a QSVM-based intrusion detection model for identifying DDoS attacks in smart micro-grid (SMG) environments, using the CIC-DDoS2019 dataset. The model leverages the Harrow–Hassidim–Lloyd (HHL) algorithm and IBM's Qiskit platform to map classical features into quantum state space, thereby accelerating training while maintaining high classification accuracy. Experimental results showed that the QSVM model achieved between 99.91% and 99.94% in accuracy, precision, and recall, and reduced computational time by 93% compared to classical SVM, highlighting its effectiveness and efficiency for SMG cybersecurity [60].

Aljehane et al. proposed a hybrid intrusion detection model (GSAFS-OQNN) that combines Gravitational Search Algorithm-based feature selection with an optimized Quantum Neural Network (QNN), evaluated on the UNSW-NB15 dataset. The model incorporates z-score normalization, feature reduction via GSAFS, and fine-tuning of QNN parameters using the Sandpiper Optimization (SPO) algorithm to enhance detection performance. Experimental results showed that GSAFS-OQNN achieved an average accuracy of 99.79%, precision of 98.99%, and F1-score of 98.72%. It outperformed other benchmark techniques such as GA-LR, Tabu Search–Random Forest (TS-RF), and Ensemble Automatic Feature Selection–Random Forest (EAFS-RF) in both classification performance and computational efficiency [61].

Rajawat et al. introduced a QML-based security assessment framework for the IoMT, introducing a fused semi-supervised learning model evaluated against classical and quantum approaches. The model was tested on various IoMT devices—such as heart rate monitors, insulin pumps, and pacemakers—using simulated traffic data. Experimental results showed that the proposed quantum deep learning model achieved high precision (up to 99.34%) and recall (up to 90.35%). It outperformed conventional models like

Reinforcement Learning (RL), CNN, and standard QNN in accuracy and vulnerability detection across diverse IoMT applications [62].

Makhadmeh et al. proposed a crossover-integrated Marine Predator Algorithm (MPAC) for feature selection in IDS within IoT environments, and evaluated it on eight benchmark datasets including NSL-KDD, CICIDS2017, and NF-BoT-IoT-v2. The MPAC method enhances the original MPA by incorporating single-point, two-point, and uniform crossover operators to improve exploitation and local search capability. Experimental results showed that MPAC outperformed five competitive optimizers in nearly 90% of the comparisons. It achieved the highest performance across six of the eight datasets, demonstrating robust accuracy, reduced feature dimensionality, and superior generalization for IoT-related network intrusion detection [63].

Unlike previous studies in the field of IoT security, this thesis focuses on two critical and application-specific domains: the IoMT and the AG-IoT. While much of the existing literature relies on general-purpose or outdated datasets, this work utilizes two recent, realistic, and publicly available datasets—IoMT-TrafficData and Farm-Flow—each tailored to reflect real-world conditions in their respective domains. Another key distinction of this study lies in its systematic approach to model optimization. Instead of using default parameters or manual tuning, hyperparameters were optimized using the WaOA—a recent metaheuristic technique—marking its first known application to classical ML models according to the existing literature.

In addition, this thesis presents a binary adaptation of the SFOA for feature selection, which, to the best of our knowledge, has not been previously applied. This method effectively addresses the high dimensionality challenge in the Farm-Flow dataset. To further accommodate the limitations of current quantum computing hardware, MI ranking was employed to reduce the feature space prior to model training. Furthermore, unlike previous studies that primarily examined general quantum models, this study proposes a specially designed hybrid QML architecture. By integrating variational quantum circuits with classical dense layers, the design offers a solution focused on intrusion detection in smart farming networks. This area is relatively underrepresented in the field of QML research.

3. MATERIALS AND METHODS

3.1. Internet of Things (IoT)

Many smart devices such as phones, televisions, and watches operate with internet connectivity. Thanks to advancements in mobile networks and internet infrastructure, the number and variety of such devices are rapidly increasing. This development has made data exchange not only between computers but also among various devices and objects inevitable. The interconnection of numerous items—from cars to clothing, from utility meters to sports equipment—has given rise to what is known as the IoT (see Figure 3.1).

Figure 3.1. The Internet of Things (IoT) [64]

The IoT offers numerous advantages across various domains [5], [65], [66], [67]. It enhances individuals' quality of life and contributes significantly to time efficiency and resource savings. The seamless collection of vast amounts of data facilitates informed decision-making and improves productivity, particularly in industrial settings. Moreover, IoT enables the efficient acquisition of higher-quality outputs and simplifies the monitoring of operational processes. These capabilities not only streamline business workflows but also lead to increased customer satisfaction. Despite these benefits, IoT also presents several challenges. One of the primary concerns is the risk to data confidentiality

during transmission. The likelihood of security breaches rises with increased interconnectivity, and the complexity of IoT systems can elevate the probability of system failures. Additionally, issues related to personal privacy become more pronounced, and the ever-growing volume of data necessitates substantial storage capacity. While IoT has transformative potential, it also requires careful consideration of its associated risks.

3.1.1. IoT components

There are various types of IoT devices, most of which rely on components such as controllers, sensors, and actuators to perform their functions. Sensors embedded in these devices detect environmental events, changes, and physical parameters, and transmit the collected data to the controller in real time. The controller then analyzes and processes this data. If the data volume exceeds the processing capacity of the controller, it is sent to the cloud for further processing. Once the data has been analyzed, the controller activates actuators to carry out the necessary actions. This process is continuously repeated as long as the system is operational (see Figure 3.2). The main components and their functions are shown in Table 3.1.

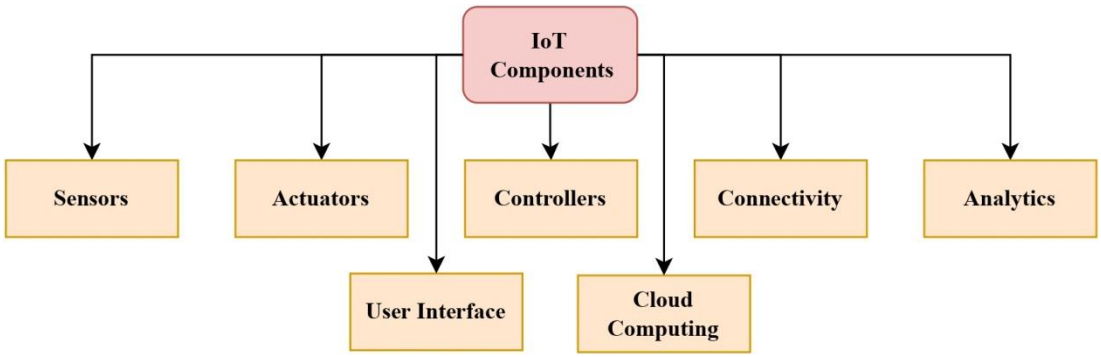


Figure 3.2. IoT components

3.1.2. Security issues in IoT systems

IoT systems are composed of integrated structures where various components operate collaboratively. However, the inherent hardware limitations of these components and the overall system architecture make IoT environments highly vulnerable from a security perspective [68].

Table 3.1. Functions of IoT system components

Component	Function
Sensors	Detect environmental data and send it to the controller for further processing.
Actuators	Perform physical actions in response to control signals, such as switching, moving, or adjusting environmental conditions.
Controllers	Receive sensor data, process it locally or forward it for analysis, and manage communication between IoT devices.
Connectivity	Enables power supply, device interconnection, and data transmission via physical or wireless communication channels.
Analytics	Processes raw sensor data into useful information to support decision-making in the IoT system.
User Interface	Enables users to interact with and manage IoT devices through applications or physical interfaces.
Cloud Computing	Provides remote storage, data processing, and application access for managing large-scale IoT data.

IoT systems are particularly exposed to cyberattacks due to their distributed nature, constrained resources, and lack of standardized security protocols [68]. The devices used in such systems are often designed to be low-cost and lightweight, which results in limited processing power and battery capacity—making the implementation of robust encryption and authentication mechanisms difficult [69]. Furthermore, the frequent use of default credentials, infrequent software updates, and inadequate data protection practices exacerbate these risks. The constant connectivity and autonomous operation of IoT devices make it easier for attackers to exploit weaknesses across different architectural layers [68]. Each layer is briefly described as follows. These are organized according to the seven-layer reference model, which includes the perception, abstraction, network, transport, computing, operation, and application layers.

1. **Perception Layer:** This layer includes physical sensing and actuating devices such as sensors, RFID tags, and controllers. Its main role is to collect data from the physical environment and convert it into digital signals for processing. Security at this layer is critical to prevent data tampering or unauthorized access at the source.

2. **Abstraction Layer:** Acts as a translator by harmonizing the diverse formats and functionalities of IoT devices. It provides a common language or protocol standard, ensuring smooth interoperability between heterogeneous components.
3. **Network Layer:** Responsible for the transmission and routing of data between devices and systems. It handles networking tasks such as data forwarding, addressing, and secure communications, making it a crucial point for defending against routing or DoS attacks.
4. **Transport Layer:** Facilitates reliable data transmission between services. It manages end-to-end communication and ensures Quality of Service (QoS) and secure delivery of data packets, playing a vital role in preventing interception or spoofing.
5. **Computing Layer:** Focuses on processing and analyzing the large volumes of data generated by IoT devices. It integrates technologies such as cloud computing, edge computing, big data, and ML to enhance performance and security.
6. **Operation Layer:** Handles service supervision, business model execution, decision-making, and monitoring. It is also responsible for ensuring QoS across layers and is vital for maintaining real-time control and system reliability.
7. **Application Layer:** Interacts directly with end-users by providing services across various IoT domains like smart homes, healthcare, agriculture, etc. It delivers the final output of the system and ensures personalized and secure service delivery.

Figures 3.3 and 3.4 present a layered taxonomy of threats and attacks in IoT. The threat taxonomy classifies potential vulnerabilities such as eavesdropping, illegal access, and data tampering that may exist due to hardware limitations, weak authentication, or protocol flaws. On the other hand, the attack taxonomy outlines concrete malicious actions—such as DoS attacks, tag cloning, MITM, and malware injection—that exploit these vulnerabilities in a targeted manner. This dual-layered classification not only facilitates a better understanding of how IoT systems can be compromised but also supports the development of layer-specific mitigation strategies. By aligning threats and attacks with their respective architectural contexts, these taxonomies serve as essential tools for designing robust and adaptable IoT security frameworks.

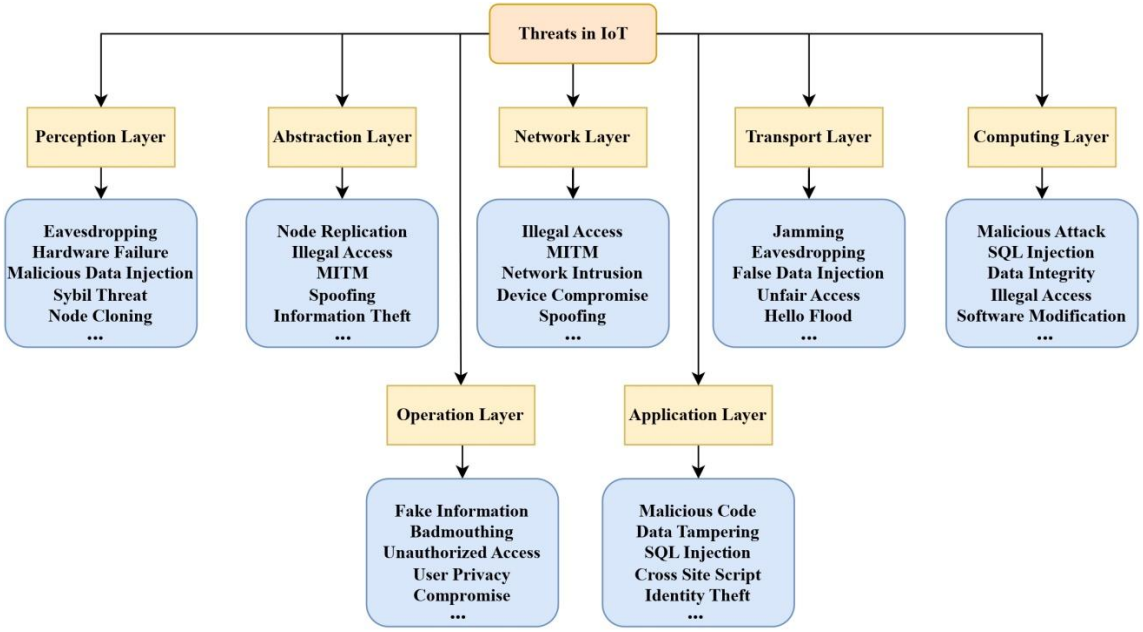


Figure 3.3. Taxonomy of IoT threats [70]

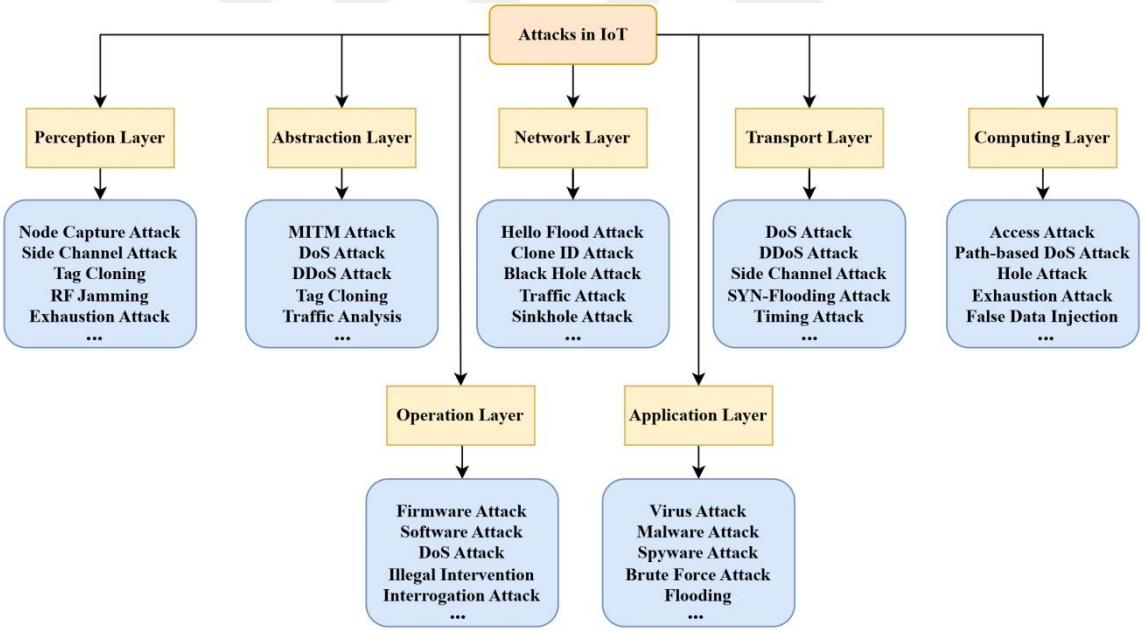


Figure 3.4. Taxonomy of IoT attacks [70]

3.1.3. Intrusion detection systems (IDS) in IoT environments

IDS in IoT networks are commonly categorized based on detection methods, deployment strategies, and system architecture [8]. Signature-based systems rely on predefined patterns to detect known threats, while anomaly-based systems focus on identifying deviations from normal behavior to uncover unknown or evolving attacks. Hybrid models, which combine both approaches, aim to maximize detection capabilities. From a deployment

perspective, IDS can be implemented at the network level or directly on host devices, with adaptations for cloud-based, virtual, and wireless environments. While these classifications provide a useful framework for understanding IDS types, they are not always sufficient in addressing the evolving challenges of modern IoT environments. Traditional classifications provide a basic understanding of IDS approaches. However, the increasing complexity and dynamic nature of IoT networks have revealed the limitations of static, rule-based methods. In response to these challenges, recent studies have increasingly focused on ML-based IDS models, which offer adaptability, enhanced accuracy, and the ability to detect emerging and previously unseen threats [71].

3.2. Metaheuristic Algorithms

Metaheuristic algorithms have emerged as robust and versatile approaches for tackling complex, nonlinear, and high-dimensional optimization problems that are often beyond the capability of conventional optimization techniques [72]. They are generally inspired by biological, physical, or social processes in nature and are guided by heuristic approaches in the pursuit of global solutions (see Figure 3.5).

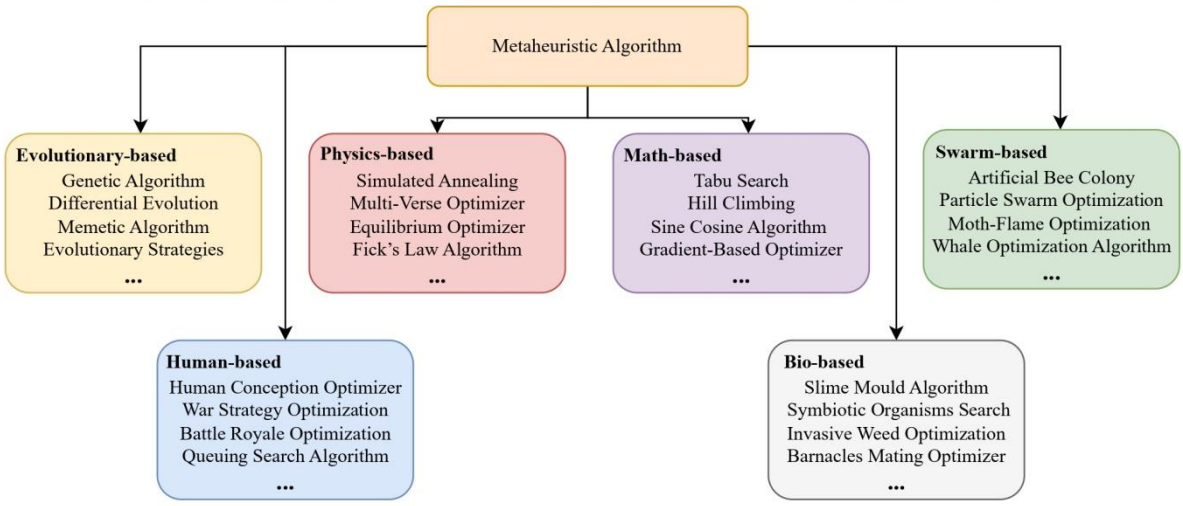


Figure 3.5. Inspirational phenomena of metaheuristics

Drawing inspiration from natural mechanisms such as adaptation, cooperation, and evolution enables these algorithms to operate in a flexible, adaptive, and efficient manner when addressing dynamic and complex problems. As a result, they are widely applied in challenging domains such as engineering design, ML, signal processing, financial modelling, route optimization, image processing, and especially in feature selection,

hyperparameter tuning, and clustering tasks [73], [74]. Popular algorithms used to solve such problems include GA [75], PSO [75], Differential Evolution (DE) [76], Ant Colony Optimization (ACO) [77], Artificial Bee Colony (ABC) [78], GWO [79], Whale Optimization Algorithm (WOA) [80], and Sine Cosine Algorithm (SCA) [81]. On the other hand, more recent algorithms such as the Puma Optimizer (PO) [82], Greylag Goose Optimization (GGO) [83], Crayfish Optimization Algorithm (COA) [84], Arithmetic Optimization Algorithm (AOA) [84], and Marine Predator Algorithm (MPA) [85] have demonstrated effective performance in solving various types of complex problems.

We also utilized two recently proposed metaheuristic algorithms: SFOA and WaOA. SFOA employs a powerful hybrid search mechanism that combines five-dimensional and unidimensional exploration strategies [86]. This design enhances the algorithm's global search diversity. In the exploitation phase, it mimics the preying and regeneration behaviors of starfish, which helps ensure stable convergence and improve solution accuracy. Similarly, WaOA is inspired by walrus behaviors such as feeding, migration, and predator evasion [87]. It operates through three distinct phases that effectively balance exploration and exploitation. This balance helps prevent premature convergence and enables WaOA to deliver competitive performance on both benchmark functions and real-world optimization problems. Leveraging these strengths the utilized metaheuristics were applied for both hyperparameter tuning and feature selection and their effective search capabilities contributed to improved overall model performance. Details of these algorithms are provided in the following subsections.

3.2.1. Starfish optimization algorithm (SFOA)

The SFOA is a recently proposed bio-inspired metaheuristic algorithm that mimics the natural behaviors of starfish—such as exploration, preying, and regeneration—to solve complex global optimization problems efficiently and effectively [86]. The mathematical framework of the algorithm is described below.

Exploration

The SFOA replicates the starfish's search behavior by utilizing the movement of its five arms, each ending with eyes. In this phase, a novel search pattern is introduced, combining a five-dimensional search strategy when D is greater than five, and a unidimensional

search strategy when D is equal to or less than five, tailored to different optimization scenarios. The dimension threshold is defined with reference to the five arms (or eyes) of a starfish. When the dimension of the optimization problem exceeds five, the search space becomes extensive, requiring the starfish to utilize all five arms to explore the environment. The arms of the starfish need to know the best position among the search agents in order to guide their movement. Eq. 3.1 provides the mathematical model for this phase.

$$\begin{aligned} Y_{i,p}^{T} &= X_{i,p}^{T} + a_1\left(X_{\text{best},p}^{T} - X_{i,p}^{T}\right).\cos\,\theta, \quad r \le 0.5 \\ Y_{i,p}^{T} &= X_{i,p}^{T} - a_1\left(X_{\text{best},p}^{T} - X_{i,p}^{T}\right).\sin\,\theta, \quad r > 0.5 \end{aligned} \tag{3.1}$$

where $X_{i,p}^{T}$ and $Y_{i,p}^{T}$ represent the current and new positions, respectively, of the ith starfish in dimension p. $X_{\text{best},p}^{T}$ denotes the pth dimension of the current best position. The dimension p is randomly chosen from five dimensions of the total D. r is a random number in (0,1), and the parameters a_1 and θ are defined in Eqs. 3.2-3.3, respectively.

$$a_1 = (2r - 1)\pi \tag{3.2}$$

$$\theta = \frac{\pi}{2} \cdot \frac{T}{T_{\max}} \tag{3.3}$$

where T and $T_{\max}$ denote the current and maximum number of iterations, respectively. The sine and cosine indicate that the arms of the starfish may rotate either left or right with equal probability in order to approach food sources. a_1 is randomly selected for each candidate and iteration to update the position, while the angle θ, which belongs to the interval $[0, \pi/2]$, varies depending on the current iteration. These two parameters are used to measure the influence of the distance between the current best position and the current position in the selected updating dimension. For optimization problems where $D > 5$, the five-dimensional search strategy defined in Eq. 3.1 is applied to update only five dimensions of the position vector. This approach helps maintain the search capability and improves efficiency compared to the full vector-based search. If the updated position exceeds the bounds of the design variables, the arms tend to remain in their previous positions rather than moving to the new one. Eq. 3.4 provides the mathematical model representing this behavior. In this expression, $u_{b,p}$ and $l_{b,p}$ represent the upper and lower bounds of the design variables respectively, and p denotes the updated dimension.

$$X_{i,p}^{T+1} = \begin{cases} Y_{i,p}^{T}, & l_{b,p} \le Y_{i,p}^{T} \le u_{b,p} \\ X_{i,p}^{T}, & \text{otherwise.} \end{cases} \tag{3.4}$$

If the dimensionality of the optimization problem is less than or equal to 5 ($D \le 5$), the exploration phase adopts a unidimensional search strategy to update the position. Only a single arm of the starfish is activated to search for the food source, relying on location information obtained from other individuals. The updated position is given in Eq. 3.5.

$$Y_{i,q}^{T} = E_t X_{i,p}^{T} + A_1\left(X_{k_1,p}^{T} - X_{i,p}^{T}\right) + A_2\left(X_{k_2,p}^{T} - X_{i,p}^{T}\right) \tag{3.5}$$

where $X_{k_2,p}^{T}$ and $X_{k_1,p}^{T}$ denote the p-dimensional positions of two randomly selected starfish, respectively. A_1 and A_2 are random values selected from the range (-1, 1). E_t represents the energy of the starfish and is calculated in Eq. 3.6.

$$E_t = \frac{T_{\max} - T}{T_{\max}} \cos\ \theta \tag{3.6}$$

where θ is computed using Eq. 3.3. As with the previous update strategy, if the newly obtained position of a starfish falls outside the boundaries of the design variables, the starfish is more likely to remain at its previous position rather than move to the updated one.

Exploitation

Exploitation phase incorporates two update strategies inspired by regeneration and preying behaviors to search for global solutions. To model the preying behavior of starfish, SFOA adopts a parallel two-directional search strategy that utilizes the positional information of other starfish along with the current best position in the population. Five distances are calculated between the current best position and other starfish. Subsequently, two of these distances are randomly selected and used to update the position of each starfish accordingly. The parallel two-directional search strategy is employed throughout this process. Eq. 3.7 provides the formulation for calculating these distances.

$$d_m = \left(X_{\text{best}}^{T} - X_{m_p}^{T}\right), m = 1, \ldots, 5 \tag{3.7}$$

where the five distances calculated between the global best position and other starfish are denoted by d_m, while five randomly selected starfish are represented by m_p. Eq. 3.8 defines the position update rule during the preying phase.

$$Y_i^T = X_i^T + r_1 d_{m1} + r_2 d_{m2} \tag{3.8}$$

where d_{m1} and d_{m2} are two distances randomly selected from the set d_m, which contains the calculated distances between the global best position and other starfish. The coefficients r_1 and r_2 are randomly generated numbers between 0 and 1. In the parallel two-directional search strategy, starfish candidates move either toward or away from a guiding solution, ensuring equal chances to escape local optima.

Starfish are particularly vulnerable to predators due to their slow movement during predation. When threatened, a starfish may detach an arm to escape capture. The regeneration phase, driven by this biological behavior, is applied only to the last starfish in the population (that is, $i = N$). Since regeneration in nature takes several months, the movement speed in this phase is considered extremely slow. Eq. 3.9 models the position update rule in the regeneration phase.

$$Y_i^T = \exp\left(-\frac{T \times N}{T_{max}}\right) X_i^T \tag{3.9}$$

where T and T_{max} represent the current iteration and the maximum number of iterations, respectively, while N denotes the size of the population. If the position obtained from Eq. 3.8 or Eq. 3.9 exceeds the boundaries of the design variables, it can be adjusted as follows. Figure 3.6 illustrates the overall flowchart of the SFOA.

$$X_i^{T+1} = \begin{cases} Y_i^T, & \text{if } l_b \le Y_i^T \le u_b \\ l_b, & \text{if } Y_i^T < l_b \\ u_b, & \text{if } Y_i^T > u_b \end{cases} \tag{3.10}$$

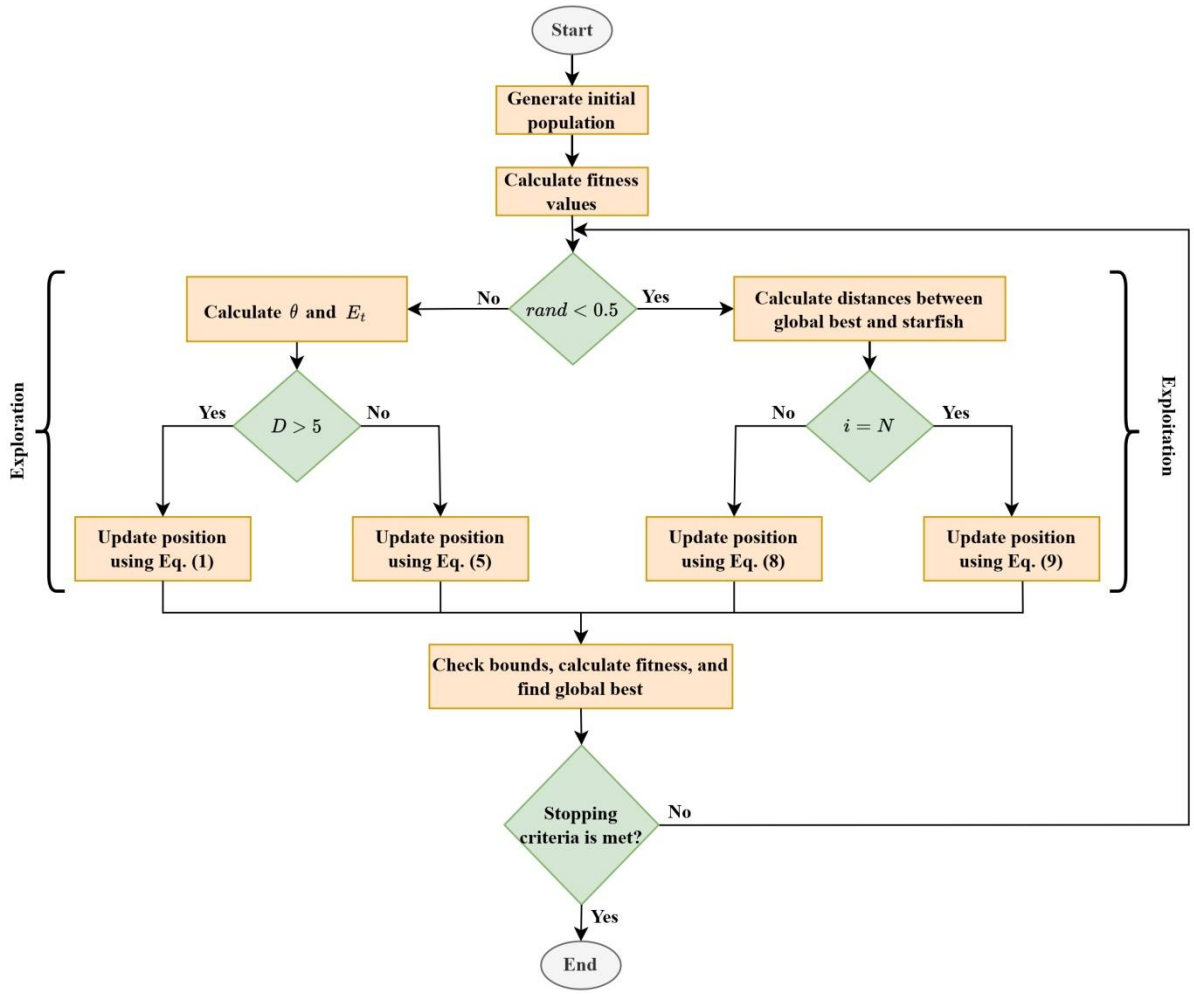


Figure 3.6. The flowchart of the SFOA [86]

3.2.2. Walrus optimization algorithm (WaOA)

The WaOA is proposed based on the natural behaviors of walruses, particularly their feeding, migration, and predator evasion/fighting strategies [87]. The algorithm offers a three-phase structure for effectively exploring a vast solution space in optimization problems: the first phase is exploration (feeding strategy), the second phase is migration, and the third phase is local search (predator evasion and fighting/exploitation). Mathematical models for these phases are as follows:

Exploration

In WaOA's foraging phase, the walrus with the longest tusks leads the group. Since tusk length is analogous to a candidate solution's objective function quality, the walrus with the

highest fitness value is regarded as the strongest. This feeding behavior drives the walruses to explore various regions of the search space, thereby enhancing WaOA's global exploration capability. The position update is mathematically modeled in Eqs. 3.11 and 3.12: Eq. 11 generates new positions, and if a newly generated position yields a superior objective function value compared to the previous position, the previous position is replaced, as formalized in Eq. 12.

$$x_{i,j}^{p_1} = x_{i,j} + \text{rand}_{i,j} \cdot \left(SW_j - I_{i,j} \cdot x_{i,j}\right) \tag{3.11}$$

$$X_i = \begin{cases} X_i^{p_1}, & F_i^{p_1} < F_i \\ X_i, & \text{else.} \end{cases} \tag{3.12}$$

The new position of the ith walrus, denoted as $X_i^{p_1}$, is updated based on its individual dimensions $x_{i,j}^{p_1}$. The objective function value for this position is represented as $F_i^{p_1}$. Random values $\text{rand}_{i,j}$ are generated within the range [0,1], while SW refers to the best candidate solution, recognized as the strongest walrus. $I_{i,j}$ are randomly selected integers that can take values of either 1 or 2. Assigning $I_{i,j} = 2$ enhances the exploration capability of the algorithm by introducing larger positional changes in the walruses, whereas $I_{i,j} = 1$ maintains a standard movement pattern. These mechanisms contribute to the global search process, aiding in escaping local optima and improving the discovery of the true optimal region within the problem-solving space.

Migration

Walruses begin to migrate as the weather warms. These migration processes are used in WaOA to discover suitable regions in the search space. Their behaviors are mathematically modeled by Eqs. 3.13 and 3.14. This modeling shows that each walrus migrates to the position of another walrus randomly selected within the search space. Accordingly, while the proposed new position is given by Eq. 3.13, Eq. 3.14 indicates that if the new position yields a better value of the objective function, it replaces the current position.

$$x_{i,j}^{p_2} = \begin{cases} x_{i,j} + \text{rand}_{i,j} \cdot \left(x_{k,j} - I_{i,j} \cdot x_{ij}\right), & \text{if } F_k < F_i, \\ x_{i,j} + \text{rand}_{i,j} \cdot \left(x_{i,j} - x_{k,j}\right), & \text{else.} \end{cases} \tag{3.13}$$

$$X_i = \begin{cases} X_i^{p_2}, & \text{if } F_i^{p_2} < F_i \\ X_i, & \text{else.} \end{cases} \tag{3.14}$$

where $X_i^{p_2}$denotes the newly generated position for the ith walrus during the second phase, with $x_{i,j}^{p_2}$ representing its jth dimension and $F_i^{p_2}$ indicating its objective function value. Furthermore, X_k $(k \in \{1,2,\ldots,N\}, k \neq i)$ refers to the location of the walrus selected for the ith walrus to migrate toward, where $x_{k,j}$ is the jth dimension of that walrus, and F_k is its objective function value.

Exploitation

Walruses change their positions in the vicinity of their current location to escape from and fight off predators. This behavioral style enhances the exploitation capability among candidate solutions in WaOA. The process takes place around each walrus's position. A walrus's movement is defined within a "walrus-centered" neighborhood region of a certain radius. In the initial stages of the algorithm, global search is prioritized to locate the best points in the search space. The radius of this neighborhood is initially set to the highest value and then decreases as the algorithm proceeds. Therefore, local lower and upper bounds are employed at this stage of WaOA to adjust the radius throughout the iterations of the algorithm. To implement this mechanism in WaOA, a neighborhood is defined around each walrus. Within this neighborhood, a new position is first generated randomly using Eqs. 3.15-3.16. If the objective function value improves, this new position replaces the previous one according to Eq. 3.17.

$$x_{i,j}^{p_3} = x_{i,j} + \left(lb_{\text{local},j}^t + \left(ub_{\text{local},j}^t - \text{rand} \cdot lb_{\text{local},j}^t\right)\right) \tag{3.15}$$

$$\text{Local bounds: } lb_{\text{local},j}^t = \frac{lb_j}{t}, ub_{\text{local},j}^t = \frac{ub_j}{t} \tag{3.16}$$

$$X_i = \begin{cases} X_i^{p_3}, & \text{if } F_i^{p_3} < F_i, \\ X_i, & \text{else.} \end{cases} \tag{3.17}$$

where $X_i^{p_3}$ is the newly generated position of the ith walrus during the third phase—its jth dimension given by $x_{i,j}^{p_3}$ and its objective function value by $F_i^{p_3}$. The variable t serves as the iteration counter, while ub_j and lb_j specify the upper and lower bounds of the jth variable, respectively. Likewise, $ub_{\text{local},j}^t$ and $lb_{\text{local},j}^t$ represent the local upper and lower

bounds for the jth dimension, respectively, enabling a focused search in the neighborhood of candidate solutions. Figure 3.7 illustrates the overall flowchart of the WaOA.

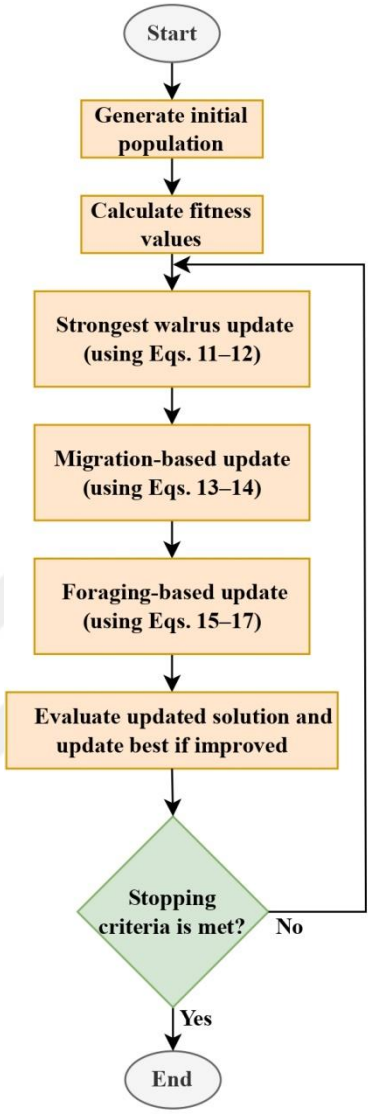


Figure 3.7. The flowchart of the WaOA [87]

3.2.3. Solution representation for feature selection

Metaheuristic algorithms provide a powerful approach for selecting optimal feature subsets in high-dimensional datasets. These algorithms aim to eliminate irrelevant or redundant features and optimize model performance through their global search capabilities. Solutions are typically represented by a binary vector, where each feature is assigned a value of "1" if selected or "0" if not (**see** Figure 3.8). This representation allows the algorithms to effectively explore the solution space and identify meaningful features to enhance model performance. To perform feature selection, metaheuristics convert

continuous solution space values into binary format using a sigmoid-based transformation function (Eqs. 3.18-3.19).

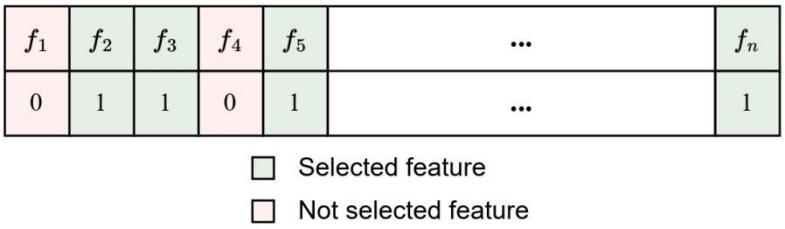


Figure 3.8. Solution illustration as a binary vector

$$x_{sigmoid} = \frac{1}{1+e^{-x}} \tag{3.18}$$

$$x_{binary} = \begin{cases} 1, & \text{if } x_{sigmoid} \geq rand \\ 0, & \text{if } x_{sigmoid} < rand \end{cases} \tag{3.19}$$

where x represents a value in the continuous solution space. The transformed binary solution obtained through feature selection is denoted as x_{binary}. A randomly generated value $rand \in [0,1]$ serves as the threshold for binarization. Figure 3.9 illustrates the sigmoid-based transformation function. This function allows a smooth probabilistic thresholding of continuous values and improves the consistency of binary solution generation.

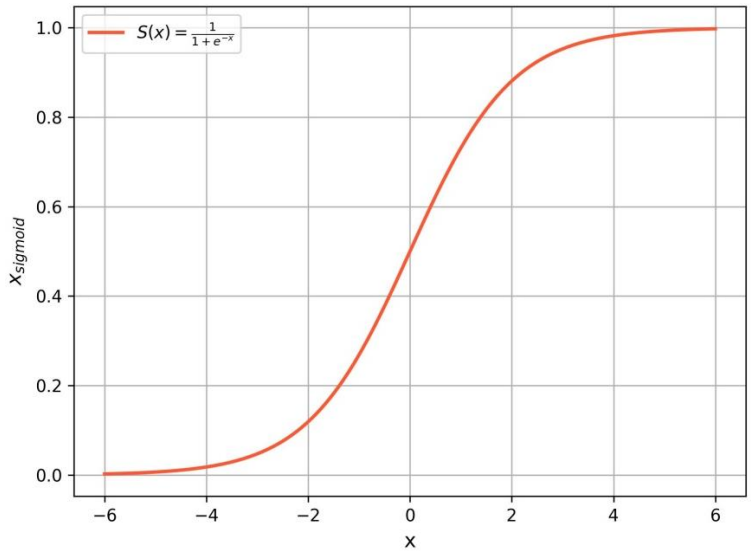


Figure 3.9. Sigmoid-based binary transformation function

The fitness function plays a critical role in the feature selection process by ensuring the identification of the most optimal features, while enhancing both the accuracy and efficiency of ML algorithms. The formulation of the fitness function is presented in Eqs. 3.20-3.21.

$$Error_{rate} = 1 - Accuracy \tag{3.20}$$

$$f_{fitness} = a * Error_{rate} + \beta * \frac{F_s}{F_a} \tag{3.21}$$

where, $Error_{rate}$ is determined based on the classification accuracy obtained using the kNN algorithm with $k = 5$. F_a and F_s represent the total number of features in the dataset and the number of selected features, respectively. a and β are weight coefficients in the fitness function that determine the importance of the two main objectives: classification accuracy and the number of selected features, with their values set to 0.90 and 0.10, respectively. The overall procedure is outlined in the pseudocode of Algorithm 1.

Algorithm 1: Metaheuristic-based wrapper feature selection using SFOA and kNN

Input: Dataset **D** with m features and class labels, **N**: population size, **T**: maximum number of iterations, **k**: number of neighbors in kNN, **α** and **β**: weight coefficients for accuracy and feature count, metaheuristic-specific parameters

Output: Best selected feature subset, number of selected features, convergence curve of fitness values

1. Initialize population:
2. **for each** individual **i** = **1** to **N**:
3. - Generate a continuous vector $\boldsymbol{X_i}$ of length m (each element represents a feature selection probability)
4. Evaluate initial fitness:
5. **for each** solution $\boldsymbol{X_i}$**:**
6. **a.** Apply sigmoid-based binary conversion using Eq. 3.19
7. **b.** Select features indicated by the resulting binary vector
8. **c.** Train a kNN classifier on selected features
9. **d.** Calculate classification accuracy
10. **e.** Calculate fitness using Eq. 3.21
11. Memorize best solutions identified so far:
12. **-** Memorize $\boldsymbol{P_i}$ **(**individual best**)** and $\boldsymbol{G}$ (global best)
13. For iteration **t** = **1** to **T**:
14. **a.** Update population using metaheuristic-specific strategies:
15. - Apply exploration, exploitation, and regeneration strategies
16. **b**. Evaluate each updated solution**:**
17. **i.** Apply sigmoid-based binary conversion using Eq. 3.19
18. **ii.** Select features and evaluate kNN performance
19. **iii.** Calculate fitness as in **Step 4**
20. **iv.** Update $\boldsymbol{P_i}$ and $\boldsymbol{G}$ if improved
21. **c.** Memorize the global best fitness value in the convergence curve
22. After T iterations:
23. - Convert the final global best solution into binary format and extract the indices of the selected features
24. Return selected feature indices, their count, and the convergence curve**.**

3.2.4. Hyperparameter tuning with metaheuristic optimization

To enhance the performance of the ML models, hyperparameter tuning was conducted using a metaheuristic optimization approach. Instead of relying on conventional methods such as grid search or random search, the previously described WaOA was employed due to its efficiency in navigating high-dimensional and complex search spaces. Each individual in the WaOA population represented a unique set of hyperparameter values, and fitness was evaluated based on classification accuracy on the testing set. Table 3.2 presents the hyperparameter search space for each ML algorithm, as described in the next section.

Table 3.2. Parameter space/range of ML algorithms

Algorithm	Parameter	Range/Space
DT	criterion	[“gini”, “entropy”]
	max_depth	[1, 20]
	min_samples_split	[2, 10]
	min_samples_leaf	[1, 5]
LR	C	[0.01, 10]
	max_iter	[5000, 10000]
NB	var_smoothing	[1e-10, 1e-1]
RF	n_estimators	[10, 500]
	max_depth	[1, 20]
	min_samples_split	[2, 10]
	min_samples_leaf	[1, 10]
	max_features	[0.1, 1]
	bootstrap	[“False”, “True”]
XGBoost	n_estimators	[10, 500]
	max_depth	[1, 20]
	learning_rate	[0.01, 0.5]
	subsample	[0.5, 1]
	colsample_bytree	[0.5, 1]
	gamma	[0, 5]

In the classical ML experiments, the hyperparameter search space provided in Table 3.2 was consistently applied across both the IoMT and AG-IoT datasets. To explore the defined ranges effectively, values for parameters such as n_estimators, max_iter, and others were dynamically adjusted by randomly searching within the specified intervals. This randomized search process contributed to a more flexible tuning mechanism, enabling the discovery of well-performing parameter sets tailored to each dataset.

3.3. Machine Learning

ML is a fundamental branch of AI that enables systems to learn patterns from data and make predictions or decisions without being explicitly programmed [9]. ML techniques are broadly categorized into three main types based on the nature of the learning process and the availability of labeled data: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning, the model is trained on a labeled dataset where the input-output relationships are known [88]. This approach is commonly applied to tasks such as classification and regression. Unsupervised learning, on the other hand, deals with unlabeled data [89]. It aims to discover hidden structures or groupings within the dataset, typically through clustering or association techniques. Semi-supervised learning integrates elements from both supervised and unsupervised learning paradigms [90]. It leverages a small amount of labeled data alongside a large quantity of unlabeled data, often enhancing performance in real-world scenarios where labeling is expensive or time-consuming. Figure 3.10 illustrates the general taxonomy of ML algorithms, highlighting the types of learning. It also presents the nature of the target variables and representative algorithms commonly used within each category. The ML algorithms utilized in this thesis are discussed in detail in the corresponding subsection.

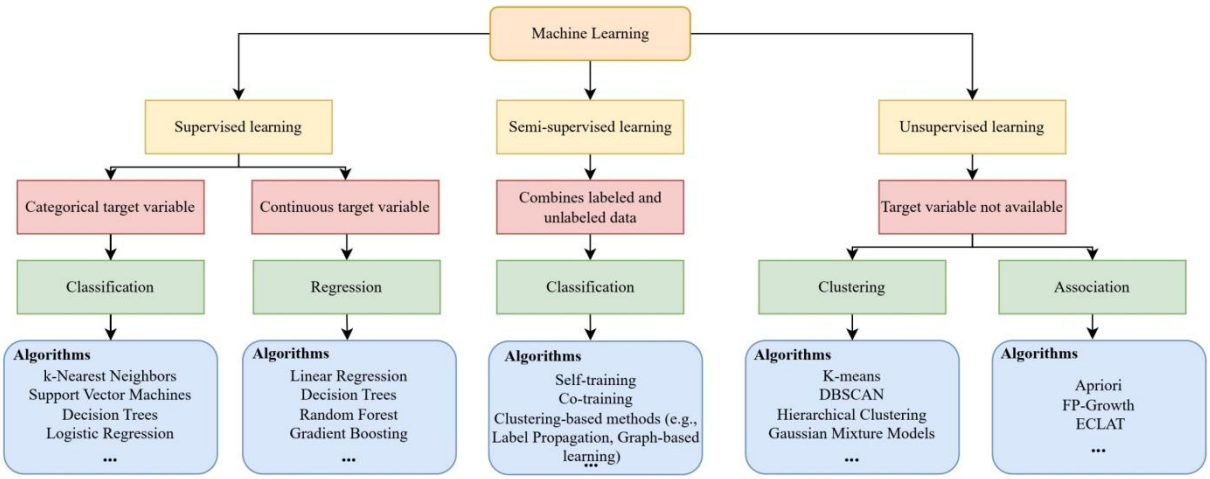


Figure 3.10. The general taxonomy of ML algorithms

3.3.1. Decision tree (DT)

DT is a widely adopted ML algorithm within the family of tree-based methods, suitable for both classification and regression tasks [91]. The DT structure is composed of internal decision nodes and terminal leaf nodes, which are constructed based on input features and learning objectives. Each decision node represents a conditional test on a specific feature and leads to branches corresponding to possible outcomes. Leaf nodes, positioned at the end of each branch, provide the final output in the form of a class label or a predicted value. The general workflow and architecture of a DT model are illustrated in Figure 3.11.

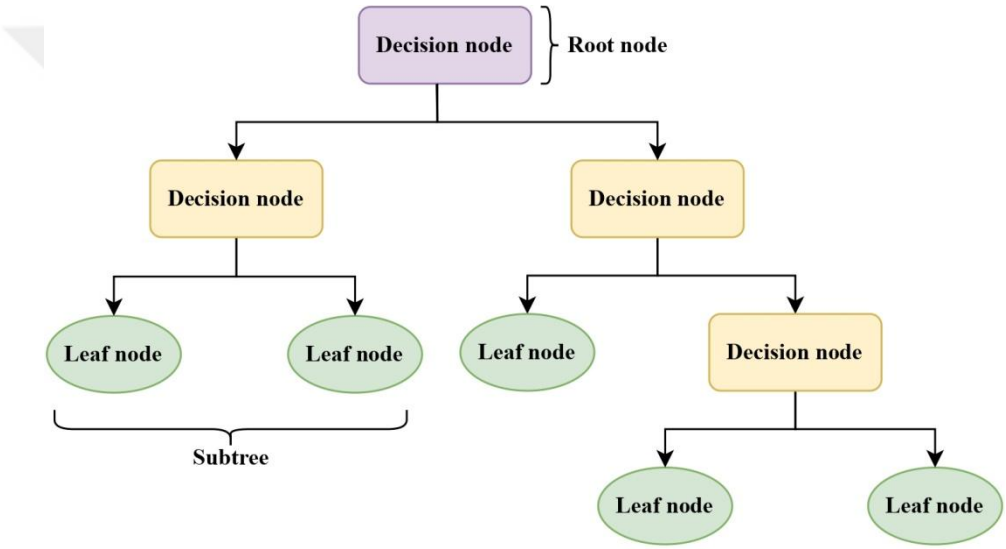


Figure 3.11. The general representation of the DT

3.3.2. Logistic regression (LR)

LR is a popular supervised learning method primarily employed for binary classification problems [92]. It can be adapted for multiclass scenarios using approaches such as one-vs-rest or multinomial logistic regression. In contrast to linear regression, which outputs continuous values, LR predicts the likelihood that an input instance belongs to a specific class. This is done by modeling the association between input features and the log-odds of the outcome. The sigmoid (logistic) function transforms the linear combination of features into a probability ranging from 0 to 1. The model parameters are typically optimized via maximum likelihood estimation, aiming to identify the coefficients that best explain the observed outcomes. Figure 3.12 shows the working principle of the LR algorithm.

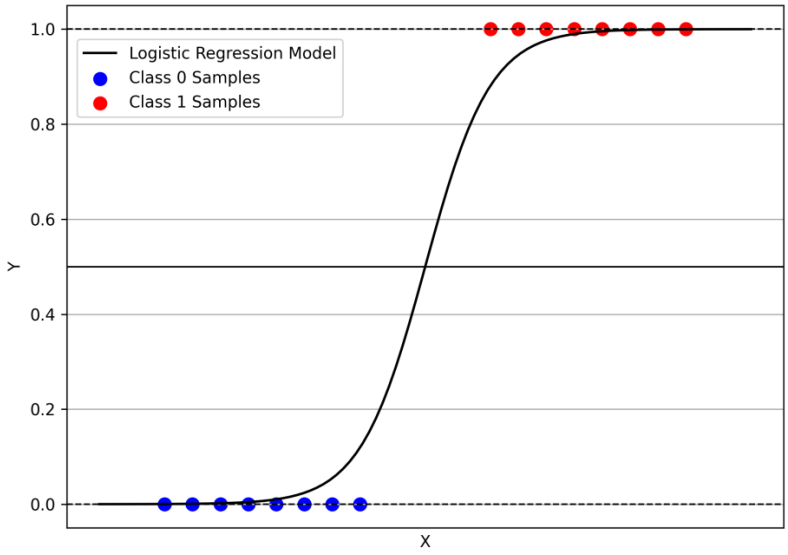


Figure 3.12. The working principle of the LR

3.3.3. Naive bayes (NB)

NB is a probabilistic classification algorithm based on Bayes' Theorem [93]. It computes the posterior probability of a class given a set of input features by combining the prior probability of the class with the likelihood of the observed features. The fundamental formula used in NB is given in Eq. 3.22:

$$P(C \mid X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{3.22}$$

In this equation, $P(C \mid X)$ represents the posterior probability of class C given the feature vector X. $P(X \mid C)$ is the likelihood of observing the features given the class, and $P(C)$ is the prior probability of the class. The denominator $P(X)$ serves as a normalization constant and is typically ignored during class comparisons. The class with the highest posterior probability is selected as the prediction.

3.3.4. Random forest (RF)

RF is an ensemble learning technique that combines multiple DTs to improve predictive performance [94]. A key component in building these trees is the Gini index, which is commonly employed to determine the best splitting criterion within each node of a DT.

This metric is one of the most widely used measures for evaluating branch quality in decision trees. In RF, two primary hyperparameters influence the model's performance: (1) the number of trees to be constructed, and (2) the number of input features randomly selected at each node. RF adopts the Gini index as one of the core criteria in the CART (Classification and Regression Tree) algorithm to assess split quality during tree construction. Specifically, the Gini index (also known as Gini impurity) quantifies the degree of impurity or disorder in a node, and it serves as a basis for determining optimal data partitions. The formal definition of Gini impurity is provided in Eq. 3.23.

$$\mathrm{G} = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.23}$$

where C denotes the total number of distinct classes present in the dataset. p_i represents the relative frequency (or proportion) of instances that belong to class i within a given node. In a DT, the Gini impurity is used to select the best possible split. It does this by comparing the impurity of the parent node with the weighted average impurity of the child nodes. The aim is to find a split that results in the lowest overall Gini impurity. An overall illustration of the RF framework is provided in Figure 3.13.

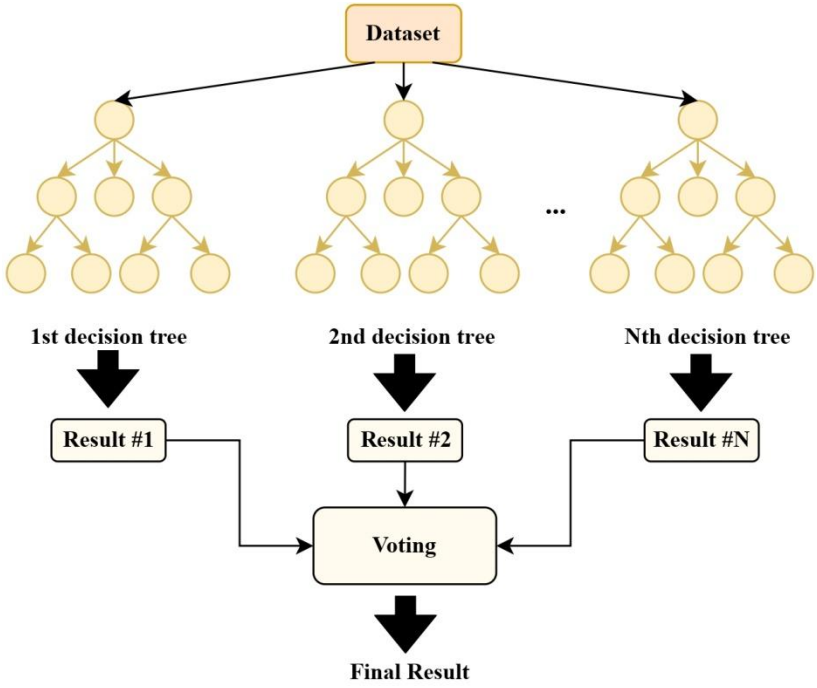


Figure 3.13. The illustration of the RF framework

3.3.5. Extreme gradient boosting (XGBoost)

XGBoost is a highly efficient and scalable ML algorithm based on gradient-boosted DTs [95]. It has gained widespread popularity due to its ability to handle large-scale datasets with high accuracy and low computational cost. Unlike RF, which consists of independently built DTs that rely on majority voting, it constructs trees sequentially. Each subsequent tree is trained to correct the residual errors of the combined predictions made by the previous ones. This algorithm employs gradient descent optimization, which enhances its effectiveness in both classification and regression tasks. Moreover, it integrates L1 and L2 regularization to mitigate overfitting, while supporting parallel computation, handling of missing values, and tree pruning—all of which contribute to its overall robustness and computational efficiency. In this framework, the final prediction is obtained by summing the outputs of all individual trees. The mathematical formulation of this process is presented in Eq. 3.24.

$$\hat{y}_i = \sum_{k=1}^{n} f_k(x_i),\ f_k \in F \tag{3.24}$$

where F represents the set of all regression trees. $f_k(x_i)$ denotes the prediction made by the kth tree for the input x_i. The predicted output for the sample x_i is expressed as $\hat{y}_i$. The complete formulation of the objective function is provided in Eq. 3.25.

$$\text{Obj}(\theta) = \sum_{k=1}^{K} \Omega(f_k) + \sum_{i=1}^{n} l(y_i, \hat{y}_i) \tag{3.25}$$

In this formulation, the term $\sum_{k=1}^{K} \Omega(f_k)$ represents the regularization component, which is designed to penalize the complexity of each individual tree in the model. $\sum_{i=1}^{n} l(y_i, \hat{y}_i)$ corresponds to the loss function, where y_i is the actual value and $\hat{y}_i$ is the predicted output. The overall architecture of the XGBoost model is illustrated in Figure 3.14.

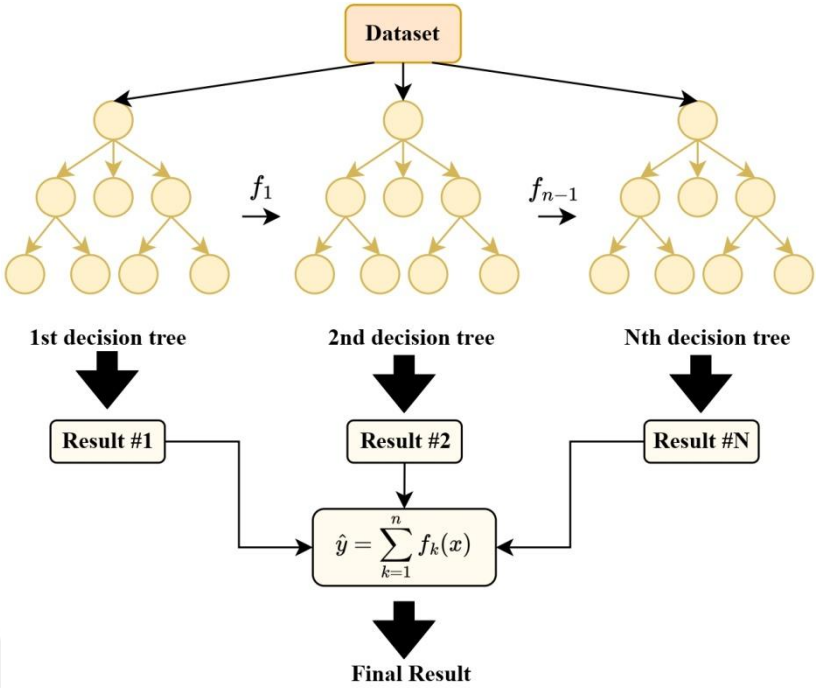


Figure 3.14. The general architecture of the XGBoost

3.4. Quantum Machine Learning (QML)

QML represents a new paradigm that fuses quantum computing with traditional techniques to exploit the computational advantages of quantum systems [96], [97], [98]. Unlike classical models that operate on bits, it utilizes quantum bits (qubits) and leverages quantum circuits to encode and process information.

3.4.1. Qubits and quantum circuits

Qubits are the fundamental units of quantum information [98],[99]. Unlike classical bits which are either 0 or 1, qubits exist in a superposition of both states simultaneously. Mathematically, a qubit state is represented as:

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in C, |\alpha|^2 + |\beta|^2 = 1$ (3.26)

The states $|0\rangle$ and $|1\rangle$ form an orthonormal basis for the 2-dimensional Hilbert space C^2. Measurement collapses the state into either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively.

Quantum circuits consist of sequences of quantum gates applied to qubits. These gates manipulate qubit states through unitary transformations, enabling complex computations with fewer resources compared to classical circuits. Entanglement is another key concept in QML. It creates correlations between qubits such that the state of one qubit directly affects the state of another, which is essential for representing high-dimensional data interactions.

3.4.2. Quantum gates: unitary operators

Quantum gates manipulate qubit states using unitary matrices, preserving the norm of the quantum state. Here are common single-qubit gates used in QML circuits:

Pauli Gate

Pauli-X: The Pauli-X gate is the quantum analogue of the classical NOT gate, performing a bit-flip operation[98], [100], [101]. It inverts the computational basis states by mapping $|0\rangle$ to $|1\rangle$ and vice versa:

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \tag{3.27}$$

In terms of quantum spin systems, this corresponds to flipping the qubit from the spin-down state to the spin-up state and vice versa. Due to this flipping behavior, it is often referred to as a bit-flip gate. On the Bloch sphere, the Pauli-X gate represents a rotation of the qubit state by π radians around the X-axis, effectively reflecting the qubit state across the X-axis. This geometric interpretation highlights its role in changing the sign and orientation of the quantum state in a predictable and reversible manner.

Pauli-Y: The Pauli-Y gate introduces a combined bit-flip and phase-flip to a qubit [98], [100], [101]. Mathematically, it is represented by the following matrix:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{3.28}$$

This gate transforms basis states as follows:

$$Y|1\rangle = -i|0\rangle, \quad Y|0\rangle = i|1\rangle \tag{3.29}$$

The Pauli-Y gate does not merely exchange $|0\rangle$ and $|1\rangle$; it also multiplies them by a phase factor of $\pm i$, introducing a relative phase between the quantum states. This property makes it essential in constructing certain entangled states and quantum interference patterns. Geometrically, on the Bloch sphere, the Pauli-Y gate performs a π rotation around the Y-axis, producing a complex-valued evolution of the quantum state.

Pauli-Z: The Pauli-Z gate is a phase-flip gate, meaning it leaves the $|0\rangle$ state unchanged while inverting the sign of the $|1\rangle$ state [98], [101]:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.30}$$

$$Z|0\rangle = |0\rangle, \; Z|1\rangle = -|1\rangle \tag{3.31}$$

This operation is crucial in phase-based quantum algorithms and in constructing entangled states like the Bell states. While it does not affect measurement probabilities in the computational basis, it alters the relative phase between states, which becomes significant during interference. On the Bloch sphere, the Pauli-Z gate corresponds to a π rotation around the Z-axis, flipping the phase of the quantum state while maintaining its probability amplitudes.

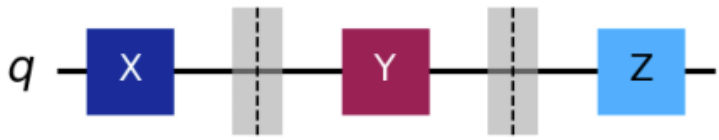


Figure 3.15. Pauli-X, Y, Z gates applied to a single qubit.

Hadamard (H): The Hadamard gate is a fundamental quantum gate that creates superposition [98], [100], [101]. It maps a computational basis state to an equal-weight superposition of $|0\rangle$ and $|1\rangle$, with a relative phase depending on the input:

$$H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3.32}$$

This gate is widely used in quantum algorithms to initialize qubits into a superposed state and to create interference effects. On the Bloch sphere**,** it performs a π rotation around the axis between X and Z, mapping the poles ($|0\rangle$ and $|1\rangle$) onto the equator.

Rotation Gates (R_x, R_y, R_z): Rotation gates are parameterized unitary operations that rotate a qubit's state around one of the three Bloch sphere axes: X, Y, or Z.

$\boldsymbol{R_x(\theta)}$— Rotation around X-axis: Used to rotate a qubit state around the X-axis by angle θ[100], [101].

$$R_x(\theta) = e^{-i\theta X/2} = \begin{pmatrix} cos(\frac{\theta}{2}) & -i\,sin(\frac{\theta}{2}) \\ -i\,sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{3.33}$$

$\boldsymbol{R_y(\theta)}$— Rotation around Y-axis: R_y is commonly used in data encoding, as it maps real-valued features to the amplitudes of quantum states [100], [101].

$$R_y(\theta) = e^{-i\theta \boldsymbol{y}/2} = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{3.34}$$

$\boldsymbol{R_z(\theta)}$— Rotation around Z-axis: This gate introduces a relative phase between the $|0\rangle$ and $|1\rangle$ components [100], [101]. It's often used in variational quantum circuits and quantum Fourier transforms.

$$\boldsymbol{R_z}(\theta) = e^{-i\theta z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{3.35}$$

Figure 3.16 illustrates a quantum circuit composed of a sequence of expressive quantum gates. The circuit begins with a H gate applied to the first qubit to place it into a superposition state. This is followed by a series of rotation gates**:** $R_x(\pi/4)$, $R_y(\pi/3)$, and $R_z(\pi/2)$, which perform rotations around the respective X, Y, and Z axes of the Bloch sphere. These parameterized rotations enable the encoding of input features and serve as trainable components in variational quantum circuits. Finally, a CNOT gate introduces entanglement between the first and second qubits, a key element in enabling quantum correlations. This circuit layout is representative of hybrid quantum-classical neural network architectures used in QML tasks.

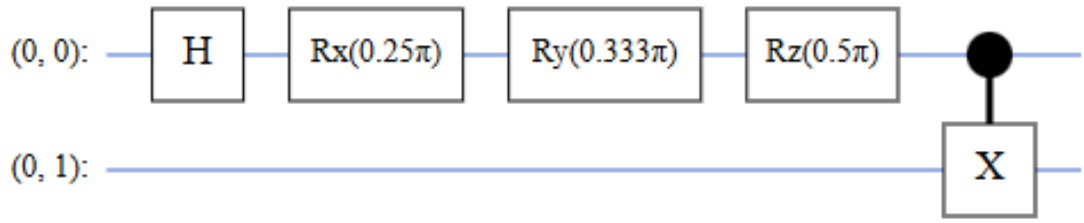


Figure 3.16. Quantum circuit with hadamard, rotation, and entanglement gates

3.4.3. Proposed models

The raw network traffic dataset used in this thesis was pre-processed and optimized using the SFOA, as applied to the Farm-Flow dataset, which will be described in detail in the following section. This nature-inspired metaheuristic algorithm was applied to reduce dimensionality and retain the most relevant features based on global search heuristics. The result of this preprocessing stage was a refined feature space tailored to improve downstream classification performance. However, despite this optimization, QML models typically perform best when operating on a limited number of input features, due to current hardware and simulation constraints as well as the exponential growth of quantum state space with additional qubits.

Feature Normalization and Encoding

Although the initial feature set was optimized using SFOA, the computational limitations of QML models required a further reduction. Therefore, SelectKBest feature selection with mutual information (MI) as the scoring criterion was applied to extract the most predictive features. MI quantifies the dependency between each input feature and the target variable, enabling the identification of those with the highest predictive relevance. Two distinct experiments were conducted:

- **Top 4 Features:** The first strategy selected the top 4 features according to MI scores. These features were used to train and evaluate the hybrid QML model.
- **Top 12 Features:** The second strategy extended the selection to the top 12 features and repeated the training and evaluation.

The comparison revealed that the QML model using the top 12 features significantly outperformed the one trained on only 4 features. Therefore, the configuration using 12 features was selected as the final setup for deeper experimentation.

In the proposed QML framework, angle encoding (also known as rotation encoding) was employed to embed classical features into quantum circuits. This method leverages the parameterization of single-qubit rotation gates to map real-valued classical inputs into quantum states. Specifically, for each input feature x_i, the corresponding qubit undergoes a rotational transformation using a $R_y(\pi x_i)$ gate. This encodes the feature information into the quantum state amplitudes by rotating the qubit around the Y-axis of the Bloch sphere. Angle encoding is both hardware-efficient and expressive, making it well-suited for near-term quantum devices and shallow circuit architectures. It enables smooth control over the quantum state space while preserving the continuous nature of the classical data, allowing the model to benefit from quantum parallelism during training and inference.

Figure 3.17 illustrates how classical input features are embedded into quantum states using the angle encoding technique. Each classical feature $x_i \in [0, 1]$ is mapped to a quantum state through a single-qubit rotation around the Y-axis using the gate $R_y(\pi x_i)$. This transformation encodes the classical data into the amplitudes of the quantum state, effectively preparing the quantum circuit for further processing by the variational layers. The process is efficient and suitable for shallow quantum circuits in near-term devices.

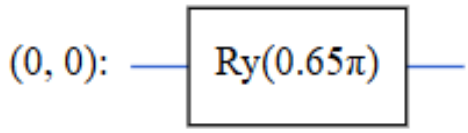


Figure 3.17. Angle encoding of classical features in quantum circuits

The input feature vector is defined as $x = [x_1, x_2, \dots, x_n] \in R^n$, comprising n = 12 features selected through MI. Each feature $x_i \in [0,1]$ is encoded into a single qubit using angle encoding, specifically through a rotation around the Y-axis**:**

$$U_{encode}(x_i) = R_y(\pi x_i) = exp(-i\frac{\pi x_i}{2}Y) \tag{3.36}$$

The total input encoding circuit over n qubits is:

$$U_{input}(X) = \bigotimes_{i=1}^{n} R_y(\pi x_i) \tag{3.37}$$

The circuit in Figure 3.18 shows how a set of classical features $[x_1, x_2, \ldots, x_n]$ is encoded into quantum states using R_y rotations. Each qubit receives a single-qubit rotation $R_y(\pi x_i)$, where $x_i \in [0, 1]$ is a normalized classical input. This process prepares the quantum register for downstream variational quantum processing.

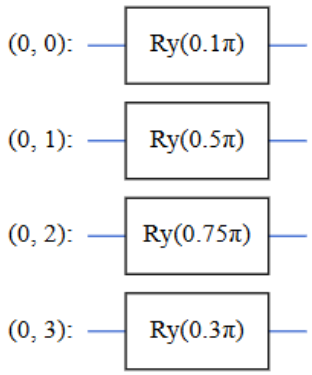


Figure 3.18. Angle encoding of multiple classical features into quantum states

Parameterized Quantum Circuit (Variational Layer)

After encoding, the quantum circuit applies a trainable variational transformation composed of single-qubit rotation gates $R_z(\theta_i^{(z)})$, $R_x(\theta_i^{(x)})$, and two-qubit CNOT gates for entanglement. For each qubit q_i, we define:

$$U_{train}^{(i)} = R_z(\theta_i^{(z)}) \,.\, R_x(\theta_i^{(x)}) \tag{3.38}$$

The entanglement is introduced through nearest-neighbour CNOT gates:

$$U_{entangle} = \prod_{i=1}^{n-1} CNOT(q_i, q_{i+1}) \tag{3.39}$$

The full parameterized quantum circuit (PQC) becomes:

$$U_{PQC}(\theta) = U_{entangle}(\bigotimes_{i=1}^{n} U_{train}^{(i)}) \tag{3.40}$$

Figure 3.19 presents the graphical representations of the parameterized quantum circuits used in this thesis. These circuits illustrate implementations with both 4 and 12 input features.

Measurement and Quantum Output

The final quantum state after encoding and PQC is:

$$|\psi(x,\theta)\rangle = U_{PQC}(\theta) \,.\, U_{input}(X) \,.\, |0\rangle^{\otimes n} \tag{3.41}$$

We perform expectation value measurements on each qubit with respect to the Pauli-Z observable:

$$z_i = \langle \psi(x,\theta) \mid Z_i \mid \psi(x,\theta) \rangle \tag{3.42}$$

These form the quantum output vector:

$$z = [z_1, z_2, \dots, z_n] \tag{3.43}$$

Hybrid QML vs. Pure QML Experiments

To evaluate the effectiveness of quantum circuits in intrusion detection tasks, we conducted an ablation study contains three different case studies:

- **Case 1:** Full-QML (Angle-Encoded) Model
- **Case 2:** Hybrid Quantum-Classical Classification Model
- **Case 3:** Hybrid Model with Multiple Dense Layers

Both models were designed using the same input dataset (top 4 or 12 features selected via MI) and utilized the same quantum circuit for data encoding and processing. The difference lies in the architecture following the quantum layer. An ablation study has been conducted both for binary and multi class classification experiments.

Case 1: Full-QML (Angle-Encoded) Model

The Full-QML (Angle-Encoded) Model is a purely quantum architecture that eliminates all classical neural network layers, relying solely on quantum operations for both feature encoding and classification. Each classical input feature is encoded into a qubit using angle-based R_y rotations, effectively mapping real-valued data into quantum states. These qubits are then processed through a PQC composed of trainable rotation gates (i.e., R_z and R_x) and entangling CNOT gates, typically arranged in multiple layers to capture complex relationships. The final output is derived directly from the expectation values of quantum observables (Pauli-Z measurements). These expectation values serve as the only intermediate outputs used to infer class predictions, with no intermediate Dense layers. This architecture is designed to test the learning capacity of quantum circuits in isolation and explore the boundaries of fully QML in supervised learning tasks. The input feature vector is defined as:

$$x = [x_1, x_2, \dots, x_n] \in R^n \tag{3.44}$$

Each feature is encoded as a rotation on a dedicated qubit:

$$|\psi_{\text{enc}}\rangle = \otimes_{i=1}^{n} R_y(x_i \cdot \pi)|0\rangle \tag{3.45}$$

Apply trainable quantum gates:

$$U(\theta) = \prod_{\ell=1}^{L} \left(\otimes_{i=1}^{n} R_z\left(\theta_i^{(\ell)}\right) R_x\left(\phi_i^{(\ell)}\right) \cdot \text{CNOTs} \right) \tag{3.46}$$

$\ell = 1, \dots, L$: This denotes the number of layers in the circuit. Each layer consists of parameterized single-qubit rotations followed by entangling gates. The quantum state becomes:

$$|\psi_{out}\rangle = U(\theta)|\psi_{enc}\rangle \tag{3.47}$$

Binary Classification: In this binary classification variant, all qubits are measured in the Z basis at the output of the PQC. The resulting vector of expectation values $\mathrm{z} =$

$[\langle Z_0\rangle, \langle Z_1\rangle, \ldots, \langle Z_{n-1}\rangle]$ is passed to classical dense layers. These layers apply nonlinear transformations and a final sigmoid activation to produce a probability between 0 and 1. The model is trained using a binary cross-entropy loss, and all trainable parameters are jointly optimized across both quantum and classical components. This hybrid configuration enables end-to-end quantum-classical learning, where the quantum circuit acts as a trainable feature extractor, and the classical layers serve as the classifier. Output is computed as:

$$\hat{y} = \sigma(f(\mathrm{z})) \tag{3.48}$$

Multi-Class Classification: For multi-class classification, the circuit measures the expectation values of the first k qubits, where k is the number of output classes. These expectation values form a vector $z = [\langle Z_0\rangle, \langle Z_1\rangle, \ldots, \langle Z_{\mathrm{k-1}}\rangle]$, which is passed through a softmax function to yield class probabilities. This approach preserves the fully quantum nature of the model while enabling it to scale to multi-class tasks. All transformations, entanglement, and decision-making are handled by the quantum circuit alone, with the softmax function acting only on the raw quantum outputs. Measure first *k* qubits:

$$z = [\langle Z_0\rangle, \ldots, \langle Z_{k-1}\rangle] \quad \Rightarrow \quad \hat{y} = \mathrm{softmax}(z) \tag{3.49}$$

Case 2: Hybrid Quantum-Classical Classification Model

The Hybrid Quantum-Classical Classification Model combines the representational power of PQCs with the flexible decision boundaries of classical neural networks. In this architecture, classical input features are first encoded into a quantum state using angle encoding, typically via R_y rotations. The encoded qubits are processed by a multi-layer quantum circuit comprising trainable R_z and R_x gates and CNOT entanglement gates, allowing the circuit to learn complex nonlinear transformations in quantum Hilbert space.

The quantum circuit produces a vector of expectation values based on measured observables (usually Pauli-Z operators), which are then passed to a classical Dense layer for the final classification. This design benefits from quantum expressivity in feature transformation while leveraging classical neural networks for learnable output mappings,

making it especially suitable for practical scenarios on today's NISQ (Noisy Intermediate-Scale Quantum) devices. Each feature is encoded as a rotation applied to a dedicated qubit, which is then processed by the PQC, generating the following output:

$$z = [\langle Z_0 \rangle, \dots, \langle Z_m \rangle] \in R^m \tag{3.50}$$

Binary Classification: In the binary classification setup, the quantum circuit output is forwarded to a Dense layer with one neuron and a sigmoid activation function. This classical layer transforms the quantum output into a probability score between 0 and 1, enabling the use of binary cross-entropy loss during training. While the PQC handles most of the feature transformation, the final Dense layer allows for a simple yet flexible decision boundary, making the model robust and trainable even in the presence of noisy or complex patterns. The output in this case will be:

$$\hat{y} = \sigma(w^{\top z} + b) \tag{3.51}$$

Multi-Class Classification: For multi-class classification, multiple qubit expectation values are collected and passed into a classical Dense layer with k output units, followed by a softmax activation. This setup enables the model to produce a valid probability distribution across all classes. The classical layer acts as a learnable projection from the quantum output space to the class space, while the quantum layers perform the heavy lifting in terms of nonlinear feature extraction and entanglement modeling. This hybrid setup achieves a balance between quantum innovation and classical reliability.

$$\hat{y} = \text{softmax}(wz + b) \tag{3.52}$$

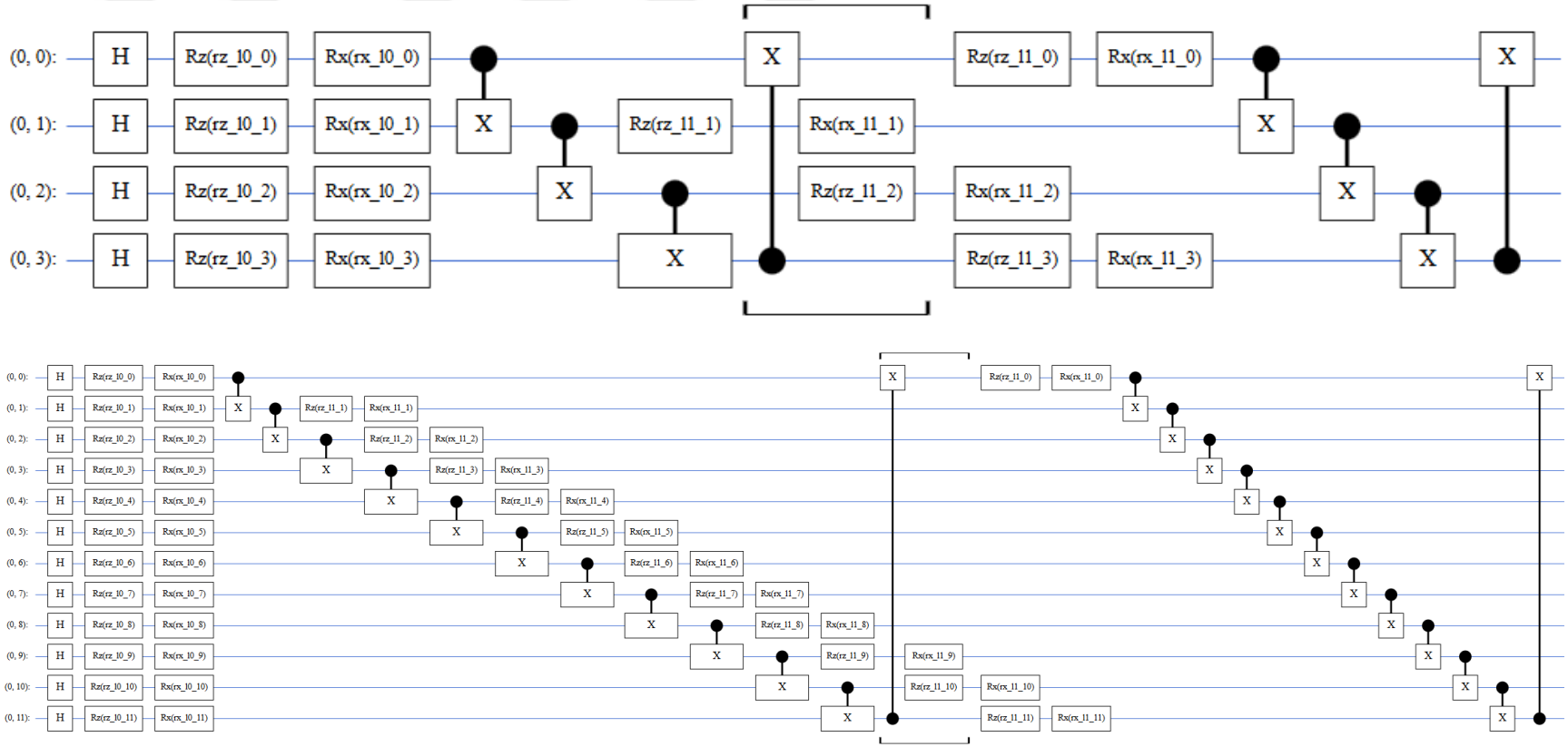


Figure 3.19. Graphical representations of the parameterized quantum circuits with 4 and 12 features

Case 3: Hybrid Model with Multiple Dense Layers

To achieve better performance in this case study, a revised model architecture was adopted—employing quantum phase encoding followed by deeper classical layers. This configuration was designed to enhance the model's capacity to capture complex patterns in a more challenging multiclass intrusion detection scenario. The dataset included network traffic instances labeled across multiple categories, each representing different types of cyberattacks. The preprocessing steps remained consistent with the binary classification task, involving data normalization and the selection of the top twelve most informative features based on MI scores. These features were encoded into quantum states via angle encoding, where each feature was mapped to a rotation using $R_y(\theta)$ gates applied to a dedicated qubit, resulting in a 12-qubit quantum state per input sample. The quantum circuit architecture employed in the hybrid model was carefully designed for both expressivity and hardware efficiency. Each qubit was passed through a sequence of trainable $R_z(\theta)$ and $R_x(\theta)$ gates, forming multiple variational layers, followed by a linear entanglement scheme implemented using CNOT gates between adjacent qubits, including circular connections to ensure full inter-qubit interaction. The outputs of the quantum circuit—expectation values of Pauli-Z measurements—served as quantum feature embeddings and were passed to a classical neural network head for further classification. In the binary classification variant, the quantum outputs were fed into a classical Dense layer with 32 neurons and ReLU activation, followed by a Dropout layer (rate = 0.3) for regularization. A final Dense layer with sigmoid activation produced the binary prediction. This structure enabled the model to combine quantum-learned feature representations with classical post-processing for improved accuracy and robustness. For the multi-class classification setting, the classical head was deepened to better model complex class boundaries. Specifically, the quantum outputs were passed through two Dense layers: the first with 64 neurons and the second with 32 neurons, both using ReLU activation, followed by a final Dense layer with softmax activation to produce class probabilities. This deeper classical backend facilitated the extraction of higher-order representations from the quantum-transformed features, enhancing multiclass discrimination. The model was compiled using the sparse categorical cross-entropy loss function and optimized with the Adam optimizer (learning rate = 0.001). To mitigate the effects of class imbalance, class weights were computed and incorporated during training. The model was trained for 30 epochs with a batch size of 64, and performance was evaluated on a hold-out validation

set. In conclusion, in this model, each feature is encoded as a rotation applied to a dedicated qubit, which is then processed by the PQC, generating the following output:

$$\boldsymbol{z} = [\langle Z_0 \rangle, \dots, \langle Z_m \rangle] \in R^m \tag{3.53}$$

This output will be fed to the classical section as follow:

$$h^{(1)} = \mathrm{ReLU}\left(W^{(1)} z + b^{(1)}\right) \tag{3.54}$$

$$h^{(2)} = \mathrm{ReLU}\left(W^{(2)} h^{(1)} + b^{(2)}\right) \tag{3.55}$$

In case of binary classification, the output will be:

$$\hat{y} = \sigma\left(W^{(3)} h^{(2)} + b^{(3)}\right) \tag{3.56}$$

In case of multi class classification, the output will be:

$$\hat{y} = \mathrm{softmax}\left(W^{(3)} h^{(2)} + b^{(3)}\right) \tag{3.57}$$

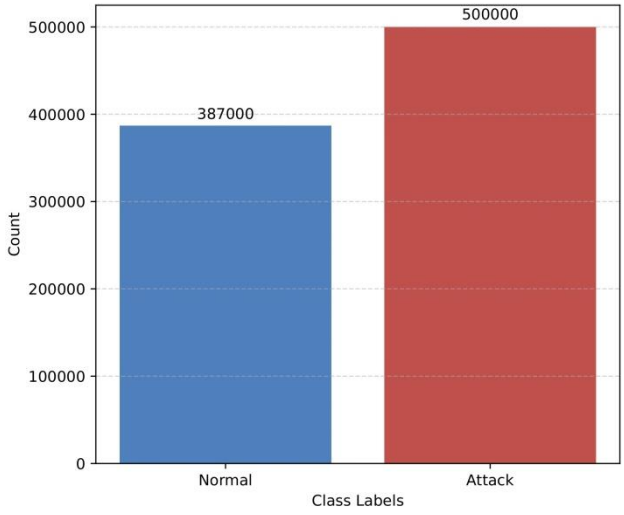
3.5. Datasets

In this thesis, two distinct publicly available datasets were employed to evaluate and benchmark the performance of the proposed intrusion detection approaches. These datasets represent different IoT domains, including smart vehicles, smart agriculture, and medical IoT environments.

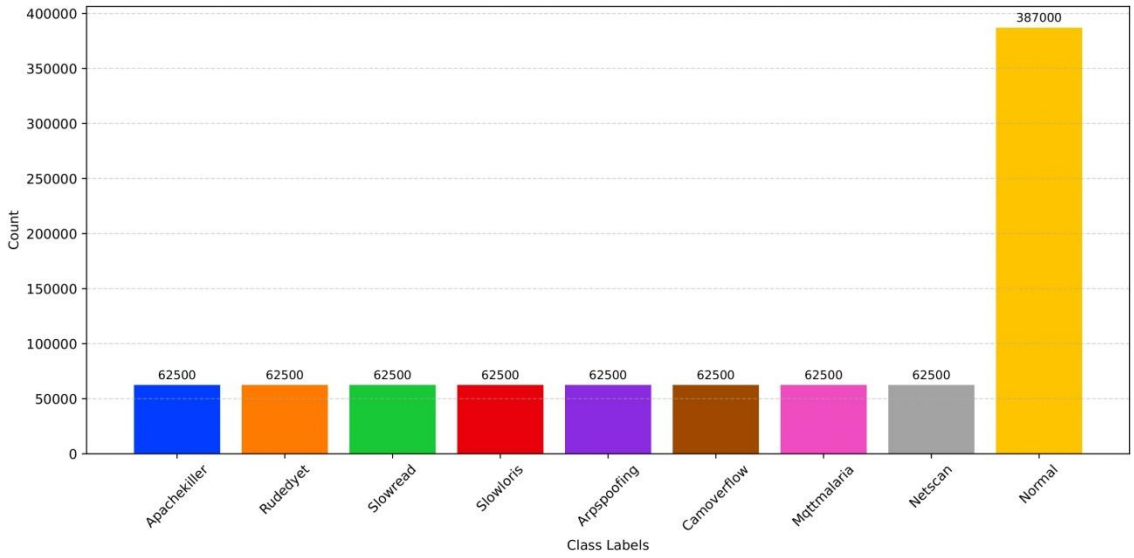
3.5.1. IoMT-TrafficData

The IoMT-TrafficData dataset consists of network traffic data from an IoMT scenario, designed for intrusion detection research [11]. It includes both benign and malicious traffic generated by eight different attack types such as Apache Killer, ARP Spoofing, and CAM Table Overflow. The dataset is divided into two main categories: IP-based and Bluetooth traffic. The IP-based data is further split into packet-based and flow-based approaches, capturing network packet and flow information. Data collection was carried out over approximately 120 hours for benign traffic and 5 hours for malicious traffic, using tools

like tcpdump, Zeek Flowmeter, and Tshark. The dataset provides a rich collection of labeled network traffic, annotated as either benign or malicious to support binary classification tasks. Additionally, it includes detailed labels for multiple attack types, enabling the development and evaluation of ML-based IDS in multiclass classification settings for IoMT environments. In this thesis, we utilized the IP-based data for our experiments. The distribution of classes in the original IP-based packet and IP-based flow datasets is illustrated in Figures 3.20–3.21, respectively. In this study, we utilized the preprocessed version of the dataset provided by the original authors, and the corresponding train-test splits used in our experiments are summarized in Table 3.3.



(a)



(b)

Figure 3.20. Class distributions in the IP-based packet dataset for (a) binary and (b) multiclass classification.

3.5.2. Farm-Flow

The Farm-Flow dataset was developed to address the scarcity of realistic datasets for intrusion detection in smart agriculture networks [12]. It is based on a scalable smart greenhouse scenario that simulates an AG-IoT environment using microcontrollers, sensors, actuators, and a Raspberry Pi server. The dataset contains over 1.3 million network traffic instances, encompassing both benign activity and eight types of cyberattacks. The distribution of classes in the original Farm-Flow dataset is illustrated in Figure 3.22. In this study, we employed its pre-processed version, and the corresponding data split into training, validation, and test sets used in our experiments is presented in Table 3.4.

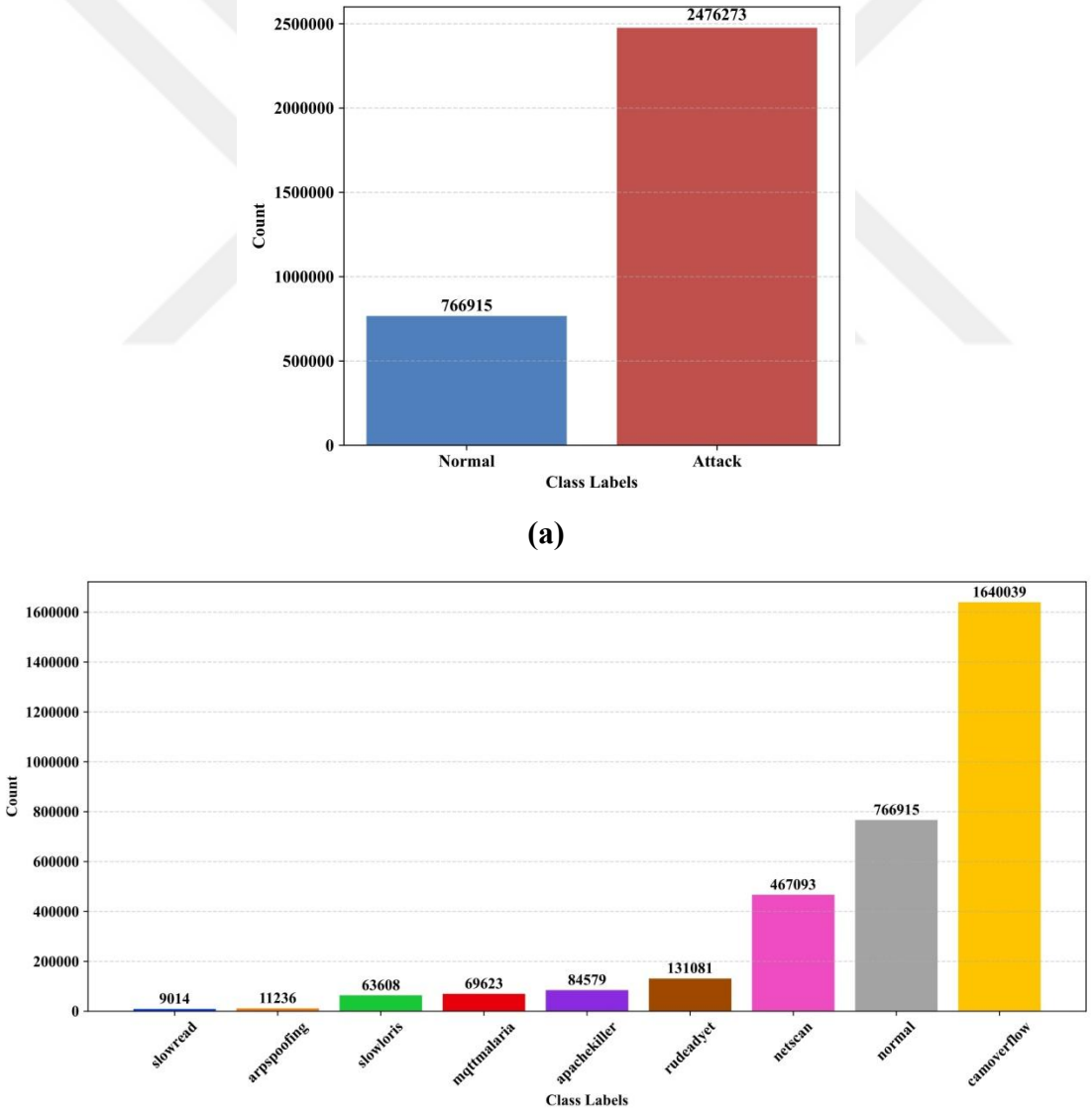


Figure 3.21. Class distributions in the IP-based flow dataset for (a) binary and (b) multiclass classification.

Table 3.3. Train and test sample distribution on the IP-based datasets

Task	Class Label	Packet Train Count	Packet Test Count	Flow Train Count	Flow Test Count
Binary	Normal	281497	93490	168750	75000
	Attack	281003	94010	160730	71436
Multiclass	Normal	281497	93490	168750	75000
	Apachekiller	35222	11645	1317	578
	Rudeadyet	35171	11723	40566	17902
	Slowread	34911	11850	3755	1733
	Slowloris	35123	11742	35764	15811
	Arpspoofing	35114	11670	6271	2861
	Camoverflow	35208	11714	35200	15566
	Mqttmalaria	35232	11753	2782	1252
	Netscan	35022	11913	35075	15733

Table 3.4. Train, validation, and test sample distribution on the Farm-Flow dataset

Task	Class Label	Train Count	Validation Count	Test Count
Binary	Normal	213660	71220	1774
	Attack	207150	69051	1771
Multiclass	Normal	43208	14403	1774
	Arp_Spoofing	1514	505	673
	HTTP_Flood	43209	14402	1774
	ICMP_Flood	43208	14403	1774
	MQTT_Flood	43208	14403	1774
	Port_scanning	43209	14402	1774
	TCP_Flood	43208	14403	1774
	UDP_Flood	43208	14403	1774

As shown in Table 3.4, while the train and test sets were used for all classical ML experiments, a validation set was additionally included for the QML experiments to tune quantum circuit parameters and assess model performance during training.

4. RESULTS AND DISCUSSION

4.1. Experimental Setup

All classical ML experiments in this study were performed on a high-performance workstation running Windows 11 Pro 64-bit. This system featured an 11th Gen Intel(R) Core (TM) i7-11700 processor with 16 logical threads, 32 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU with 12 GB of dedicated VRAM. The setup provided sufficient resources for handling classical preprocessing, feature engineering, and model training using libraries such as Scikit-learn, XGBoost, and TensorFlow. This environment was capable of efficiently managing large datasets (over 500,000 records) and executing various classical ML algorithms for both binary and multiclass classification tasks.

QML experiments were conducted on a separate high-performance workstation also running Windows 11 Pro 64-bit. The system was equipped with an 11th Gen Intel(R) Core (TM) i7-11700 CPU operating at 2.50 GHz (16 logical processors) and 64 GB of RAM, ensuring smooth handling of large-scale training data and complex quantum circuit simulations. GPU acceleration was enabled using an NVIDIA GeForce RTX 3060 GPU with 12 GB of dedicated VRAM, supporting efficient parallel computation and faster model convergence. The quantum circuits and hybrid models were implemented using TensorFlow Quantum (TFQ), a framework that integrates TensorFlow with Cirq for quantum circuit modeling and training. All quantum circuits were simulated on the CPU/GPU backend without relying on quantum hardware.

4.2. Evaluation Metrics

The performance of the proposed classification models was evaluated using four widely used metrics: accuracy, precision, recall, and F1-score. These metrics provide a well-rounded understanding of the model's ability to correctly classify instances and handle imbalanced data. They are all derived from the confusion matrix. Figure 4.1 illustrates the structure of the confusion matrix used for evaluation.

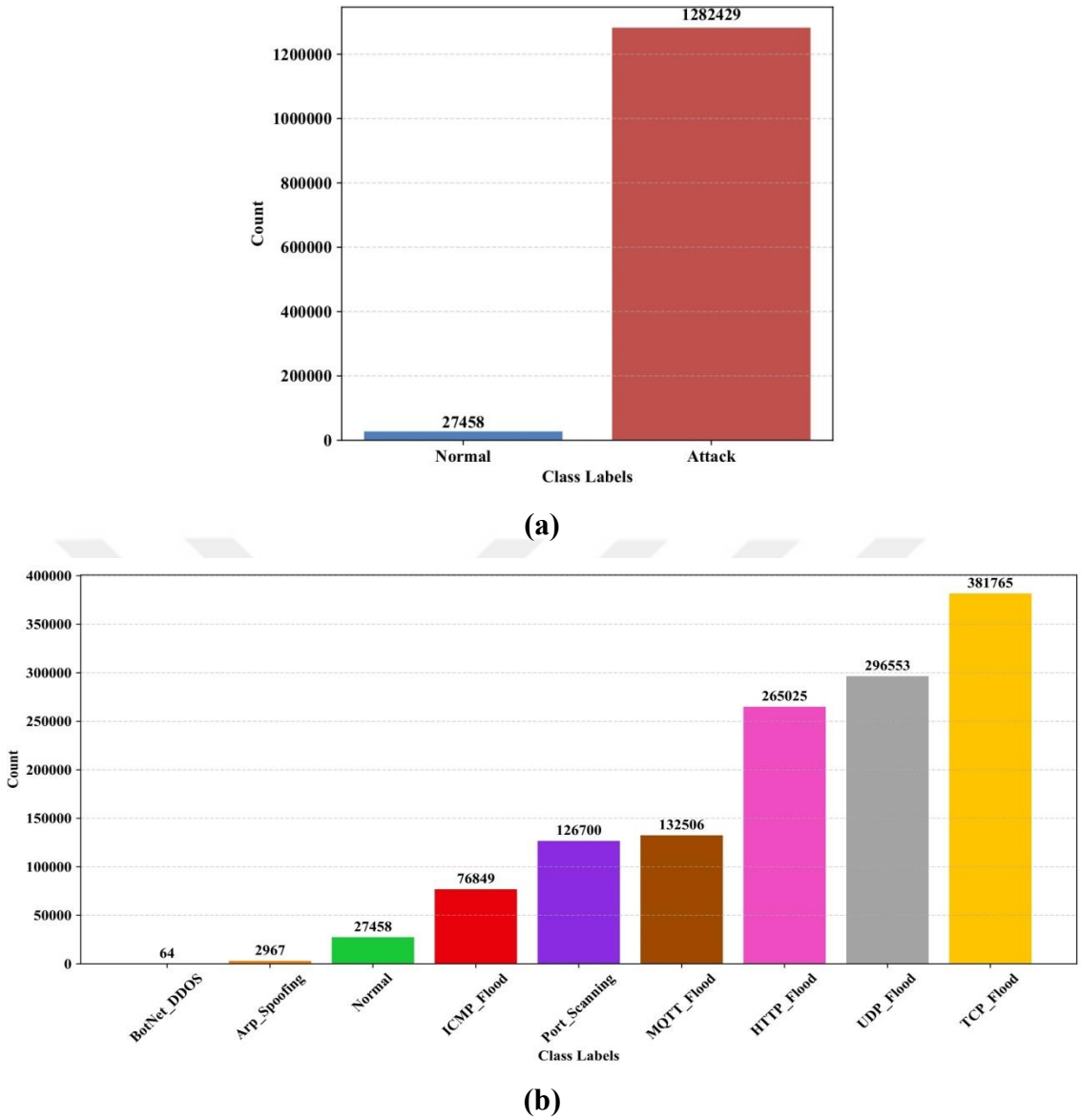


Figure 3.22. Class distributions in the Farm-Flow dataset for (a) binary and (b) multiclass classification.

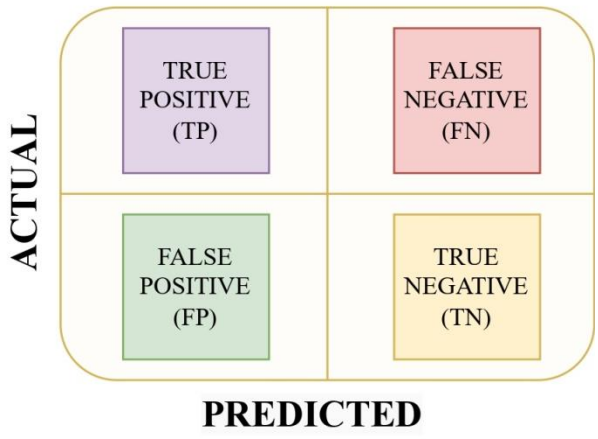


Figure 4.1. General representation of a confusion matrix

where TP denotes the number of correctly identified positive instances. FN represents positive instances that were incorrectly classified as negative. FP refers to negative instances that were mistakenly predicted as positive, and TN indicates the number of correctly identified negative instances. The function and formula of each evaluation metric are summarized in Table 4.1.

Table 4.1. Descriptions of evaluation metrics

Metric	Equation	Description
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Measures the proportion of correctly classified instances, assessing the overall performance of the model.
Recall	$\frac{TP}{TP + FN}$	Evaluates the proportion of actual positives correctly identified by the model.
Precision	$\frac{TP}{TP + FP}$	Measures the proportion of predicted positives that are actually positive.
F1-score	$2 * \frac{Recall * Precision}{Recall + Precision}$	Represents the harmonic mean of precision and recall, balancing their trade-off for overall performance evaluation.

4.3. Obtained Results

4.3.1. Hyperparameter tuning results

In this thesis, the hyperparameters of traditional ML models were optimized using the WaOA. To ensure consistency and comparability with the IoMT-TrafficData benchmark, the preprocessing steps proposed by Areia et al. [11]—including data cleaning, feature extraction, and normalization—were directly adopted without modification. For the Farm-Flow dataset, the pre-processed version released by Ferreira et al. [12] was adopted, which incorporates essential preprocessing steps such as normalization and data balancing. Utilizing the same preprocessing pipeline, our experiments were conducted using enhanced classification models optimized with a recent metaheuristic algorithm. The best hyperparameter values determined by the WaOA for datasets are presented in Table 4.2.

Table 4.2. The best hyperparameter values selected by the WaOA

Algorithm	Parameters	IP-based packet		IP-based flow		Farm-Flow	
		Binary	**Multiclass**	**Binary**	**Multiclass**	**Binary**	**Multiclass**
DT	criterion	"entropy"	"entropy"	"entropy"	"entropy"	"entropy"	"entropy"
	max_depth	8	15	9	13	20	16
	min_samples_split	4	8	8	3	10	9
	min_samples_leaf	2	4	2	1	5	5
LR	C	10	9.7672	2.1640	9.4055	0.01	9.7596
	max_iter	10000	10000	9951	10000	5000	9394
NB	var_smoothing	1e-10	1e-10	0.0015	0.0015	0.0003	1e-10
RF	n_estimators	400	244	314	321	222	385
	max_depth	9	11	11	14	12	19
	min_samples_split	10	7	2	4	6	5
	min_samples_leaf	2	5	1	5	6	3
	max_features	0.9933	0.8682	0.6345	0.4460	0.4349	0.9919
	bootstrap	False	False	False	False	False	True
XGBoost	n_estimators	346	192	490	265	414	175
	max_depth	14	8	3	3	17	19
	learning_rate	0.3460	0.1914	0.2559	0.4215	0.3475	0.1703
	subsample	0.9141	0.5662	1	1	0.8835	1
	colsample_bytree	0.9141	0.5662	0.8253	1	0.9863	1
	gamma	3.4318	1.9023	0.5321	0.2089	3.4300	1.0512

As shown in Table 4.2, WaOA adapts each model's hyperparameters according to its structural characteristics and the nature of the data. The following observations relate specifically to the IP-based packet and flow datasets from the IoMT-TrafficData benchmark. In DT models, multiclass classification tasks require deeper trees: for instance, the depth increases from 8 to 15 for packet data and from 9 to 13 for flow data. The packet-multiclass configuration also sets min_samples_split to 8 and min_samples_leaf to 4 to mitigate overfitting. In contrast, flow-multiclass reduces these values to 3 and 1, respectively, enhancing its capacity to handle noise without compromising generalization. LR consistently uses max_iter ≈ 10,000 to ensure complete convergence across all tasks. While the regularization strength (C) is adjusted based on the dataset and classification type—ranging from 2.1640 in flow-binary to 10 in packet-binary—no scenario requires fewer iterations. In NB, var_smoothing is fixed at 1e-10 for packet data but increased to 0.0015 for flow data, providing better smoothing for variability and reducing sensitivity to noisy inputs. RF shows considerable variation in n_estimators (from 244 to 490) and increases max_depth to 14 for multiclass tasks. Bootstrap sampling is disabled in all cases, allowing each tree to train on the complete dataset. To preserve model diversity,

max_features is lowered in complex scenarios (e.g., to 0.4460 in flow-multiclass), and parameters such as min_samples_split and min_samples_leaf are task-tuned to balance bias and variance. XGBoost exhibits the most extensive tuning, with significant variation in both structural and regularization-related parameters. For example, packet-binary uses a deep structure (max_depth = 14) and a high learning_rate of 0.3460, while flow-binary opts for a shallower tree (max_depth = 3) and a lower learning_rate of 0.2559. Subsampling ratios (subsample and colsample_bytree) and regularization via gamma are also customized per scenario, further preventing overfitting by controlling tree complexity.

In contrast, the Farm-Flow dataset—a real-world agricultural dataset—demonstrates distinct tuning patterns that reflect its unique characteristics. In DT models, tree depth increases significantly (up to 20 in binary and 16 in multiclass tasks). Despite this increase in depth, overfitting was effectively prevented by setting higher values for min_samples_split and min_samples_leaf, such as (10, 5) in binary classification. In LR, the algorithm required much stronger regularization for binary classification (C = 0.01), indicating higher feature variance, while max_iter was reduced to 5000 in line with faster convergence. NB models adjusted smoothing values to better handle data variability: 0.0003 for binary classification and 1e-10 for multiclass, implying well-separated class distributions in the latter. RF configurations in Farm-Flow were notable for enabling bootstrap = True, which was unique to this dataset and helped manage data heterogeneity. Depth increased further to 19 in multiclass classification, while n_estimators was set to 222 for binary and 385 for multiclass, and max_features values were dataset-specific to preserve model robustness. Finally, XGBoost models for Farm-Flow were configured with relatively deep trees (max_depth = 17 and 19) and strong learning rates (e.g., 0.3475). These configurations were complemented by fine-tuned sampling and regularization parameters (gamma, subsample), which capitalized on the dataset's informative structure while effectively avoiding overfitting.

4.3.2. Classical machine learning results

IoMT Results

The classification results obtained from the IP-based packet dataset are presented in Table 4.3, while those derived from the IP-based flow dataset are reported in Table 4.4. In both

tables, the best values are highlighted in bold. While the authors employed SVM as one of their classification models, it is well-known that SVM suffers from certain limitations [102]. These include high computational complexity and sensitivity to the choice of kernel function, particularly when dealing with large-scale datasets like IoMT-TrafficData. In contrast, we applied XGBoost, a gradient boosting framework known for its high efficiency, scalability, and ability to handle missing data effectively. Moreover, XGBoost is less prone to overfitting compared to traditional models like SVM, especially when robust hyperparameter tuning techniques are employed. In the presented tables, the best performance metrics are highlighted in bold for clear comparison.

Table 4.3. Comparison of classification performance between the original authors' methods and our thesis for IP-based packet dataset

Task	Algorithm	Areia et al. [11]				Our study			
		Accuracy	**Recall**	**Precision**	**F1 score**	**Accuracy**	**Recall**	**Precision**	**F1 score**
Binary	DT	99.15	99.15	99.15	99.15	**99.89**	**99.89**	**99.89**	**99.89**
	LR	92.03	92.03	93.12	92.07	**92.27**	**92.27**	**93.30**	**92.24**
	NB	92.45	92.45	93.44	**92.94**	**92.56**	**92.56**	**93.52**	92.51
	RF	**99.89**	**99.89**	**99.89**	**99.89**	**99.89**	**99.89**	**99.89**	**99.89**
	SVM/XGBoost	92.03	92.03	93.13	92.08	**99.73**	**99.73**	**99.73**	**99.73**
Multiclass	DT	93.89	93.89	97.07	95.19	**99.48**	**99.48**	**99.50**	**99.47**
	LR	91.50	91.50	**93.40**	**92.44**	**91.62**	**91.62**	92.35	89.68
	NB	**79.65**	**79.65**	93.40	83.03	**79.65**	**79.65**	**93.41**	**83.04**
	RF	99.47	99.47	99.49	**99.48**	**99.48**	**99.48**	**99.50**	**99.48**
	SVM/XGBoost	91.50	91.50	92.26	91.88	**99.31**	**99.31**	**99.33**	**99.31**

When examining the binary classification results in Table 4.3, RF achieved the best performance in both our study and the original paper. A success rate of 99.89% was achieved across all metrics. This result indicates that the RF algorithm has already reached optimal performance on the IoMT-TrafficData dataset, and the effect of the optimization process remained limited. However, XGBoost significantly outperformed the SVM algorithm. While the F1-score for SVM was 92.08%, it reached 99.73% with XGBoost, representing an improvement of approximately 7.65%. The rapid training time, overfitting control, and strong generalization capability of XGBoost played a significant role in this success. Improvements are also clearly visible in other algorithms. LR improved slightly from 92.07% to 92.24%, and the NB showed a minor change from 92.94% to 92.51%. A more notable enhancement is observed in the DT algorithm, which increased its performance from 99.15% to 99.89% across all metrics. This improvement of 0.74% indicates that our optimization process had a positive impact on DT, pushing it to achieve the same performance level as RF. When examining multiclass classification, performance

improvements are observed in most metrics across all algorithms. DT achieved the most significant improvement in accuracy, increasing by 5.59%. In LR, some metrics showed improvements, while others experienced declines. NB results remained almost unchanged in both our study and the original study. This consistency suggests that the optimization process has little or no effect on NB, possibly due to its simple probabilistic nature which struggles to capture complex patterns in multi-class classification tasks. For RF, the results remained virtually unchanged, with an F1-score of 99.48% in both our study and the original paper. However, unlike NB, this lack of improvement is not a limitation but rather an indication of the model's exceptional performance. The consistently high results suggest that RF was already operating at near-optimal performance, effectively capturing complex patterns within the dataset even before applying further optimization. XGBoost, on the other hand, demonstrated remarkable performance improvements, with the F1-score increasing from 91.88% to 99.31%. This substantial enhancement of approximately 7.43% highlights the effectiveness of XGBoost in handling complex multiclass classification tasks. The substantial improvement in XGBoost's F1-score can be attributed to its high sensitivity to hyperparameter configurations. Unlike Random Forest, which is relatively robust even with default settings, XGBoost's performance is greatly influenced by factors such as tree depth, learning rate, regularization (gamma), and sampling ratios. The WaOA algorithm effectively tuned these parameters to optimize the learning process, reduce overfitting, and enhance generalization—especially critical in multiclass scenarios where class boundaries are more complex. Consequently, this tuning led to a dramatic 7.43% performance gain, highlighting the strength of XGBoost when appropriately optimized.

In Table 4.4, noticeable improvements are observed across all metrics for every algorithm in binary classification. The most significant enhancement in accuracy is achieved by NB, with an approximate increase of 10.05%. While DT, LR, and RF already demonstrated strong performance in the original paper, our study achieved superior results across all metrics. XGBoost emerged as one of the most successful algorithms, achieving a consistent score of 99.93% across all metrics, demonstrating its robustness and efficiency in handling binary classification tasks. Likewise, in multiclass classification, improvements are observed in NB except for precision. It is also observed that XGBoost achieved the best performance, even if only by a very small margin. Overall, the optimization process positively impacted all models, leading to an increase in performance.

Table 4.4. Comparison of classification performance between the original authors' methods and our thesis for IP-based flow dataset

Task	Algorithm	Areia et al. [11]				Our study			
		Accuracy	Recall	Precision	F1 score	Accuracy	Recall	Precision	F1 score
Binary	DT	99.90	99.90	99.90	99.90	**99.93**	**99.93**	**99.93**	**99.93**
	LR	99.41	99.41	99.41	99.41	**99.42**	**99.42**	**99.42**	**99.42**
	NB	87.90	87.90	90.09	88.98	**97.95**	**97.95**	**97.95**	**97.95**
	RF	99.89	99.89	99.89	99.89	**99.93**	**99.93**	**99.93**	**99.93**
	SVM/XGBoost	99.39	99.39	99.39	99.39	**99.93**	**99.93**	**99.93**	**99.93**
Multiclass	DT	99.45	99.45	99.45	99.45	**99.57**	**99.57**	**99.57**	**99.57**
	LR	96.95	96.95	97.13	97.04	**97.35**	**97.35**	**97.52**	**97.35**
	NB	92.50	92.50	**92.97**	92.61	**92.78**	**92.78**	92.90	**92.78**
	RF	99.52	99.52	99.52	99.52	**99.57**	**99.57**	**99.57**	**99.57**
	SVM/XGBoost	96.93	96.93	96.99	96.96	**99.58**	**99.58**	**99.58**	**99.58**

The most effective models identified are DT for IP-based packet binary classification, RF for IP-based packet multiclass classification, and XGBoost for both IP-based flow binary classification and IP-based flow multiclass classification. Figure 4.2 presents the confusion matrices for binary classification, with DT applied to IP-based packet on the top and XGBoost applied to IP-based flow on the bottom. Figure 4.3 shows the confusion matrices for multiclass classification, with RF applied to IP-based packet on the top and XGBoost applied to IP-based flow on the bottom.

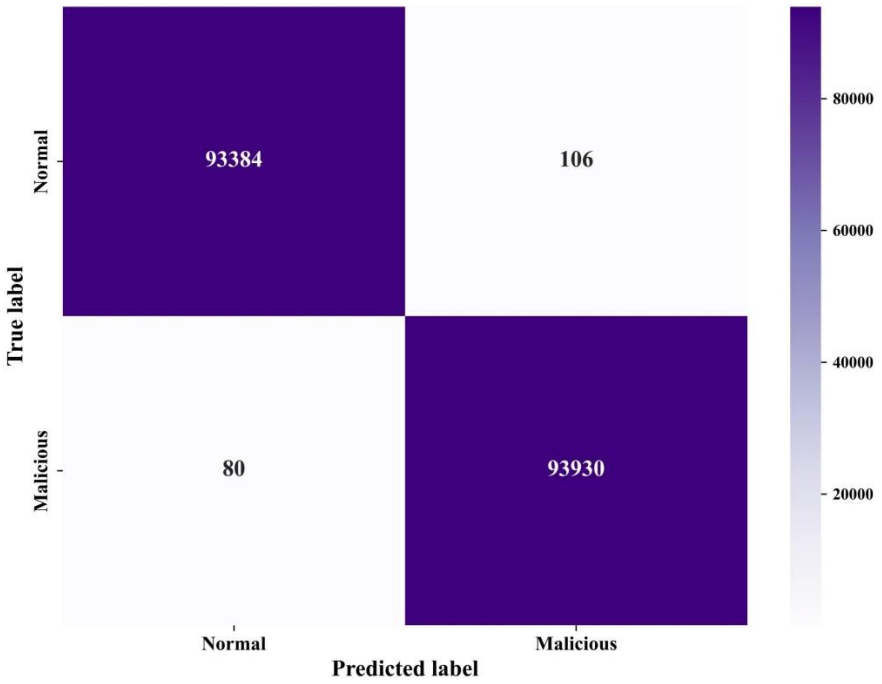


Figure 4.2. Confusion Matrices for Binary Classification: DT for IP-based packet (top), XGBoost for IP-based flow (bottom)

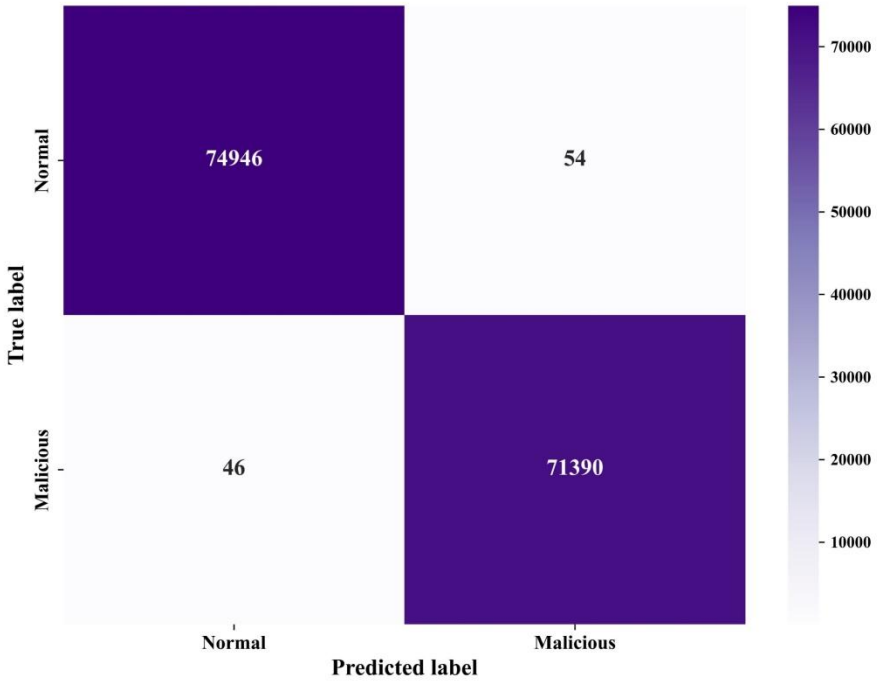


Figure 4.2. (continued) Confusion Matrices for Binary Classification: DT for IP-based packet (top), XGBoost for IP-based flow (bottom)

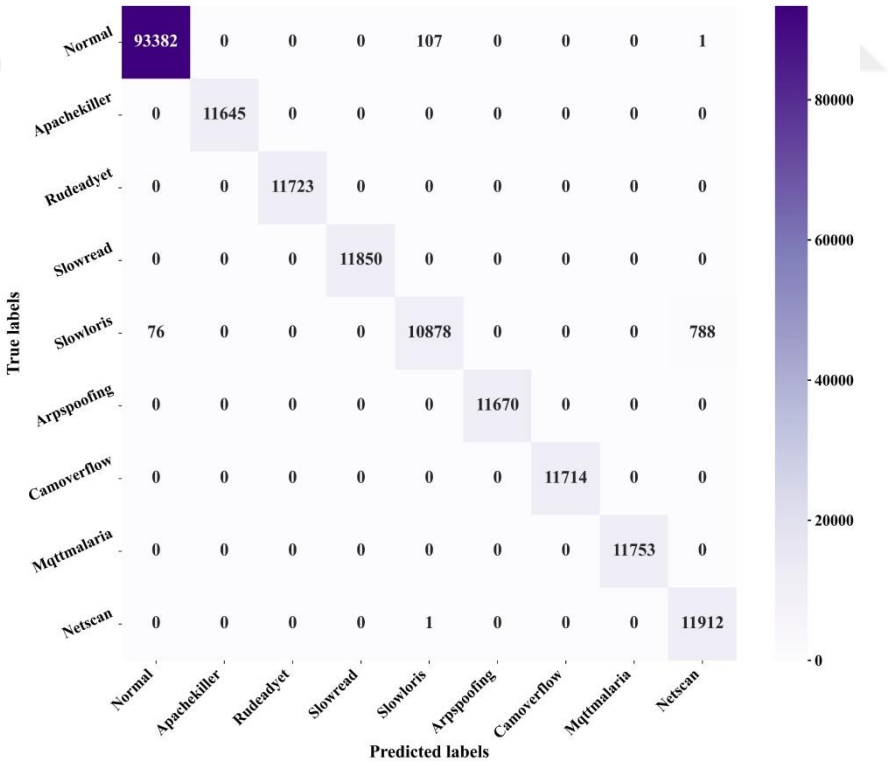


Figure 4.3. Confusion Matrices for Multiclass Classification: RF for IP-based packet (top), and XGBoost for IP-based flow (bottom)

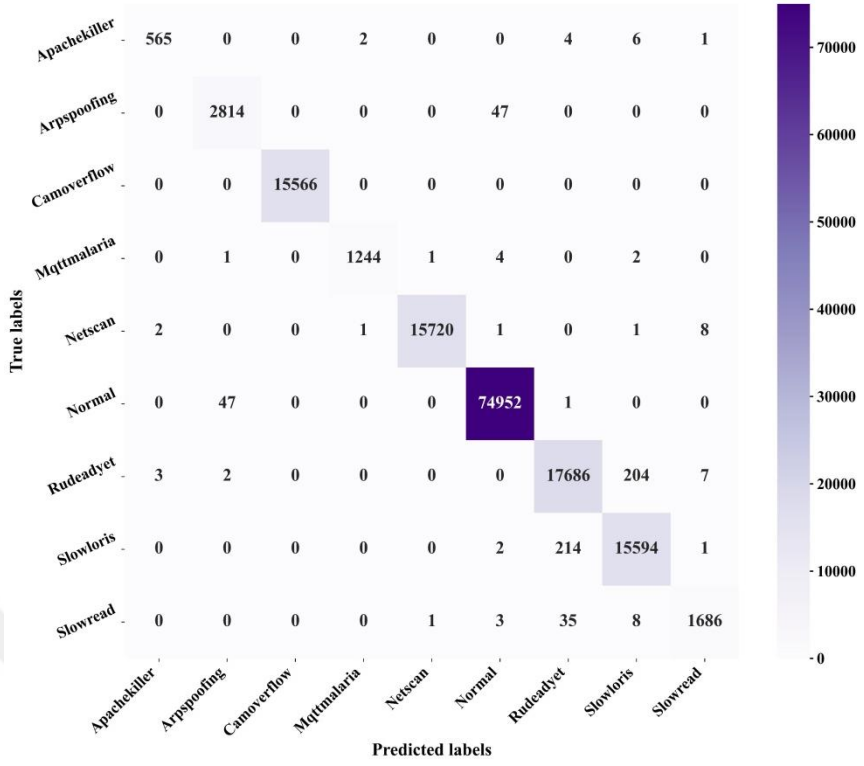


Figure 4.3. (continued) Confusion Matrices for Multiclass Classification: RF for IP-based packet (top), and XGBoost for IP-based flow (bottom)

The RF model achieves highly accurate classification across the majority of classes. For example, it correctly predicts 93,382 instances of the Normal class, as well as high accuracy for attacks such as Apachekiller (11,645), Rudeadyet (11,723), Slowread (11,850), and Mqttmalaria (11,753). Most confusion is minimal; however, for the Slowloris class, 76 instances are misclassified as Normal and 788 as Netscan, indicating some degree of overlap or misinterpretation among these specific classes. The XGBoost model also delivers strong results, successfully identifying large numbers of instances for several classes including Normal (74,952), Rudeadyet (17,686), Netscan (15,720), and Camoverflow (15,566). Other classes such as Arpspoofing (2,814), Mqttmalaria (1,244), and Apachekiller (565) are also predicted with relatively high accuracy, though minor misclassifications are observed. For instance, the Slowread class is largely predicted correctly, with 1,686 accurate classifications and only a few instances misclassified as Rudeadyet (35), Slowloris (8), Mqttmalaria (3), or Normal (1). These results demonstrate the overall reliability and effectiveness of ML models in accurately distinguishing between multiple attack types and Normal traffic within the context of multiclass classification.

AG-IoT Results

The classification results obtained from the Farm-Flow dataset are presented in Table 4.5, with the best values highlighted in bold. While the original authors used a DNN model, we preferred XGBoost due to its faster training time, better handling of tabular data, and lower risk of overfitting.

Table 4.5. Comparison of classification performance between the original authors' methods and our thesis for Farm-Flow dataset

Task	Algorithm	Ferreira et al. [12]				Our study			
		Accuracy	**Recall**	**Precision**	**F1 score**	**Accuracy**	**Recall**	**Precision**	**F1 score**
Binary	DT	86.01	86.01	86.30	85.98	**92.98**	**92.98**	**93.73**	**92.95**
	LR	84.51	84.51	84.80	84.48	**92.47**	**92.47**	**93.29**	**92.43**
	NB	88.60	88.60	89.62	88.53	**90.55**	**90.55**	**91.64**	**90.49**
	RF	86.91	86.91	87.12	86.89	**92.83**	**92.83**	**93.67**	**92.80**
	DNN/XGBoost	92.67	92.67	93.55	92.63	**93.34**	**93.34**	**94.05**	**93.32**
Multiclass	DT	75.81	75.81	81.45	**72.50**	**76.72**	**76.72**	**83.27**	72.07
	LR	75.37	75.37	74.51	70.20	**75.50**	**75.50**	**80.35**	**70.50**
	NB	60.80	60.80	**77.68**	**57.93**	**60.93**	**60.93**	67.20	57.32
	RF	76.37	76.37	81.70	**72.30**	**76.72**	**76.72**	**83.52**	72.00
	DNN/XGBoost	74.96	74.96	78.70	69.18	**76.76**	**76.76**	**83.34**	**72.05**

According to Table 4.5, the models optimized by the WaOA outperform those developed by Ferreira et al. [12] on the Farm-Flow dataset. Notably, in the binary classification task, the XGBoost model achieves the highest performance with 93.34% accuracy and an F1 score of 93.32%. These results surpass the original DNN model's performance (92.67% accuracy, 92.63% F1 score). Traditional models such as DT, LR, NB, and RF also demonstrate consistent improvements across all metrics. In multiclass classification, the WaOA-optimized XGBoost model again shows competitive performance (76.76% accuracy, 72.05% F1 score), slightly outperforming the original DNN. Models like DT and RF also demonstrate meaningful gains compared to the baselines reported by the original authors. It is important to emphasize that no additional preprocessing was performed in this study. Instead, the preprocessed version released by Ferreira et al. was used directly. This approach isolates the impact of WaOA-based hyperparameter tuning and allows for a fair and focused comparison of model optimization effectiveness.

The most effective models identified for the Farm-Flow dataset are XGBoost for binary classification and DT for multiclass classification. Figure 4.4 presents the corresponding

confusion matrices, with XGBoost (binary classification) shown on the top and DT (multiclass classification) on the bottom.

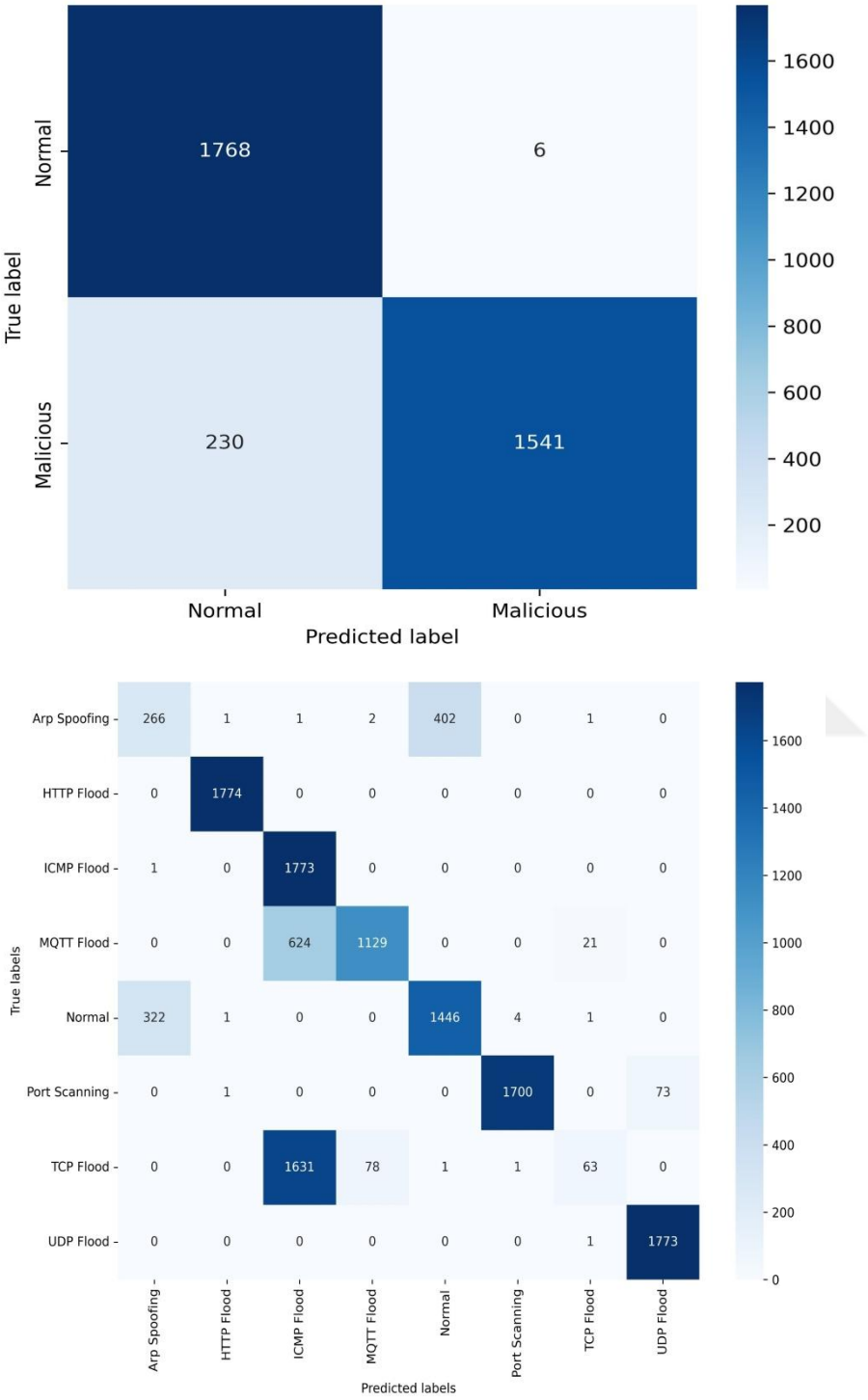


Figure 4.4. Confusion Matrices for the Farm-Flow Dataset: XGBoost for binary classification (top), and DT for multiclass classification (bottom)

According to Figure 4.4, the XGBoost model achieved strong binary classification performance on the Farm-Flow dataset. It correctly predicted 1,768 Normal and 1,541 Malicious instances. Only 6 Normal samples were misclassified as Malicious, while 230 Malicious samples were incorrectly labeled as Normal, indicating high precision but a slightly higher false negative rate. In multiclass classification, the DT model showed accurate results for several classes—correctly predicting, for example, 1,774 HTTP Flood, 1,773 ICMP Flood, and 1,700 Port Scanning instances. However, it struggled with classes like Arp Spoofing, where 402 instances were misclassified as Normal, and MQTT Flood, with 624 instances mislabeled, mostly as ICMP Flood. These results highlight strong overall performance but also reveal challenges in distinguishing between certain overlapping classes in AG-IoT environments.

4.3.3. Quantum machine learning results

Due to the high computational cost of quantum simulations, the QML experiments were conducted only for the AG-IoT domain. Accordingly, Table 4.6 presents the features selected by the SFOA for both binary and multiclass classification tasks on the Farm-Flow dataset. While preserving the preprocessing provided by the original authors, feature selection was performed independently using SFOA. These features are used as inputs for training the QML models. Their effectiveness is subsequently evaluated for both task types.

Table 4.6. Features selected in both binary and multiclass tasks

Task	Selected Features
Both Binary and Multiclass	fwd_pkts_payload.tot, flow_iat.std, idle.std, bwd_pkts_payload.std, bwd_iat.tot, fwd_iat.max, flow_pkts_payload.tot, active.std, bwd_PSH_flag_count, fwd_pkts_payload.min, active.min, orig_ip_bytes, bwd_header_size_max, fwd_header_size_min, idle.min, fwd_bulk_packets, bwd_bulk_bytes, bwd_bulk_rate, fwd_pkts_per_sec, conn_state, idle.tot, flow_pkts_per_sec, fwd_bulk_bytes, active.max, bwd_iat.min, flow_ACK_flag_count, bwd_pkts_payload.max, flow_duration, fwd_bulk_rate, flow_pkts_payload.avg, fwd_subflow_pkts, fwd_header_size_max, data_pkts_difference
Only in Multiclass	flow_iat.tot, fwd_PSH_flag_count

As shown in Table 4.6, most of the features were commonly selected for both binary and multiclass classification tasks. Notably, all features selected for the binary task were also present in the multiclass feature set. However, the multiclass task included two additional features—*flow_iat.tot* and *fwd_PSH_flag_count*—which were not selected for the binary task. This highlights a substantial overlap in the optimal feature sets identified by the SFOA, indicating that both tasks share similar underlying patterns in the data. Based on these feature subsets, QML models were constructed for both classification tasks. Due to input size limitations of quantum circuits, an additional feature selection was performed using the MI method. The model architectures are summarized in Table 4.7, and their evaluation results are presented below. While a total of seven distinct model configurations were explored, accuracy and loss plots are presented only for three representative models: the base model (M1) for binary classification, the best-performing binary model (M2), and the top-performing multiclass model (M5). These plots were included to illustrate differences in training behavior and convergence across baseline and optimized configurations.

Table 4.7. An overview of ablation experiments QML models

Model ID	Features	Task	Quantum Layers	Classical Layers	Output Type
M1	4	Binary	PQC (3 layers)	Dense(32) + Dropout + Dense(1)	Sigmoid
M2	12	Binary	PQC (3 layers)	Dense(32) + Dropout + Dense(1)	Sigmoid
M3	12	Binary	PQC (3 layers)	Dense(1)	Sigmoid
M4	12	Binary	PQC only	None	Mean + Sigmoid
M5	12	Multiclass	PQC (3 layers)	Dense(32) + Dense(64) + Dense(8)	Softmax
M6	12	Multiclass	PQC (3 layers)	Dense(8)	Softmax

Binary Classification

Hybrid Quantum–Classical Models for Binary Classification Using 4 and 12 Features (M1-M2). In this experiment, hybrid quantum–classical models were developed to perform binary classification on network intrusion data using the top features selected through MI analysis. Two separate models were constructed based on different input dimensionalities: the first model was trained using the top four features, while the second employed the top twelve features. Since the model trained with four features (M1) yielded comparatively

lower performance than the others, subsequent experiments proceeded with the twelve-feature configuration. In both cases, the input features were encoded into quantum states using angle encoding with $R_y(\pi x_i)$ rotations, enabling smooth control over qubit states and hardware-efficient implementation. The PQC consisted of three variational layers composed of R_z and R_x rotations along with CNOT entanglement gates. For both models, this quantum backbone was followed by a classical neural head composed of a 32-unit ReLU-activated Dense layer, a dropout layer, and a final sigmoid output node. The training performance of both hybrid quantum–classical models was evaluated using accuracy and loss metrics. Figure 4.5 illustrates the training and validation accuracy and loss curves for the model trained with the top four features, while Figure 4.6 presents the corresponding performance curves for the model trained with the top twelve features.

As illustrated in Figure 4.5, the training and validation accuracy trends over 30 epochs demonstrate strong and stable performance. The model achieved a training accuracy of 97.4% and a validation accuracy peaking at 97.9%, indicating excellent generalization capability and stability without overfitting. The loss curves show a steady decline throughout training, with validation loss reaching a minimum of 0.0782, further confirming convergence. Figure 4.6 demonstrates enhanced performance, with both accuracy and loss curves reflecting consistently high results. In this case, the model achieved a peak validation accuracy of 99.33%, showing a notable improvement over the previous experiment with four features. Additionally, the validation loss steadily declined to 0.0340 in the final epoch, demonstrating rapid convergence and strong generalization.

Binary Classification using Quantum Circuit with Classical Output Layer (M3). In this experimental setting, the hybrid model was further simplified to assess the expressive capacity of a quantum neural network when coupled with only a single classical Dense output layer. As before, twelve features were encoded via angle encoding using $R_y(\pi x_i)$ gates. The PQC architecture included three variational layers of parameterized R_z, R_x gates, and circular CNOT entanglement. It was followed by a single classical sigmoid-activated Dense layer, omitting the intermediate dense and dropout layers used in prior hybrid configurations.

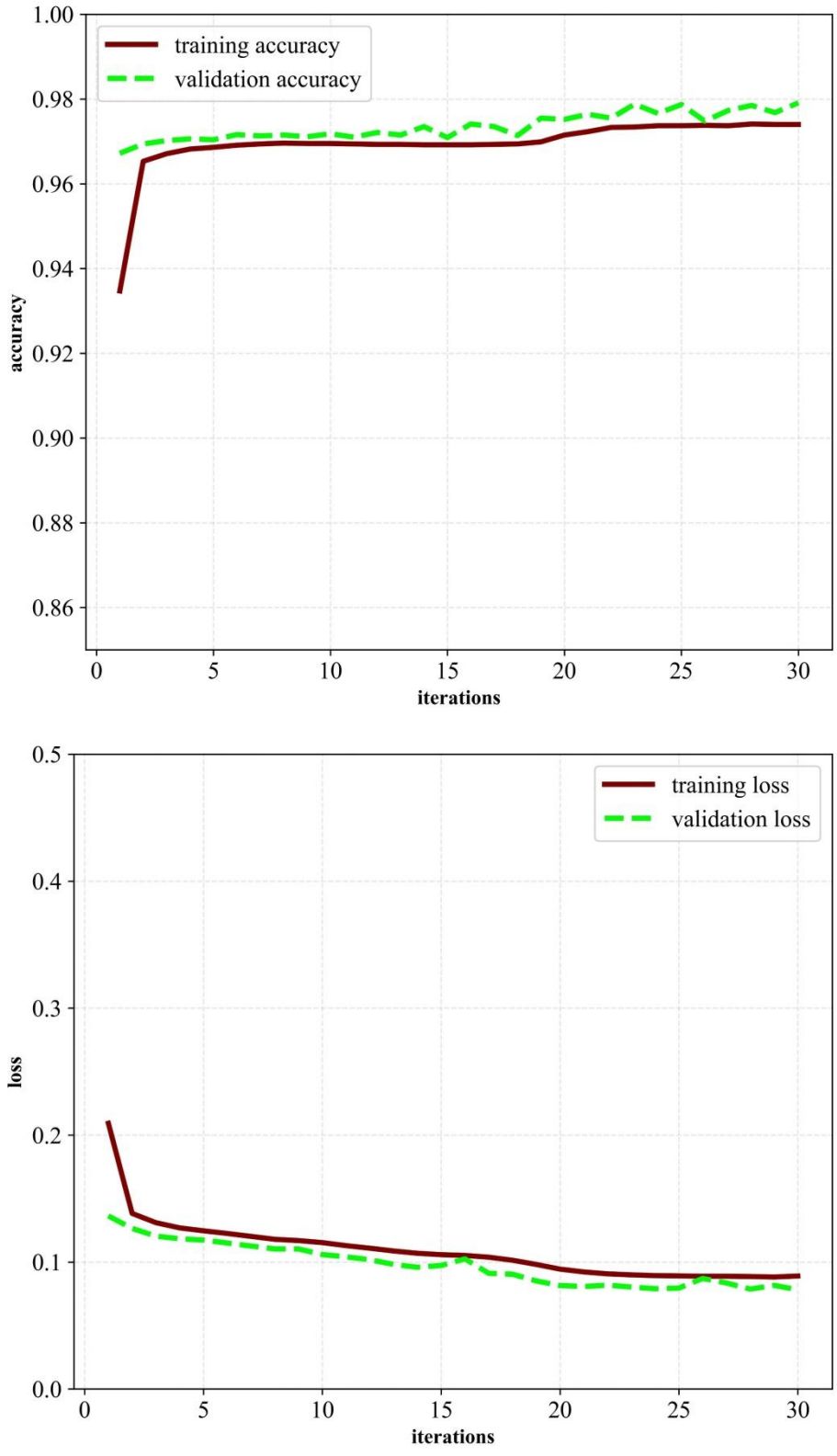


Figure 4.5. Training and validation accuracy and loss curves of the hybrid quantum–classical model using 4 selected features.

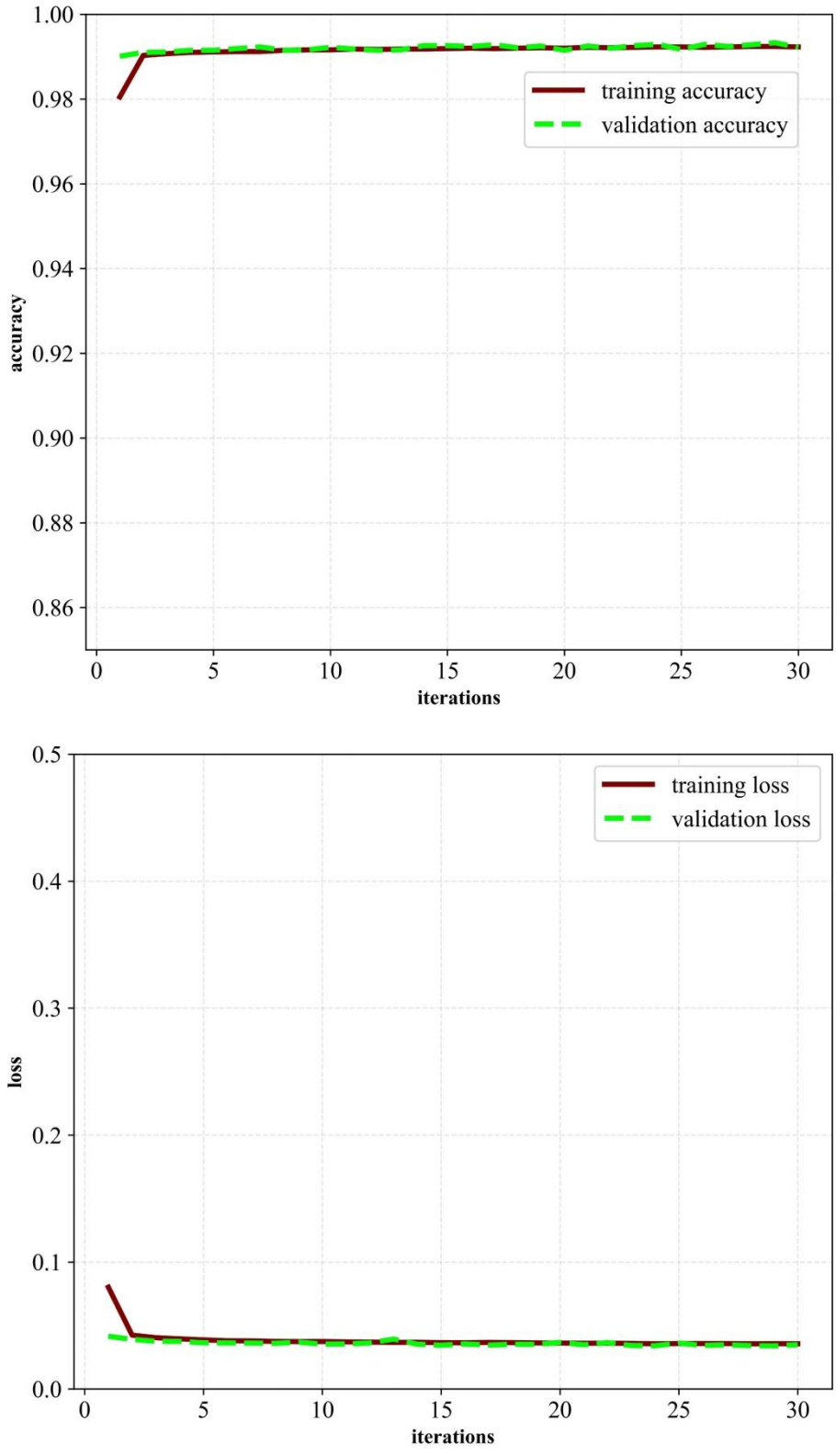


Figure 4.6. Training and validation accuracy and loss curves of the hybrid quantum–classical model using 12 selected features

Binary Classification using Pure Quantum Model (M4). In this experimental scenario, a fully quantum approach was adopted to evaluate the classification potential of a pure quantum model without any classical neural components. The twelve selected features were angle-encoded using $R_y(\pi x_i)$ gates, and passed through a three-layer PQC composed of R_z, R_x, and circular CNOT gates. The quantum output was produced by computing the expectation values of the Pauli-Z operators on each qubit, then averaging the results to form a scalar prediction. A sigmoid function was subsequently applied to this scalar to yield binary classification probabilities.

The results of the four binary classification models proposed in this study are presented in Table 4.8. The best-performing values are highlighted in bold to emphasize the top results across our models.

Table 4.8. Performance comparison of author of the used dataset and proposed binary models

Metric	**Ferreira et al.** [12]					**Our study**			
	DT	**LR**	**NB**	**RF**	**DNN**	**M1**	**M2**	**M3**	**M4**
Accuracy	86.01	84.51	88.60	86.91	**92.67***	89.80	**90.10**†	89.50	88.97
Recall	86.01	84.51	88.60	86.91	**92.67***	**90.00**†	**90.00**†	89.00	89.00
Precision	86.30	84.80	89.62	87.12	**93.55***	90.00	**91.00**†	90.00	**91.00**†
F1 score	85.98	84.48	88.53	86.89	**92.63***	**90.00**†	**90.00**†	89.00	89.00

Note: * = best results in original study, † = best results in proposed models.

As shown in Table 4.8, the DNN model proposed by Ferreira et al. achieved the highest performance among the original models across all metrics, with an accuracy of 92.67% and an F1 score of 92.63%. However, our proposed models—particularly M2—demonstrated highly competitive performance, achieving 90.10% accuracy and 90.00% F1 score. Notably, these results surpass not only traditional ML models such as DT, LR, NB, and RF in terms of accuracy, but also exhibit robust performance despite leveraging a quantum or hybrid architecture. The M2 model achieved 90.10% accuracy, which is 1.50 higher than the best-performing traditional model (NB) and only 2.57 lower than the DNN. In terms of F1 score, M2 reached 90.00%, exceeding NB by 1.47 and narrowing the gap with DNN to just 2.63. These results highlight the effectiveness and competitiveness of our approach in binary classification tasks, particularly within quantum-assisted learning frameworks. The confusion matrix of the M2 model is presented in Figure 4.7.

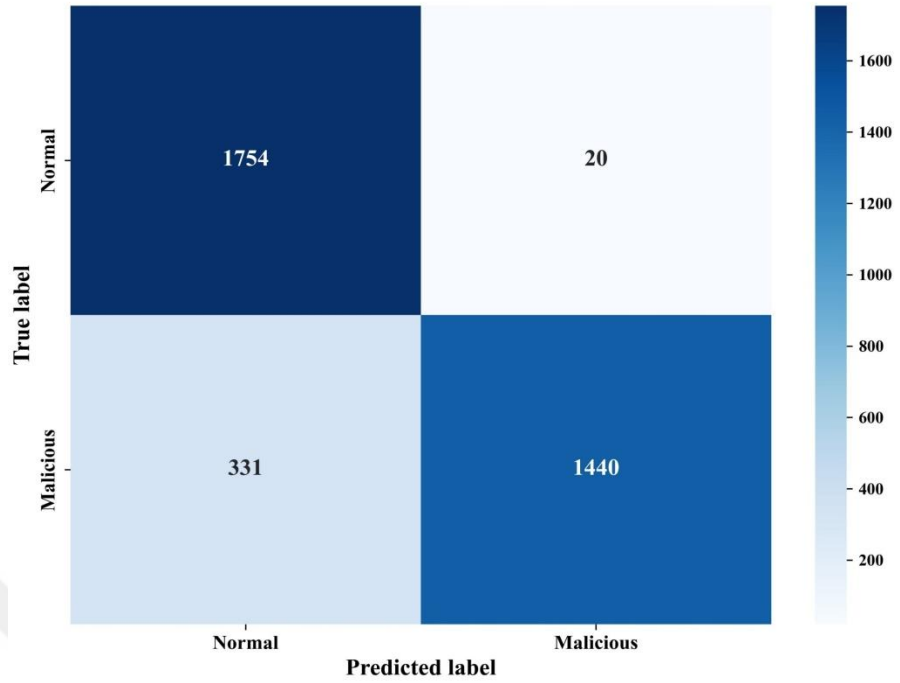


Figure 4.7. The confusion matrix of the M2 model

The model correctly classified 1754 out of 1774 normal instances, resulting in only 20 false positives, indicating strong performance in identifying normal traffic. On the other hand, it correctly identified 1440 out of 1771 malicious instances but misclassified 331 of them as normal, leading to a relatively higher false negative count.

Multiclass Classification

Multi-Class Classification using Hybrid Quantum-Classical Model (M5). To evaluate the scalability of the proposed QML framework in a multi-class intrusion detection scenario, this experiment employed a hybrid model integrating quantum feature encoding with a deep classical neural head. The input data consisted of twelve selected features normalized to [0, 1] and embedded using angle encoding via $R_y(\pi x_i)$ rotations. A three-layer PQC was used to entangle and transform the encoded features. The resulting expectation values were passed to two sequential Dense layers (32 and 64 units, respectively) with ReLU activations, culminating in a softmax-activated output layer corresponding to the eight intrusion classes. Figure 4.8 illustrates the training and validation accuracy and loss curves for this multiclass model.

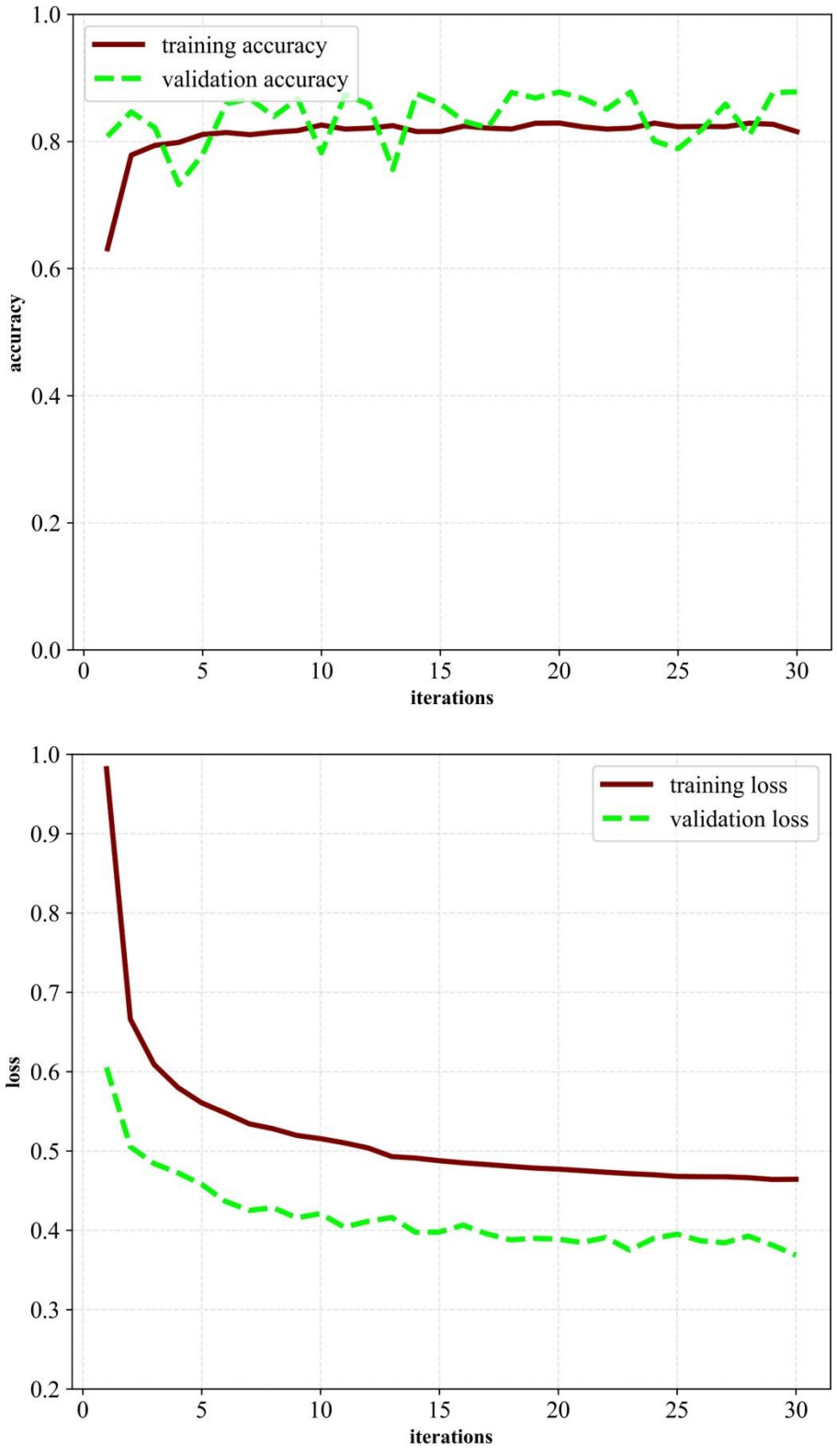


Figure 4.8. Training and validation accuracy and loss curves of the hybrid quantum–classical model

As observed in Figure 4.8, the training and validation accuracy and loss curves indicate a stable learning process. Although minor fluctuations in validation accuracy were observed

across epochs—likely due to class imbalance and label complexity—the model ultimately achieved a final validation accuracy of 92.4%, suggesting strong generalization. Meanwhile, the validation loss stabilized around 0.26, and the training loss steadily decreased, reflecting effective convergence and consistent optimization dynamics throughout training.

Multi-Class Classification using Quantum + Single Dense Layer Model (M6). This experiment investigates a simplified hybrid architecture designed to reduce classical post-processing while preserving the representational power of quantum-enhanced feature extraction. The twelve most informative features were embedded using angle encoding via $R_y(\pi x_i)$ rotations into a 12-qubit quantum circuit comprising three variational layers of R_z, R_x, and CNOT entanglement gates. The resulting quantum observables were directly passed to a single Dense layer with a softmax activation for eight-class classification.

The results of the original authors' models and the two multiclass classification models proposed in this study are presented in Table 4.9. The best-performing results from both the original authors' models and our models are highlighted in bold to facilitate direct comparison.

Table 4.9. Performance comparison of author of the used dataset and proposed multiclass models

Metric	Ferreira et al. [12]					Our study	
	DT	LR	NB	RF	DNN	M5	M6
Accuracy	75.81	75.37	60.80	**76.37***	74.96	**84.60†**	81.80
Recall	75.81	75.37	60.80	**76.37***	74.96	**85.00†**	82.00
Precision	81.45	74.51	77.68	**81.70***	78.70	**86.00†**	84.00
F1 score	**72.50***	70.20	57.93	72.30	69.18	**84.00†**	82.00

Note: * = best results in original study, † = best results in proposed models.

According to the results presented in Table 4.9**,** the proposed M5 model outperforms all baseline models in terms of accuracy, recall, precision, and F1 score. This demonstrates its effectiveness in multiclass classification tasks compared to the models reported by the original authors. Specifically, the M5 model achieves the highest accuracy (84.60%), recall (85.00%), precision (86.00%), and F1 score (84.00%), indicating its strong ability to correctly identify and classify multiple attack types. The M6 model also performs competitively, with an accuracy of 81.80% and F1 score of 82.00%, surpassing all baseline

models except M5. In contrast, the classical models presented by Ferreira et al.—particularly NB and DNN—exhibit lower performance, especially in recall and F1 score, which are crucial for reliable multiclass detection. These results highlight the advantage of the proposed QML-based models in terms of both predictive power and robustness when applied to complex IoT intrusion detection tasks. As the best-performing model, the confusion matrix of M5 is presented in Figure 4.9.

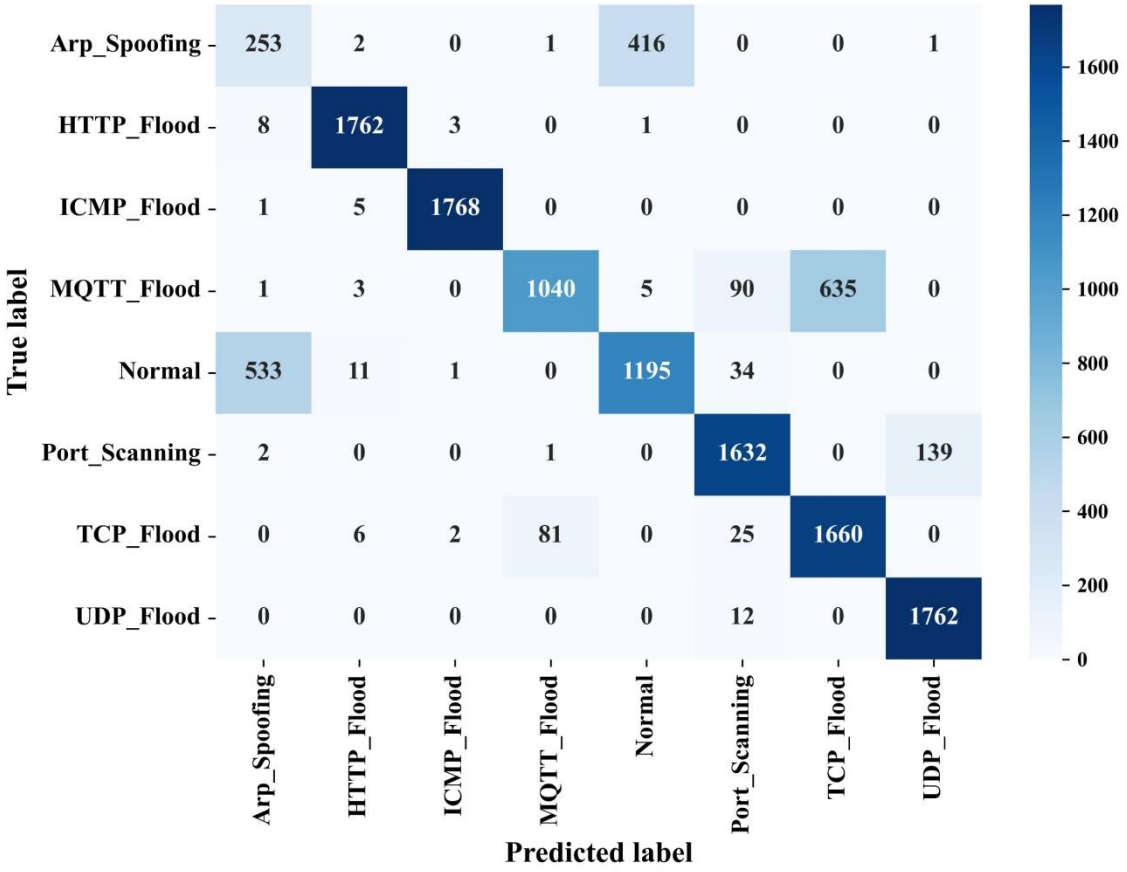


Figure 4.9. The confusion matrix of the M5 model

According to Figure 4.9, the M5 model correctly classifies the majority of attack types with high precision. For instance, it accurately predicts 1,762 out of 1,774 UDP_Flood, 1,762 out of 1,774 HTTP_Flood, and 1,768 out of 1,774 ICMP_Flood instances. However, the model misclassifies 416 out of 673 Arp_Spoofing samples as Normal, indicating significant confusion between these two classes. Similarly, among the MQTT_Flood samples, while 1,040 are correctly classified, 90 are predicted as Port_Scanning and 635 as Port_ Flood, reflecting notable overlap in feature representations. Despite these issues, strong performance is observed for Port_Scanning (1,632/1,774 correct) and TCP_Flood

(1,660/1,774 correct), suggesting reliable detection in those categories. Overall, the model performs well in distinguishing most classes but may benefit from refinement in separating attack types with closely related patterns, particularly between Arp_Spoofing, MQTT_Flood, and Normal traffic.

Training Efficiency and Resource Consumption

Table 4.10 summarizes the quantum encoding and training durations across six different model configurations. The comparison highlights how feature dimensionality and architectural complexity influence computational efficiency.

Table 4.10. Quantum encoding and training time comparison across models

Task	Model	Description	Encoding Times (s)	Training Time (s)
Binary	M1	Hybrid (4 features)	176.72	6,689.08
Binary	M2	Hybrid (12 features)	443.48	21,306.54
Binary	M3	Quantum + Single Dense Layer (12 features)	443.48	20,082.74
Binary	M4	Pure Quantum Model (12 features)	443.48	20,146.18
Multiclass	M5	Hybrid Deep Classical Layers (12 features)	343.66	14,528.58
Multiclass	M6	Quantum + Single Dense Layer (12 features)	304.75	14,458.22

As shown in Table 4.9., quantum encoding and training durations vary significantly based on model complexity and architectural choices:

- **M1** (Hybrid with 4 features) recorded the lowest total time, with fast encoding (176.72s) and short training time (6,689.08s), highlighting the efficiency of low-dimensional input.
- All 12-feature binary models (**M2–M4**) shared the same encoding time (443.48s), indicating that encoding time is primarily influenced by input dimensionality rather than architectural depth.
- **M3** and **M4**, which either reduced or eliminated classical layers, achieved shorter training durations compared to the full hybrid model **M2**, suggesting that classical post-processing contributes significantly to computational overhead.
- In the multiclass classification task, **M6** achieved the fastest training (14,458.22s) and encoding (304.75s), owing to its minimal classical structure.
- **M5**, with its deep classical layers, resulted in higher encoding (343.66s) and training times (14,528.58s), reinforcing the trade-off between model expressiveness and computational efficiency.

5. CONCLUSION

This thesis investigated the integration of classical and quantum machine learning (ML and QML) models to improve intrusion detection in IoT systems, with a particular focus on the Internet of Medical Things (IoMT) and Agricultural IoT (AG-IoT) domains. To address challenges such as high dimensionality, class imbalance, and dynamic traffic patterns, a dual-framework approach was proposed that combines the strengths of classical and quantum paradigms. In the classical ML pipeline, hyperparameter tuning was performed using the WaOA, resulting in significant performance gains. For example, in the IoMT experiments, XGBoost and DT classifiers exceeded 99% accuracy and F1-scores in both binary and multiclass tasks. Similarly, in the AG-IoT experiments, WaOA-optimized models achieved strong results, with XGBoost reaching 93.34% accuracy in binary classification and up to 72.05% F1-score in multiclass settings. On the quantum side, a hybrid QML architecture was proposed for the AG-IoT domain, combining PQC with classical dense layers. To reduce computational load and improve interpretability, a binary version of the SFOA was used for feature selection, followed by MI-based refinement. The best hybrid QML model achieved 90.10% accuracy and a 90.00% F1-score in binary classification, and 84.60% accuracy and 84.00% F1-score in the multiclass setting—outperforming the classical baselines on the same dataset. These findings suggest that QML can provide performance gains in complex IoT security scenarios when combined with effective feature selection. Despite these promising results, certain limitations remain. The quantum circuits were executed in idealized, noise-free simulators. The practical deployment of these models on real NISQ devices is currently constrained by short coherence times, gate noise, and limited qubit connectivity. Although shallow circuit designs (12 qubits, 24 parameters) were adopted, the overhead introduced by entanglement operations like CNOT gates remains non-trivial. Moreover, the scalability and resilience of QML models in noisy environments are yet to be thoroughly validated. Several critical directions for future research have been identified. One key priority is transitioning from simulation-based evaluations to real-world implementations by incorporating noise-aware training methodologies and utilizing quantum cloud platforms (e.g., IBMQ) or actual quantum hardware. This transition is vital for assessing the robustness and applicability of hybrid quantum-classical models under realistic, noisy conditions. Additionally, designing hardware-efficient quantum circuit architectures and

employing quantum transfer learning strategies could substantially reduce training complexity while enhancing model scalability and adaptability in high-dimensional IoT contexts. Moreover, integrating advanced techniques such as explainable artificial intelligence (XAI), federated learning, and real-time stream processing will be essential to address the growing demands for interpretability, data privacy, and responsiveness in dynamic and decentralized IoT security systems.

In conclusion, this thesis demonstrates that the integration of nature-inspired optimization with both classical and quantum learning approaches provides a compelling strategy for enhancing the security of IoT infrastructures. Classical models remain reliable and effective, especially when refined through evolutionary algorithms. Meanwhile, QML techniques hold distinct promise for managing the growing complexity and computational challenges of emerging IoT ecosystems. Fully unlocking this potential, however, will depend not only on advancements in quantum hardware but also on sustained innovation in algorithm design, rigorous performance evaluation, and the development of scalable deployment frameworks tailored to diverse IoT contexts.

REFERENCES

[1] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of Things (IoT): A literature review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, 2015.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, and A. A. Khan, "A review and state of art of Internet of Things (IoT)," *Archives of Computational Methods in Engineering*, pp. 1–19, 2021.

[4] L. Y. Rock, F. P. Tajudeen, and Y. W. Chung, "Usage and impact of the internet-of-things-based smart home technology: a quality-of-life perspective," *Universal Access in the Information Society,*, vol. 23, no. 1, pp. 345–364, 2024.

[5] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of Things in agriculture, recent advances and future challenges," *Biosystems Engineering*, vol. 164, pp. 31–48, 2017.

[6] "https://iot-analytics.com/, [Accessed Date: 25-10-2024]."

[7] "https://www.sonicwall.com/threat-report, [Accessed Date: 25-10-2024]."

[8] M. M. Rahman, S. Al Shakil, and M. R. Mustakim, "A survey on intrusion detection system in IoT networks," *Cyber Security and Applications*, vol. 3, p. 100082, 2025.

[9] E. Alpaydin, *Machine learning*. MIT press, 2021.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] J. Areia, I. Bispo, L. Santos, and R. L. de C. Costa, "IoMT-TrafficData: Dataset and tools for benchmarking intrusion detection in internet of medical things," *IEEE Access*, 2024.

[12] R. Ferreira, I. Bispo, C. Rabadão, L. Santos, and R. L. de C. Costa, "Farm-flow dataset: Intrusion detection in smart agriculture based on network flows," *Computers and Electrical Engineering*, vol. 121, p. 109892, 2025.

[13] R. Saadouni, C. Gherbi, Z. Aliouat, Y. Harbi, and A. Khacha, “Intrusion detection systems for IoT based on bio-inspired and machine learning techniques: a systematic review of the literature,” *Cluster Computing*, vol. 27, no. 7, pp. 8655–8681, 2024.

[14] Q. A. Al-Haija and A. Droos, “A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT),” *Expert Systems*, vol. 42, no. 2, p. e13726, 2025.

[15] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, “Anomaly-based intrusion detection system for IoT networks through deep learning model,” *Computers and Electrical Engineering*, vol. 99, p. 107810, 2022.

[16] S. M. Kasongo, “A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework,” *Computer Communications*, vol. 199, pp. 113–125, 2023.

[17] A. V Turukmane and R. Devendiran, “M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning,” *Computers & Security*, vol. 137, p. 103587, 2024.

[18] H. Nandanwar and R. Katarya, “Deep learning enabled intrusion detection system for Industrial IOT environment,” *Expert Systems with Applications*, vol. 249, p. 123808, 2024.

[19] H. Nandanwar and R. Katarya, “TL-BILSTM IoT: transfer learning model for prediction of intrusion detection system in IoT environment,” *International Journal of Information Security*, vol. 23, no. 2, pp. 1251–1277, 2024.

[20] M. A. Hossain and M. S. Islam, “Ensuring network security with a robust intrusion detection system using ensemble-based machine learning,” *Array*, vol. 19, p. 100306, 2023.

[21] A. Awajan, “A novel deep learning-based intrusion detection system for IOT networks,” *Computers*, vol. 12, no. 2, p. 34, 2023.

[22] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, “Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms,” *Sensors*, vol. 24, no. 2, p. 713, 2024.

[23] S. Racherla, P. Sripathi, N. Faruqui, M. A. Kabir, M. Whaiduzzaman, and S. A. Shah, “Deep-IDS: A Real-Time Intrusion Detector for IoT Nodes Using Deep Learning,” *IEEE Access*, 2024.

[24] S. Hizal, U. Cavusoglu, and D. Akgun, “A novel deep learning-based intrusion detection system for IoT DDoS security,” *Internet of Things*, vol. 28, p. 101336, 2024.

[25] A. Kaushik and H. Al-Raweshidy, “A novel intrusion detection system for internet of things devices and data,” *Wireless Networks*, vol. 30, no. 1, pp. 285–294, 2024.

[26] X. Wang, L. Dai, and G. Yang, “A network intrusion detection system based on deep learning in the IoT,” *The Journal of Supercomputing*, vol. 80, no. 16, pp. 24520–24558, 2024.

[27] A. V. Hanafi, A. Ghaffari, H. Rezaei, A. Valipour, and B. Arasteh, “Intrusion detection in Internet of things using improved binary golden jackal optimization algorithm and LSTM,” *Cluster Computing*, vol. 27, no. 3, pp. 2673–2690, 2024.

[28] B. Sharma, L. Sharma, C. Lal, and S. Roy, “Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach,” *Expert Systems with Applications*, vol. 238, p. 121751, 2024.

[29] O. A. Alkhudaydi, M. Krichen, and A. D. Alghamdi, “A deep learning methodology for predicting cybersecurity attacks on the internet of things,” *Information*, vol. 14, no. 10, p. 550, 2023.

[30] İ. Avcı and M. Koca, “Predicting ddos attacks using machine learning algorithms in building management systems,” *Electronics (Basel)*, vol. 12, no. 19, p. 4142, 2023.

[31] S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, “A novel attack detection scheme for the industrial internet of things using a lightweight random neural network,” *IEEE Access*, vol. 8, pp. 89337–89350, 2020.

[32] R. V Mendonca, J. C. Silva, R. L. Rosa, M. Saadi, D. Z. Rodriguez, and A. Farouk, “A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms,” *Expert Systems*, vol. 39, no. 5, p. e12917, 2022.

[33] S. Ullah, M. A. Khan, J. Ahmad, S. S. Jamal, Z. e. Huma, M. T. Hassan, N. Pitropakis, Arshad, and W. J. Buchanan, “HDL-IDS: A hybrid deep learning

architecture for intrusion detection in the Internet of Vehicles," *Sensors*, vol. 22, no. 4, p. 1340, 2022.

[34] S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Bouliia, A. Koubaa, S. U. Jan, and M. M. Iqbal Ch, "TNN-IDS: Transformer neural network-based intrusion detection system for MQTT-enabled IoT Networks," *Computer Networks*, vol. 237, p. 110072, 2023.

[35] G. Balhareth and M. Ilyas, "Optimized intrusion detection for IoMT networks with tree-based machine learning and filter-based feature selection," *Sensors*, vol. 24, no. 17, p. 5712, 2024.

[36] P. Kulshrestha and T. V Vijay Kumar, "Machine learning based intrusion detection system for IoMT," *International Journal of System Assurance Engineering and Management*, vol. 15, no. 5, pp. 1802–1814, 2024.

[37] E. Alalwany, B. Alsharif, Y. Alotaibi, A. Alfahaid, I. Mahgoub, and M. Ilyas, "Stacking Ensemble Deep Learning for Real-Time Intrusion Detection in IoMT Environments," *Sensors*, vol. 25, no. 3, p. 624, 2025.

[38] A. Berguiga, A. Harchay, and A. Massaoudi, "HIDS-IoMT: A deep Learning-Based intelligent intrusion detection system for the internet of medical things," *IEEE Access*, 2025.

[39] G. Lazrek, K. Chetioui, Y. Balboul, and S. Mazer, "An RFE/Ridge-ML/DL based anomaly intrusion detection approach for securing IoMT system," *Results in Engineering*, vol. 23, p. 102659, 2024.

[40] D. Praveena Anjelin and S. Ganesh Kumar, "An effective classification using enhanced elephant herding optimization with convolution neural network for intrusion detection in IoMT architecture," *Cluster Computing*, vol. 27, no. 9, pp. 12341–12359, 2024.

[41] M. Ibrahim and A. Al-Wadi, "Enhancing IoMT network security using ensemble learning-based intrusion detection systems," *Journal of Engineering Research*, 2024.

[42] N. Faruqui, M. A. Yousuf, M. Whaiduzzaman, A. K. M. Azad, S. A. Alyami, P. Liò, M. A. Kabir, and M. A. Moni, "SafetyMed: A novel IoMT intrusion detection

system using CNN-LSTM hybridization," *Electronics (Basel)*, vol. 12, no. 17, p. 3541, 2023.

[43] K. Gupta, D. K. Sharma, K. D. Gupta, and A. Kumar, "A tree classifier based network intrusion detection model for Internet of Medical Things," *Computers and Electrical Engineering*, vol. 102, p. 108158, 2022.

[44] G. Akar, S. Sahmoud, M. Onat, Ü. Cavusoglu, and E. Malondo, "L2D2: A Novel LSTM Model for Multi-Class Intrusion Detection Systems in the Era of IoMT," *IEEE Access*, 2025.

[45] R. Y. Aburasain, "Enhanced black widow optimization with hybrid deep learning enabled intrusion detection in Internet of Things-based smart farming," *IEEE Access*, vol. 12, pp. 16621–16631, 2024.

[46] K. Kethineni and G. Pradeepini, "Intrusion detection in internet of things-based smart farming using hybrid deep learning framework," *Cluster Computing*, vol. 27, no. 2, pp. 1719–1732, 2024.

[47] K. Zidi, K. Ben Abdellafou, A. Aljuhani, O. Taouali, and M. F. Harkat, "Novel intrusion detection system based on a downsized kernel method for cybersecurity in smart agriculture," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108579, 2024.

[48] A. Raghuvanshi, U. K. Singh, G. S. Sajja, H. Pallathadka, E. Asenso, M. Kamal, A. Singh, and K. Phasinam, "Intrusion detection using machine learning for risk mitigation in IoT-enabled smart irrigation in smart farming," *Journal of Food Quality*, vol. 2022, no. 1, p. 3955514, 2022.

[49] H. Zhou, H. Zou, P. Zhou, Y. Shen, D. Li, and W. Li, "CBCTL-IDS: A Transfer Learning-Based Intrusion Detection System Optimized With the Black Kite Algorithm for IoT-Enabled Smart Agriculture," *IEEE Access*, vol. 13, pp. 46601–46615, 2025.

[50] R. Saadouni, C. Gherbi, Z. Aliouat, Y. Harbi, A. Khacha, and H. Mabed, "Securing smart agriculture networks using bio-inspired feature selection and transfer learning for effective image-based intrusion detection," *Internet of Things*, vol. 29, p. 101422, 2025.

[51] S. Li, Z. Wang, S. Yang, X. Luo, D. He, and S. Chan, “Internet of Things intrusion detection: Research and practice of NSENet and LSTM fusion models,” *Egyptian Informatics Journal*, vol. 26, p. 100476, 2024.

[52] Z. Li and W. Yao, “A two stage lightweight approach for intrusion detection in Internet of Things,” *Expert Systems with Applications*, vol. 257, p. 124965, 2024.

[53] H. Ghasemi and S. Babaie, “A new intrusion detection system based on SVM–GWO algorithms for Internet of Things,” *Wireless Networks*, vol. 30, no. 4, pp. 2173–2185, 2024.

[54] M. Hdaib, S. Rajasegarar, and L. Pan, “Quantum deep learning-based anomaly detection for enhanced network security,” *Quantum Machine Intelligence*, vol. 6, no. 1, p. 26, 2024.

[55] R. Kumar and M. Swarnkar, “QuIDS: A Quantum Support Vector machine-based Intrusion Detection System for IoT networks,” *Journal of Network and Computer Applications*, vol. 234, p. 104072, 2025.

[56] A. Bellante, T. Fioravanti, M. Carminati, S. Zanero, and A. Luongo, “Evaluating the potential of quantum machine learning in cybersecurity: A case-study on PCA-based intrusion detection systems,” *Computers & Security*, p. 104341, 2025.

[57] M. Al-Hawawreh and M. S. Hossain, “A Human-Centered Quantum Machine Learning Framework for Attack Detection in IoT-Based Healthcare Industry 5.0,” *IEEE Internet Things Journal*, 2025.

[58] M. Kalinin and V. Krundyshev, “Security intrusion detection using quantum machine learning techniques,” *Journal of Computer Virology and Hacking Techniques*, vol. 19, no. 1, pp. 125–136, 2023.

[59] E. I. Elsedimy, H. Elhadidy, and S. M. M. Abohashish, “A novel intrusion detection system based on a hybrid quantum support vector machine and improved Grey Wolf optimizer,” *Cluster Computing*, vol. 27, no. 7, pp. 9917–9935, 2024.

[60] D. Said, “Quantum computing and machine learning for cybersecurity: Distributed denial of service (DDoS) attack detection on smart micro-grid,” *Energies (Basel)*, vol. 16, no. 8, p. 3572, 2023.

[61] N. O. Aljehane, H. A. Mengash, S. B. H. Hassine, F. A. Alotaibi, A. S. Salama, and S. Abdelbagi, “Optimizing intrusion detection using intelligent feature selection

with machine learning model," *Alexandria Engineering Journal*, vol. 91, pp. 39–49, 2024.

[62] A. S. Rajawat, S. B. Goyal, P. Bedi, T. Jan, M. Whaiduzzaman, and M. Prasad, "Quantum machine learning for security assessment in the internet of medical things (IoMT)," *Future Internet*, vol. 15, no. 8, p. 271, 2023.

[63] S. N. Makhadmeh, S. Fraihat, M. Awad, Y. Sanjalawe, M. A. Al-Betar, and M. A. Awadallah, "A crossover-integrated Marine Predator Algorithm for feature selection in intrusion detection systems within IoT environments," *Internet of Things*, p. 101536, 2025.

[64] "https://www.freepik.com/, [Accessed Date: 08-05-2025].".

[65] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J*, vol. 1, no. 1, pp. 22–32, 2014.

[66] A. A. El-Saleh, A. M. Sheikh, M. A. M. Albreem, and M. S. Honnurvali, "The internet of medical things (IoMT): opportunities and challenges," *Wireless Networks*, vol. 31, no. 1, pp. 327–344, 2025.

[67] F. Shrouf and G. Miragliotta, "Energy management based on Internet of Things: practices and framework for adoption in production management," *Journal of Cleaner Production*, vol. 100, pp. 235–246, 2015.

[68] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J*, vol. 4, no. 5, pp. 1250–1258, 2017.

[69] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "IoT Privacy and security: Challenges and solutions," *Applied Sciences*, vol. 10, no. 12, p. 4102, 2020.

[70] R. R. Krishna, A. Priyadarshini, A. V Jha, B. Appasani, A. Srinivasulu, and N. Bizon, "State-of-the-art review on IoT threats and attacks: Taxonomy, challenges and solutions," *Sustainability*, vol. 13, no. 16, p. 9463, 2021.

[71] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[72] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, “Metaheuristic algorithms: A comprehensive review,” *Computational intelligence for multimedia big data on the cloud with engineering applications*, pp. 185–231, 2018.

[73] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, “Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019),” *IEEE Access*, vol. 9, pp. 26766–26791, 2021.

[74] R. Narayanan and N. Ganesh, “A Comprehensive Review of Metaheuristics for Hyperparameter Optimization in Machine Learning,” *Metaheuristics for Machine Learning: Algorithms and Applications*, pp. 37–72, 2024.

[75] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-international conference on neural networks*, ieee, 1995, pp. 1942–1948.

[76] K. V Price, “Differential evolution,” in *Handbook of optimization: From classical to modern approach*, Springer, 2013, pp. 187–214.

[77] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2007.

[78] D. Karaboga and B. Akay, “A comparative study of artificial bee colony algorithm,” *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[79] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[80] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[81] S. Mirjalili, “SCA: a sine cosine algorithm for solving optimization problems,” *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[82] B. Abdollahzadeh, N. Khodadadi, S. Barshandeh, P. Trojovský, F. S. Gharehchopogh, E.-S. M. El-kenawy, L. Abualigah, and S. Mirjalili, “Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning,” Cluster Computing, vol. 27, no. 4, pp. 5235–5283, 2024.

[83] E.-S. M. El-Kenawy, N. Khodadadi, S. Mirjalili, A. A. Abdelhamid, M. M. Eid, and A. Ibrahim, “Greylag goose optimization: nature-inspired optimization algorithm,” *Expert Systems with Applications*, vol. 238, p. 122147, 2024.

[84] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, “The arithmetic optimization algorithm,” *Computer Methods in Applied Mechanics and Engineering*, vol. 376, p. 113609, 2021.

[85] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, “Marine Predators Algorithm: A nature-inspired metaheuristic,” *Expert Systems with Applications*, vol. 152, p. 113377, 2020.

[86] C. Zhong, G. Li, Z. Meng, H. Li, A. R. Yildiz, and S. Mirjalili, “Starfish optimization algorithm (SFOA): a bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers,” *Neural Computing and Applications*, vol. 37, no. 5, pp. 3641–3683, 2025.

[87] P. Trojovský and M. Dehghani, “A new bio-inspired metaheuristic algorithm for solving optimization problems based on walruses behavior,” *Scientific Reports*, vol. 13, no. 1, p. 8775, 2023.

[88] B. Liu, “Supervised learning,” in *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Springer, 2011, pp. 63–132.

[89] H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, “Unsupervised learning based on artificial neural network: A review,” in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, IEEE, 2018, pp. 322–327.

[90] S.-S. Learning, “Semi-supervised learning,” *CSZ2006. html*, vol. 5, no. 2, p. 1, 2006.

[91] I. D. Mienye and N. Jere, “A survey of decision trees: Concepts, algorithms, and applications,” *IEEE Access*, 2024.

[92] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[93] I. Rish, “An empirical study of the naive Bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Seattle, USA, 2001, pp. 41–46.

[94] L. Breiman, “Random forests,” *Mach Learning*, vol. 45, pp. 5–32, 2001.

[95] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[96] J. D. Martín-Guerrero and L. Lamata, “Quantum machine learning: A tutorial,” *Neurocomputing*, vol. 470, pp. 457–461, 2022.

[97] K. A. Tychola, T. Kalampokas, and G. A. Papakostas, “Quantum machine learning—an overview,” *Electronics (Basel)*, vol. 12, no. 11, p. 2379, 2023.

[98] A. Zeguendry, Z. Jarir, and M. Quafafou, “Quantum machine learning: A review and case studies,” *Entropy*, vol. 25, no. 2, p. 287, 2023.

[99] H. Riel, “Quantum computing technology,” in *2021 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2021, pp. 1–3.

[100] C. P. Williams and C. P. Williams, “Quantum gates,” *Explorations in quantum computing*, pp. 51–122, 2011.

[101] R. Kharsa, A. Bouridane, and A. Amira, “Advances in quantum machine learning and deep learning for image classification: A survey,” *Neurocomputing*, vol. 560, p. 126843, 2023.

[102] H. Bhavsar and M. H. Panchal, “A review on support vector machine for data classification,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 10, pp. 185–189, 2012.

CURRICULUM VITAE

Personal Information

Surname, name

Nationality

Date and place of birth

Telefon

e-mail

Education

Degree	Department	University	Graduation Year
M.Sc	Computer Engineering	Sivas University of Science and Technology	2025
B.Sc	Computer Engineering	Sivas Cumhuriyet University	2023

Academic Titles

Year	Department	University	Position
2024-present	Computer Engineering	Sivas University of Science and Technology	Res. Asst.

Foreign Language

English

Publications

Gul, M. F., Bakır, H. (2025). GA-ML: enhancing the prediction of water electrical conductivity through genetic algorithm-based end-to-end hyperparameter tuning. *Earth Science Informatics*, *18*(2), 191.

Arslan, S., Zoralioglu, Y., **Gul, M. F.** (2025). Comparative Analysis Of African Vultures Optimization Algorithm With Current Metaheuristics. *Osmaniye Korkut Ata Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *8*(1), 325-352.

Gul, M. F., Bakir, H. (2024, September). Improving Attack Detection in IoV Systems using GA-based Hyperparameter Optimization. In *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)* (pp. 1-5). IEEE.

Gul, M. F., Arslan, S. (2023, July). Mackey-glass time series prediction with immune plasma programming. In *2023 31st Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

Zoralioglu, Y., **Gul, M. F.**, Azizoğlu, F., Azizoğlu, G., Toprak, A. N. (2023, October). Predicting academic performance of students using machine learning techniques. In *2023 innovations in intelligent systems and applications conference (ASYU)* (pp. 1-6). IEEE.

Dikbas, S., Arslan, S., **Gul, M. F.**, Selcuklu, S. B. (2023, November). Electricity Price Forecasting Using Automatic Programming Methods. In *The International Conference on Artificial Intelligence and Applied Mathematics in Engineering* (pp. 192-203). Cham: Springer Nature Switzerland.

Gul, M. F., Bakır, H. (2025, June). Walrus optimization-enhanced machine learning intrusion detection for the Internet of Medical Things. Paper accepted at the *9th International Symposium on Innovative Approaches in Smart Technologies (ISAS 2025)*, Gaziantep, Türkiye.

Gul, M. F., Arslan, S., Selcuklu, S. B. (under review). A novel immune plasma-based automatic programming method for electricity price forecasting in the Turkish market.

Gul, M. F., and Bakır, H. (under review). Metaheuristic-Driven Two-Stage Hybrid Feature Selection and Deep Learning Optimization for Security in IoT Systems.

Gul, M. F., and Bakır, H. (submitted). A Novel Metaheuristic-Enhanced Quantum-Classical Neural Network for Attack Detection in Agriculture IoT Systems.


SİVAS
BİLİM VE TEKNOLOJİ
ÜNİVERSİTESİ
KÖKLERDEN GÖKLERE...