

Novel Resource Provisioning and Lightweight Security Protocols for IoT Edge Networks

by

AMAR ALMAINI



A thesis submitted in partial fulfilment of the requirements
of Edinburgh Napier University, for the award of

Doctor of Philosophy

School of Computing
Edinburgh Napier University
May 2024

Author's declaration

I, Amar Almaini, hereby confirm that this thesis submitted for assessment is my own work. All external sources, whether in the form of ideas, equations, figures, text, tables, programs, or any other form, have been duly acknowledged and referenced. This work has not been submitted previously for any other degree or professional qualification.

Name: Amar Almaini

Matriculation Number: 


SIGNED: DATE: 23.05.2024

Acknowledgements

First and foremost, I dedicate this thesis to the cherished memory of my father. His unwavering belief in my potential has been a guiding force behind this significant academic endeavor. While he may no longer be with us, his faith in me paved the way for this journey. I also extend heartfelt gratitude to my mother and sister. Their constant support and encouragement have been pillars of strength throughout my academic journey. Without them, this accomplishment would remain a distant dream. Special thanks are due to my director of studies, Prof. Ahmed Al-Dubai. His accessibility during challenging times and his insights about my research and writing have been invaluable. While he ensured this thesis remained a reflection of my work, his timely guidance steered me in the right direction whenever necessary. I am deeply grateful for his wisdom, patience, enthusiasm, and unwavering encouragement. Prof. Dr. Martin Schramm deserves my profound appreciation. His consistent guidance and invaluable feedback have significantly enriched this work. His support and understanding have been instrumental to the progression of this thesis. I extend my gratitude to Dr. Imed Romdhani for his consistent guidance and invaluable insights throughout this journey. A heartfelt acknowledgment goes to my colleagues at the ProtectIT institute at the Deggendorf Institute of Technology. I am especially indebted to those who have supported me over the years, contributed to my research, and co-authored scientific publications. I appreciate our enlightening scientific and technical discussions and thank those who assisted in the implementation work, with special mentions to Jakob Folz and Tobias Koßmann.

Lastly, my deepest thanks and love are for my daughter, Maya. Her presence has been my most significant motivation, reminding me never to give up throughout my studies.

Abstract

This Ph.D. thesis introduces a novel dynamic resource allocation framework tailored for Edge Computing (EC) in Internet of Things (IoT) networks, addressing the pressing challenges posed by resource limitations and escalating user demands. Edge-driven IoT networks, characterized by their reliance on locally available computational resources from a heterogeneous ensemble of devices such as sensors, vehicles, and mobile phones, present unique challenges. These resources, in contrast to their cloud counterparts, exhibit inherent variability in terms of processing power, distribution, and operating system diversity. Moreover, their connectivity is subject to fluctuations, including failures, intermittent connections, and unpredictable network entry and exit events, rendering the EC network inherently dynamic. The inadequacy of existing solutions to effectively manage the dynamic nature of resource availability at the edge underscores the necessity for a resource allocation framework capable of adapting to these dynamic conditions. To this end, we propose a dynamic resource allocation framework that dynamically assigns computational and network resources. This framework aims to minimize average service delays and achieve resource utilization balance at the edge. To realize this objective, two resource allocation models are developed using TensorFlow: a classification-based approach and a regression-based approach. Experimental results in dynamic environments demonstrate remarkable performance improvements, with the regression model achieving an 87% task completion rate within specified time constraints and the classification model achieving 56%. To underscore the practicality and efficiency of our proposed framework, two real-world use cases are explored. The first use case deals with the detection of spoofing attacks in autonomous vehicles (AVs) using Shadow-

Analyzer, a technique that identifies ghost object attacks with reduced 2D data derived from 3D point cloud information. The second use case focuses on the implementation of homomorphic encryption for secure communication, presenting a novel distributed approach to Fully Homomorphic Encryption (FHE)-based data processing. To validate the applicability and efficiency of our framework, extensive simulation experiments are conducted across various scenarios and operational conditions on a hardware testbed. These experiments yield promising results, establishing the viability of our dynamic resource allocation framework in addressing the dynamic challenges posed by resource availability at the edge in IoT networks.

Acronyms

AI Artificial Intelligence 6

AUC Area Under the Curve 152, 154, 156–158

BFV Brakerski-Fan-Vercauteren 23, 48, 113, 130

BGV Brakerski-Gentry-Vaikuntanathan 23, 113, 130

CC Cloud Computing 1, 2, 5, 7, 9, 10, 13, 25, 27–29, 54, 112, 162

CKKS Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN)
23, 48, 113, 130

CLI Command Line Interface 104

CLRD Critical Low Range Distance 141

CNN Convolutional Neural Network 139, 140, 142, 143, 154–158

DHE Distributed Homomorphic Encryption 3, 112, 113, 175

DQN Deep-Q Network 33, 36, 37

DRA Dynamic Resource Allocation 2, 16, 26

EC Edge Computing 1–3, 7–18, 21, 23, 25, 27–32, 34, 35, 38, 39, 52–54, 56, 93,
94, 111, 112, 131, 132, 160–162, 174–176, 180

FHE Fully Homomorphic Encryption 20, 23, 32, 46–49, 112, 113, 130

FPR False Positive Rate 151, 152

FSM Finite State Machine 98, 101

- GNN** Graph Neural Network 140
- HE** Homomorphic Encryption 52, 112, 113, 117
- IoT** Internet of Things 1, 2, 6, 7, 9, 11, 13, 14, 16, 18, 19, 23, 24, 33–35, 47, 48, 55, 56, 111, 162, 173, 174
- LiDAR** Light Detection and Ranging x, 3, 12, 16, 17, 19–21, 23, 49–52, 131–140, 142, 149, 150, 152, 153, 159, 160, 175
- MEC** Mobile Edge Computing 33, 35
- ML** Machine Learning viii, 2, 6, 26, 32, 55, 56
- NN** Neural Network viii, xiii, 3, 18, 19, 56–59, 61, 64, 66, 71, 72, 174, 175
- OTP** One-Time Password 176, 177
- P4** programming protocol-independent packet processors 3, 23, 39, 45, 46, 94–96, 98, 101, 102, 104, 111, 175, 176
- QoS** Quality of Service 2, 28, 29, 36, 45, 55
- ROC** Receiver Operating Characteristic 151, 152
- SDN** Software Defined Networks viii, 3, 8, 21–24, 35, 37, 39, 45, 46, 94–97, 153, 163, 165, 166, 173, 175
- SVMs** Support Vector Machines 147
- TMS** Traffic Management Systems 12, 16–18
- TPR** True Positive Rate 151, 152
- V2T** Vehicle-to-Thing 19, 20, 23, 160, 161, 175
- V2V** Vehicle-to-Vehicle 50
- V2X** Vehicle-to-Everything 3, 36, 132
- VM** Virtual Machine 27, 29

Contents

Author's declaration	i
Acknowledgements	ii
Abstract	iii
Acronyms	v
List of Figures	xiii
List of Tables	xvii
List of Publications	xx
1 Introduction	1
1.1 Cloud Computing	5
1.1.1 Resource Management in Cloud Computing	5
1.2 Edge Computing	6
1.2.1 Resource Management in Edge Computing	7
1.2.2 Computation Offloading	7
1.2.3 Network Management	8
1.2.4 Security and Privacy	9
1.2.5 Edge Computing Applications	10
1.3 Motivation	13
1.4 Problem Statement and Research Question	14
1.5 Aims and Objectives	16
1.6 Contributions	18
1.7 Contributions Complementarity and Justification	21

1.8 Thesis Structure	22
2 Literature Review	25
2.1 Introduction	25
2.2 Resource Management in Cloud Computing	25
2.3 Resource Management in Edge Computing	27
2.3.1 Resource Allocation	28
2.3.2 Resource Placement	28
2.3.3 Resource Provisioning	29
2.3.4 Resource Pooling	30
2.4 Security and Privacy	30
2.4.1 Attack Detection	30
2.4.2 Data Integrity	31
2.4.3 Access Control	31
2.4.4 Privacy	32
2.5 Developments in Resource Management for Edge Computing . . .	32
2.6 Resource Management through Machine Learning	38
2.7 Developments in Edge Security for SDN	39
2.8 Developments in Homomorphic Encryption	46
2.9 Developments in the Detection of Spoofing Attacks in Autonomous Driving	49
2.10 Summary	52
3 Neural Network-Driven Resource Management in Edge Computing	53
3.1 Introduction	53
3.2 Traditional Techniques vs. ML Techniques for Resource Allocation	55
3.2.1 Challenges with Conventional Resource Allocation Techniques	55
3.2.2 Why Do We Need ML/NN	56
3.3 AI Resource Allocation Structure	57
3.3.1 State	58
3.3.2 Action Space	60
3.3.3 Network Setup	61

3.4	AI-Models	61
3.4.1	Classifier Model	63
3.4.2	Regression Model	65
3.5	Primary Metrics	67
3.5.1	Reward	67
3.5.2	Loss	70
3.5.3	Priority Fulfilled	70
3.6	The experimental setup	71
3.7	Training	71
3.7.1	Training setup	72
3.7.2	Fine-Tuning Model Parameters	74
3.8	Performance Comparison of Models in Dynamic Environments . . .	86
3.8.1	Network Adaptation for Optimized Dynamic Testing	87
3.8.2	Performance Analysis and Evaluation of Dynamic Behavior	89
3.9	Summary	93
4	Efficient Edge Authentication for Software-Defined Networks	94
4.1	Introduction	94
4.2	Delegating Authentication to SDN Switches	95
4.3	Centralized vs. distributed approach	97
4.4	Modeling Deterministic Port Knocking Authentication using Finite State Machine	98
4.5	Authentication and Port Knocking Implementation for Secure Net- work Access	99
4.5.1	Dynamic Switch Reconfiguration for Enhanced Authentica- tion and Network Security	102
4.5.2	Enhancing Network Security: Authentication-based Defense Against Port Scans	107
4.6	The experimental setup	107
4.7	Evaluation	108
4.7.1	Switch Performance Analysis: Evaluating Time Window Va- lidity Check	108

4.7.2	Switch Performance: Evaluating the Impact of Authentication Functionality	110
4.8	Summary	111
5	Scalable Distributed Homomorphic Encryption for Complex Computational Models	112
5.1	Introduction	112
5.2	Centralized vs. Distributed Approaches in Homomorphic Encrypted Data Processing	113
5.2.1	Centralized Approach	114
5.2.2	Distributed Approach	115
5.3	The experimental setup	116
5.4	Evaluation	116
5.4.1	Distributed Homomorphic Computation with OpenMPI and SEAL	117
5.4.2	Homomorphic Encryption Schemes	117
5.4.3	Time Analysis of Processing Steps	119
5.5	Experimental Evaluation of Homomorphic Operations	119
5.5.1	Comparative Evaluation of Homomorphic Cryptographic Schemes	120
5.5.2	Comparing Performance: Centralized vs. Distributed Approaches	121
5.6	Discussion and Analysis	129
5.7	Summary	130
6	Shadow-Analyzer: Efficient Detection of Ghost Objects in Autonomous Driving using Neural Networks	131
6.1	Introduction	131
6.2	Fundamentals and Vulnerabilities of LiDAR Sensing Technology	132
6.2.1	Feasibility of Injecting False Points in LiDAR Systems	134
6.2.2	Attack Distance Feasibility	136
6.3	Attack Goal and Threat Model	137

6.4	2D Shadow-Analyzer approach	139
6.4.1	3D Object Detector	140
6.4.2	2D Bird's-eye View Generator	141
6.4.3	Shadow Verification	142
6.5	Data Reduction	143
6.6	Shadow-Analyzer Models	143
6.6.1	CNN-Sigmoid	144
6.6.2	CNN-Linear	146
6.7	The experimental setup	148
6.8	Training	149
6.9	Metrics for performance measurement	151
6.10	Evaluation	152
6.10.1	CNN-Sigmoid	154
6.10.2	CNN-Linear	156
6.11	Invalidation Attack	159
6.12	Summary	160
7	Empirical Validation of Usability: Extensive Evaluation of Our Approach in an Edge Computing Hardware Testbed	162
7.1	Introduction	162
7.2	Testbed Setup for Resource Allocation Validation	163
7.2.1	Hardware Resources	163
7.2.2	Virtualized Environments	165
7.2.3	SDN-Based Resource Allocation and Client Request Flow	166
7.3	Performance Benchmarking Using a Single Local Device	167
7.4	Remote Device Evaluation at the Edge and its Communication Delays	168
7.5	Resource Allocation Between Jetson and Raspberry Pi	169
7.6	Image Distribution and Processing Time with Four Edge-located Resources	171
7.7	Summary	173
8	Conclusions and future work	174
8.1	Thesis Summary and Objectives Review	174

8.2	Future Directions	176
8.2.1	Secure Authentication with Switch-based One-Time Pass- words	176
8.2.2	Queue Management	178
8.2.3	Deep-Q-Network Approach	179
8.2.4	Federated Learning Applicability Evaluation	179
8.3	Concluding Remarks	180
	literature	181

List of Figures

Figure 1.1 Conceptual Framework Illustrating the Interconnections Among Various Topics	4
Figure 3.1 Hierarchical Structure for Resource Allocation in the Edge – An Overview	54
Figure 3.2 Schematic representation of the AI-based resource allocation system model	58
Figure 3.3 Schematic of Network Architecture and Connectivity Setup	61
Figure 3.4 Structure of the classification model, NN architecture and input/output layers	64
Figure 3.5 Illustration of the classification model workflow utilizing nine resources	65
Figure 3.6 Structure of the regression model, NN architecture and input/output layers	66
Figure 3.7 Illustration of the regression model workflow utilizing nine resources	67
Figure 3.8 Workflow for training the AI agent in the proposed system .	72
Figure 3.9 Hyperparameter Tuning Process: A Flowchart Illustrating the Steps for Determining Optimal Hyperparameters	74
Figure 3.10 Impact of Normalization on Reward and Loss in Model Training	76
Figure 3.11 Relationship between the Number of Neurons, Reward, and Percentage of Fulfilled Priorities in the Classification Model	77
Figure 3.12 Relationship between the Number of Neurons, Reward, and Percentage of Fulfilled Priorities in the regression Model	78
Figure 3.13 Impact of Neuron Distribution on Priority Fulfillment in Classification Models	80

Figure 3.14 Impact of Neuron Distribution on Priority Fulfillment in Regression Models	82
Figure 3.15 Impact of Learning Rate on Priority Fulfillment in Classification Models	84
Figure 3.16 Impact of Learning Rate on Priority Fulfillment in Regression Models	86
Figure 3.17 Comparison of Reward Functions in Dynamic Environment: Time Evolution of Rewards in Two Models	89
Figure 3.18 Evaluation of the model's ability to select resources matching the task's requirements	91
Figure 3.19 Comparative Analysis of Model and Reward Function Performance (Green denotes first test with the lowest difficulty level, yellow is the second with moderate difficulty level, and red represents the third and most challenging test. The average performance across all tests is represented by the orange curve)	92
Figure 4.1 State Transition Diagram of Deterministic Port Knocking Authentication Using FSM	99
Figure 4.2 A Visual Representation of the Reference Topology	100
Figure 4.3 Refining the Hit/Miss construct: Identifying trustworthiness in node selection	104
Figure 4.4 CLI commands to set the initial knock-sequence	105
Figure 4.5 CLI commands to change the knock-sequence	105
Figure 4.6 Visualizing H2's Port Knocking Technique: Wireshark reveals the port-sequence manipulation for secure ticket authentication	106
Figure 4.7 Visualizing H2's Port Knocking Technique: Wireshark reveals the new port-sequence manipulation for secure ticket authentication	106
Figure 4.8 Comparing Throughput of H2 and H3 TCP Streams to H4 (H2: Dotted Curve, H3: Solid Curve) with H2 Authentication Delayed Initially	109
Figure 4.9 Doubled Instances of Ticket Expiration Within a Two-Minute Frame (Validity Time of 50 Seconds)	109

Figure 4.10 Average throughput for fifty measurements in different scenarios	110
Figure 5.1 Centralized model	114
Figure 5.2 Distributed model	116
Figure 5.3 Composition of different used times	118
Figure 5.4 Comparison of centralized and distributed approaches at different vector sizes with BVF scheme.	126
Figure 5.5 Comparison of centralized and distributed approaches at different vector sizes with BGV scheme.	127
Figure 5.6 Comparison of centralized and distributed approaches at different vector sizes with CKKS scheme.	128
Figure 6.1 A Rotating LiDAR System for 360-Degree Environmental Detection	133
Figure 6.2 Critical Safety Zone: Importance of 10-15m Ahead of Autonomous Vehicles in Spoofing Attacks.	138
Figure 6.3 Graphical Representation of LiDAR Spoofing Technique. . .	139
Figure 6.4 LiDAR Shadow Verification: 2D Image-Based Approach for Detecting Ghost Attacks.	140
Figure 6.5 Visualization of 2D Image Generation for Scene Objects. . .	141
Figure 6.6 LiDAR point separation to mitigate shadow adulteration: Removing points above object height (shown in red).	142
Figure 6.7 Comparison of 2D image generation results with genuine (a) and spoofed (b) objects.	142
Figure 6.8 Effective Reduction of Point Cloud Data: Examining Single Objects with 85% Less Data Volume.	143
Figure 6.9 Visualizing Cyclist Class Object Insertion in KITTI Benchmark Point Cloud Dataset.	150
Figure 6.10 Visualization of Object Selection for LiDAR-Based Spoofing Attack using 200 Points.	150
Figure 6.11 Presentation of a ROC curve (green) with an AUC score of 0.9. [256]	152

Figure 6.12 Training progress of the CNN-Linear model evaluated by the metrics.	157
Figure 6.13 ShadowCatcher - Scoring Problem	160
Figure 7.1 Testbed components	163
Figure 7.2 Testbed components	164
Figure 7.3 GPU vs CPU Performance.	165
Figure 7.4 Visual Representation of a Hybrid Virtual-Physical Network with SDN Control and Direct Hardware Resource Mapping.	166
Figure 7.5 Client-Controller-Resource Workflow.	167
Figure 7.6 Distribution of Images Between Jetson and Raspberry Pi. . .	171
Figure 7.7 Distribution of Images Between Jetsons and Raspberry Pis. .	172
Figure 8.1 One-Time-Password Authentication Flow	178

List of Tables

Table 2.1	Machine Learning Methods and Their Potential Applications in Resource Allocation [124].	40
Table 2.2	[Continued] Machine Learning Methods and Their Potential Applications in Resource Allocation.	41
Table 2.3	[Continued] Machine Learning Methods and Their Potential Applications in Resource Allocation.	42
Table 2.4	Summary of Research on Machine Learning-Based Resource Management: Detailing Algorithms, Simulation Tools, and Optimization Objectives [124].	43
Table 2.5	[Continued] Summary of Research on Machine Learning-Based Resource Management: Detailing Algorithms, Simulation Tools, and Optimization Objectives	44
Table 3.1	System Configuration	71
Table 3.2	Task Prioritization and Value Range Assignment Overview	73
Table 3.3	Input Features and Corresponding Value Ranges	75
Table 3.4	Overview of Formulas for Determining the Number of Hidden Neurons in Neural Networks	77
Table 3.5	Percentage of Priority Fulfillment for Classification and Regression Tests	79
Table 3.6	Resource Settings for Dynamic Testing Configuration	87
Table 3.7	Selected Models for Dynamic Testing and Associated Outcomes	88
Table 4.1	Port Knocking Authentication Process	102
Table 4.2	System Configuration	107
Table 5.1	System Configuration	116

Table 5.2	Processing time in (ms) of messages with vector size of 2^{16} using the different schemes on a single resource	120
Table 5.3	Comparison of processing times in (ms) between centralized and distributed approach with BFV scheme	123
Table 5.4	Comparison of processing times in (ms) between centralized and distributed approach with BGV scheme	124
Table 5.5	Comparison of processing times in (ms) between centralized and distributed approach with CKKS scheme	125
Table 6.1	Relationship between Object Distance and Point Density within Bounding Boxes as Detected by LiDAR.	137
Table 6.2	Overview of the CNN-Sigmoid model architecture.	144
Table 6.3	Hyperparameters for the CNN-Sigmoid Model.	146
Table 6.4	Overview of the CNN-Linear model architecture.	146
Table 6.5	Hyperparameters for the CNN-Linear model.	147
Table 6.6	System Configuration	148
Table 6.7	Performance Evaluation of the 3D Shadow Catcher Approach: Training Results [201].	153
Table 6.8	Training results of CNN-Sigmoid model evaluated on dataset 1.	154
Table 6.9	Training results of CNN-Sigmoid model evaluated on dataset 2.	155
Table 6.10	Performance analysis of CNN-Sigmoid model for classification time duration.	156
Table 6.11	Training results of CNN-Linear model evaluated on dataset 1.	157
Table 6.12	Training results of CNN-Linear model evaluated on dataset 2.	158
Table 6.13	Performance analysis of CNN-Linear model for classification time duration.	159
Table 7.1	Average Prediction Time for LiDAR Spoofing Attack Detection on Jetson Device as a Benchmark.	168
Table 7.2	Average Prediction Time for LiDAR Spoofing Attack Detection on remote Jetson Device.	169
Table 7.3	Distribution of Images and Processing Time across 2 Edge Devices.	170

Table 7.4 Distribution of Images and Processing Time across 4 Edge
Devices. 172

LIST OF PUBLICATIONS

Journal Articles

1. A. Almaini, A. Al-Dubai, I. Romdhani and M. Schramm, Lightweight edge authentication for software defined networks. *Computing* 103, 291–311 (2021). <https://doi.org/10.1007/s00607-020-00835-4>
2. A.Almaini, J.Folz, A.Al-Dubai, M.Schramm, H.Heigl, T.Koßmann, I.Romdhani, and B.Canberk, Adaptive Resource Management for Edge Computing in IoT Networks via Neural Networks, *Future Generation Computer Systems* (submitted), 2023.
3. A. Almaini, J. Folz, R. Boeder, A. Al-Dubai, T. Koßmann, M. Schramm, H. Heigl, I. Romdhani, and A. Kerrouche, Shadow-Analyzer: An Efficient Neural Networks-based Ghost Objects Detection for Autonomous Vehicles, *IEEE Intelligent Transportation Systems Transactions* (in revision), 2023.

Conference Papers

1. Michael Heigl, Laurin Doerr, Amar Almaini, Dalibor Fiala, Martin Schramm, Incident Reaction Based on Intrusion Detections' Alert Analysis, 2018 International Conference on Applied Electronics (AE), pp. 45-50, Pilsen, Sept. 2018, doi:10.23919/AE.2018.8501419

2. A. Almaini, A. Al-Dubai, I. Romdhani and M. Schramm, Delegation of Authentication to the Data Plane in Software-Defined Networks, 2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS), Shenyang, China, 2019, pp. 58-65, doi: 10.1109/IUCC/DSCI/SmartCNS.2019.00038.
3. A. Almaini, J. Folz, D. Woelfl, A. Al-Dubai, M. Schramm and M. Heigl, A New Scalable Distributed Homomorphic Encryption Scheme for High Computational Complexity Models, In IEEE International Wireless Communications and Mobile Computing (IWCMC), June 2023, pp. 890-897.

Talks

1. Amar Almaini, Authentication and port scan mitigation in the Software Defined Network Switches, Tag der Forschung 2019, Technische Hochschule Deggendorf, Deggendorf, Germany, April 2019
2. Amar Almaini, Intelligent dynamic resource allocation across edge cloud resources in secure manner, In: Konferenz des berufsbegleitenden Masterstudiengangs Cybersecurity: Ausgewählte Themen der Security Forschung, Technische Hochschule Deggendorf (THD)/Technologie Campus VilshofenOnline, 19.01.2023 (2023)

1 Introduction

The ever-evolving landscape of technology continues to present remarkable development and challenges in equal measures. At the heart of the current technological revolution lies the Internet of Things (IoT), an intricate network of interconnected devices that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. As the digital transformation accelerates, a surge in IoT devices is expected, with projections indicating that over 29 billion IoT devices will be connected to corporate networks by 2030 [1]. The automotive industry is also a key contributor to this proliferation, with autonomous driving and e-mobility trends. It is predicted that autonomous chips, which are essential for autonomous vehicle functions, will generate a staggering revenue of \$29 billion by 2030 [2]. The term IoT devices encapsulates a broad spectrum of devices connected to the internet, either wirelessly or through wired connections. In the industrial context, this predominantly refers to autonomous sensors, machines, and evaluation systems. These devices, although essential, have limited power and depend on auxiliary services for data storage and processing. A single modern vehicle, for instance, generates about five gigabytes of data per minute, contributing to the billions of terabytes of data generated annually. Cloud Computing (CC) has traditionally been the go-to solution for these data storage and processing needs. However, as data volumes balloon and requirements become increasingly demanding, such as real-time processing for autonomous driving, CC providers are reaching their capacity limits [3]. This challenge is further compounded when IoT sensors are deployed in environments with poor or no internet connectivity. When these devices are located far from the local vicinity of cloud servers, long data transfer distances lead to untenable response times [4]. Edge Computing (EC) emerges as a promising solution to these mounting concerns. By processing data closer to the user, EC minimizes latency and meets real-time requirements.

Additionally, it mitigates the risks associated with outsourcing sensitive data to third parties, thereby enhancing the control over information flow. Nevertheless, EC is not without its challenges. Unlike its cloud counterpart, EC relies on locally available computational capacity from nearby devices like cars, sensors, and mobile phones. This locally available hardware, compared to cloud hardware, is generally less powerful, more scattered, and supports a diversity of operating systems. The hardware is not necessarily specialized for a particular task and is connected via various modes, including wireless and direct connections of varying quality. Moreover, these hardware devices can fail, exit the connectivity radius, or connect to the network at unpredictable times, which results in a dynamic network. Resource allocation involves assigning each forthcoming task to the most suitable edge server, whether physical or virtual, capable of hosting it. Given the critical role of Quality of Service (QoS) in distinguishing EC from CC, accurately determining the required amount and quality of resources for a pending request is vital for EC's success. Resource allocation in EC is challenging due to the myriad variables and conditions that must be considered. The unique characteristics of EC suggest that traditional resource allocation approaches fall short. Given these challenges, exploring alternative methods for effective resource management in edge networks is essential. Machine Learning (ML), particularly deep learning methods, have shown promising results in various domains, including autonomous vehicles and data and image processing. Thus, this thesis aims to investigate the integration of ML approaches into edge networks for efficient resource allocation and management. Through this research, we seek to understand the extent to which ML can address the Dynamic Resource Allocation (DRA) problem in EC and lay the groundwork for the next generation of IoT devices and networks. The primary objective of this thesis is to contribute significantly to the field of resource allocation, which is crucial for empowering the IoT. We focus on key resource allocation challenges within EC and IoT, aiming to facilitate the seamless deployment of applications and services. To achieve this, we consider a hierarchical environment where entities with computing power, from user-level devices to the cloud, including mobiles, vehicles, sophisticated network devices, and dedicated servers, can process requests. In the subsequent chapters, we address these challenges and present various objectives and case studies, collectively representing our solution and its real-world applicability.

The first main objective is to model and implement a dynamic resource management approach in EC, analyzing existing strategies, identifying their limitations, and understanding the need for dynamic, adaptable models considering edge node heterogeneity. Chapter 3 proposes and implements a novel model based on Neural Network (NN) for dynamic resource allocation, taking into account the diverse and fluctuating capabilities of edge nodes. The second objective, which also serves as a case study, is to enhance Intelligent Transportation Management Systems through improved EC strategies. This includes exploring EC's role in enhancing Vehicle-to-Everything (V2X) communication, developing techniques to counter security threats like LiDAR spoofing attacks using edge nodes, and validating the proposed strategies through simulation. Chapter 6 introduces a novel technique to counter LiDAR spoofing attacks, employing NN and object shadow verification, requiring minimal data and focusing on 2D transformed LiDAR images for predictions. The third objective addresses security and privacy in EC systems, involving a comprehensive literature review to understand existing challenges and designing strong security measures for EC's multi-layer architecture to counter potential threats and attacks. Additionally, the thesis aims to create mechanisms ensuring user data privacy and secure interactions among edge nodes, especially in sensitive applications. Chapter 4 introduces an innovative technique for client authentication and access to edge resources through the integration of Software Defined Networks (SDN) and P4. Chapter 5 presents a Distributed Homomorphic Encryption (DHE) methodology, designed to enhance data privacy and security while ensuring efficient applicability in EC. This technique demonstrates how computing-intensive Homomorphic Encryption requests can be effectively distributed as smaller subtasks across various edge resources to significantly improve processing times. To further validate the suitability of our approach for edge applications, Chapter 7 demonstrates, through a hardware testbed that embodies the heterogeneity of the edge environment, a scenario in which our resource allocation approach is implemented on an SDN controller. In this scenario, clients send requests to detect spoofing attacks in LiDAR images, utilizing edge computing nodes running the shadow analyzer application. Figure 1.1 conceptually illustrates how the various contributions of this thesis are interconnected and integrated.

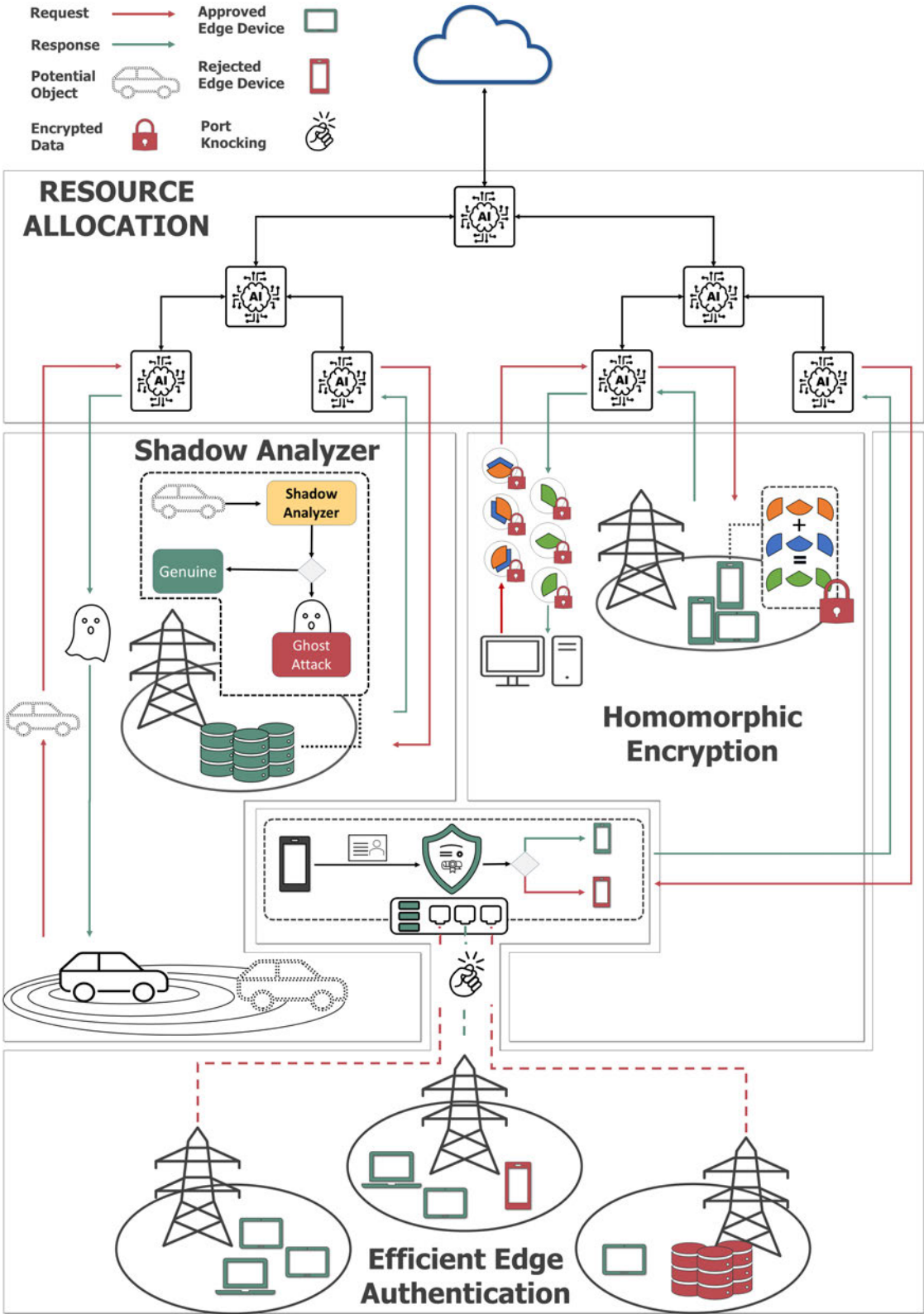


Figure 1.1: Conceptual Framework Illustrating the Interconnections Among Various Topics

1.1 Cloud Computing

The Cloud paradigm refers to a distributed system that remotely manages resources and allows users to access those resources via the Internet. This innovative approach has been transforming the IT industry by introducing greater flexibility in the consumption of IT services, enabling organizations to only pay for the resources they actually use [5]. Cloud infrastructure plays a vital role in reducing business investment costs, including expenses related to money, energy, time, and personnel. It grants businesses the ability to store, backup, and establish a private cloud network at a more affordable price. There is no longer a need to purchase physical components for data management and storage; instead, acquiring a suitable package is sufficient to efficiently handle large amounts of data. CC services are provided by data centers situated in various locations around the world.

1.1.1 Resource Management in Cloud Computing

Cloud Computing is a virtualized paradigm that enables flexible and dynamic reservation and utilization of resources. It allows for scalability by offering additional computing resources to support data or compute-intensive applications. Additionally, these resources are provided as services, enabling their sharing among multiple users, which is a key profit factor for cloud providers [6]. The demand for cloud resources is expected to increase both in personal and professional settings. However, the high demand for cloud services can lead to various challenges related to resource allocation, necessitating efficient handling of resources to cater to numerous users [7] [8]. Resource allocation in CC involves assigning processing tasks to a pool of resources within the cloud infrastructure, which comprises multiple computers.

1.2 Edge Computing

Over the past few years, there has been a significant increase in the number of connected devices, leading to congestion problems. This has prompted the consideration of processing more data at the network's Edge. According to Gartner [9], by 2025, 75% of enterprise-generated data will be handled at the Edge rather than in the centralized Cloud. Additionally, Edge Computing (EC) offers improved service in terms of latency and throughput, as highlighted by IBM [10], reducing data circulation time from 20 ms to 10 ms. The benefits of EC present lucrative market opportunities for various stakeholders, including Cloud providers, internet service providers, and numerous hardware and software companies. Research conducted by Grand View Research [11] predicts that the EC market will expand from \$3.4 billion in 2020 to \$43.4 billion in 2027, with an annual growth rate of 37.4%. Furthermore, EC has gained significant attention in academia over the past decade, leading to a surge in published papers covering topics such as EC architectures, deployment challenges, orchestration platforms, use-cases, and related technologies. EC addresses the growing demand for computing and memory resources with low-latency communication and provides solutions to privacy concerns associated with cloud-based data processing. Essentially, EC enables effective management, scalability, and security of resources at the network edge, allowing them to serve as hosts for workloads received from end devices [12]. EC is closely associated with several innovative technologies, notably the IoT. Alongside the deployment of fifth-generation networks (5G), EC plays a crucial role in enabling the proliferation of connected objects [13]. Furthermore, the rise of Artificial Intelligence (AI) and ML technologies in novel applications has created a growing need for computing resources. EC not only fulfills this requirement but also facilitates the adaptation of AI models to the network edge environment, promoting the concept of "Edge Intelligence" [14].

1.2.1 Resource Management in Edge Computing

Resource management involves the provision of appropriate and convenient edge resources (such as CPU, memory, and I/O) to meet the needs of any edge application requesting them. It also involves optimizing the utilization of the available pool of resources. In CC, an effective resource management strategy provides the cloud provider with a flexible and efficient means of managing their IT resources, which is crucial for a successful cloud business. Resource management plays a vital role in EC as it enables the consolidation of multiple dynamic, diverse, and distributed edge nodes [15].

1.2.2 Computation Offloading

Computation offloading, a field in computer science, focuses on determining whether a process should be executed locally or sent to an external commodity server for processing. This concept gained popularity with the emergence of mobile CC [16]. When mobile devices are unable to handle resource-intensive applications like Google Assistant or Apple Siri, voice recognition tasks are offloaded to the Cloud. However, with the growing interest in EC, the decision of offloading has become even more crucial and has taken on new forms. In addition to vertical or unidirectional offloading, there is now horizontal offloading, which involves transferring tasks between IoT devices, edge servers, and any destination server within the continuum of mist-fog-cloud. The authors of work [17] presented a literature review that addresses key questions in computation offloading: When and where should offloading occur? And what factors should be considered in making this decision? Offloading involves selecting suitable resources, filtering them, and determining the most appropriate ones for a given task [18].

EC recognizes four types of offloading directions, as outlined below:

1. End-device-to-End-device: When IoT and end-user devices become more powerful, devices in close proximity can collaborate. Tasks are executed locally if feasible, or they are forwarded to nearby IoT devices with lighter workloads [19].

2. End-device-to-Cloud: Incorporating the Cloud into the offloading process significantly enhances system capacity, especially for tasks that are not time-sensitive [20].
3. End-device-to-Edge-to-Cloud (hierarchical offloading): In this technique, an end-device sends requests to suitable edge servers, and the edge server determines which parts of the task should be executed locally and which should be offloaded to the Cloud [21].
4. Vertical and horizontal offloading: End-devices can simultaneously transfer tasks vertically (to edge/fog/cloud) and horizontally (to neighboring nodes). This type of offloading is well demonstrated in Vehicular Edge Computing (VEC) [22].

1.2.3 Network Management

Network management involves advancements in network infrastructure and architectures to adjust network parameters to the evolving computing paradigm. Its primary focus is to monitor, analyze, and dynamically adjust the network status as required. For successful EC, it is crucial to enable resilient and cost-effective network management methods, including access control, traffic engineering, and the adoption of the latest network technologies. This can be achieved and enhanced using, for example, Software Defined Networks (SDN) [23], an abstraction technique that separates network control from data transmission by logically centralizing network command functions in an SDN controller. The controller provides instructions to the network's forwarding devices regarding data packet handling, including where and when to transfer data. To achieve this, the Data Plane devices must support programmable switches that utilize the innovative OpenFlow protocol [24]. Overall, SDN enhances network flexibility and programmability. Furthermore, by designing optimal paths and employing efficient packet forwarding procedures, SDN can be seamlessly integrated with the EC environment [25]. In EC, various scaling functions, such as computation offloading and load balancing, require the collaboration of multiple network data plane components. In this context, SDN architecture proves beneficial as it allows the

SDN controller to optimally distribute bandwidth resources among different data flows [26]. Additionally, due to SDN's ability to have a comprehensive overview of network topology and user movement, SDN controllers assist in determining the ideal edge server destination for hosting migrated services [27]. Mobility scenarios can incorporate other specifications like throughput or user preferences to guide the SDN controller in executing optimal handovers for edge services [28]. Apart from mobility, a networking load balancing mechanism can be achieved by configuring SDN switches across multiple edge servers [29], ensuring network resilience and protecting against traffic spikes [30]. Moreover, SDN can enhance the performance of various edge applications, such as Content Delivery Networks (CDNs), as the SDN controller can leverage network aggregated information to establish the shortest paths between users and content provider servers [31]. In addition to network-related decisions, the SDN controller can fulfill various other essential decision-making tasks, including ML inference [32]. In this context, the controller can determine, based on accuracy, whether to transmit the ML task to the Edge or keep it at the IoT device level. Furthermore, one of the improved functions facilitated by SDN architecture is traffic classification. Traffic classification involves categorizing data flows, whether encrypted or not, into different categories based on packet byte information. This differentiation allows for distinguishing video surveillance traffic from e-health and email traffic. By leveraging traffic classification technologies, SDN and its integration with MEC (Multi-access Edge Computing) can enhance network congestion management [33]. When MEC servers interact with SDN controllers, they can store delayed tolerant traffic (e.g., email traffic) and redirect it after a reasonable delay. Alternatively, for latency-critical tasks, the SDN controller can select reliable links less prone to failure to handle critical traffic, such as the e-health forwarding schema [34].

1.2.4 Security and Privacy

The significant differences in the network edge environment pose a major challenge when it comes to EC security issues. In conventional CC, data is stored in secure large data centers with stringent physical protection measures such as

guards, fences, and security protocols. Furthermore, cloud providers invest heavily in cybersecurity. However, the landscape in EC is quite distinct. The physical edge devices are spread out and heterogeneous, making them more susceptible to physical attacks. Moreover, the extensive data offloading and circulation at the network edge make edge servers more vulnerable to cyber threats [35]. Despite these vulnerabilities, EC offers better privacy protection compared to CC since the data remains closer to the end users. Traditional CC requires all data to be uploaded to the cloud for processing, which is a centralized method. This process entails risks such as data loss and data leakage, compromising security and privacy. In contrast, EC processes data locally within its own scope, eliminating the need to upload data to the cloud and thereby avoiding the risks associated with network transmission. Thus, the security of data can be more effectively guaranteed. When data is compromised, the impact is localized, not affecting the entirety of the data. For instance, when using various applications on smartphones that require user data, including private information, there is a high risk of privacy leakage or attack when this data is uploaded to the cloud center [36].

1.2.5 Edge Computing Applications

EC is applied in various fields, including:

- **Smart Buildings**

Smart buildings are a key component of smart cities, aimed to enhance the efficiency, intelligence, and adaptability of buildings. These buildings intelligently monitor and control devices such as kitchen appliances, light fixtures, and TVs based on user preferences. However, managing the building environment can be computationally intensive, requiring modern intelligent features to facilitate rapid user interaction. This is where EC becomes invaluable, offering fast response times and cost-effective solutions for smart buildings. Advanced functionalities of smart buildings include estimating room occupancy, conducting outdoor video surveillance, and tracking and identifying individuals. An example of a smart building is the Burj Khalifa in Dubai, one of the most iconic buildings globally and a pioneer in intelligent construction. According to Honeywell, the nation's smart building

score, which assesses a facility's sustainability, safety, and productivity, is 65 out of 100. With a height of 2,716 feet and 160 floors, the Burj Khalifa not only holds the title of the world's tallest building but, with Honeywell's assistance, has also become one of the most advanced and environmentally friendly smart building examples. Honeywell collaborated with the building's managers in its main venue to implement several smart building features, enhancing residents' access to better air quality, lighting, and temperature control. Honeywell's IoT platform receives real-time data from the intelligent building automation system, using sophisticated algorithms to identify anomalies and maintenance issues. Facility managers can use this information to improve asset reliability and building maintenance [37].

- **Smart Industry**

Computing has successfully aligned itself with the latest requirements of Industry 4.0, a integration commonly referred to as Industrial EC [38]. The adoption of predictive maintenance has become crucial for emerging industries as it helps reduce their capital and operational expenses (CAPEX and OPEX). This approach involves equipping machines with various Industrial Internet of Things (IIoT) sensors, such as temperature, vibration, and pressure sensors. These sensors collect data, which is then transmitted to edge nodes for processing. The processed data is used to predict machine failures and errors [39]. Furthermore, the fourth industrial generation aims to incorporate Artificial Intelligence into its manufacturing processes. However, due to the inability of industrial companies to transfer their private data (e.g., production scene videos) to the cloud, they tend to rely on EC [40]. The demand for IIoT-based technologies is expected to soar in the coming years, especially in the Southeast Asian region. An example of smart industry is Zuellig Pharma, representing the healthcare sector. Accenture and Singtel teamed up to create a smart 5G-enabled warehouse for Zuellig Pharma, one of Asia's largest healthcare service providers. Technologies deployed include 5G, edge computing for real-time data analytics, augmented reality (AR) headsets and vision, and drones. The AR goggles were used for more efficient product picking, while 5G-enabled drone technologies facilitated

a full inventory count of the warehouse in 20 minutes. By deploying these smart technologies, Zuellig Pharma adopted a more data-driven, automated approach. The AR Vision Picking solution resulted in a 30% improvement in pick productivity and 100% pick accuracy. Moreover, the drone inventory counting solution achieved a 95% counting accuracy [41].

- **Smart Roads**

EC has played a significant role in enhancing the safety and intelligence of modern transportation systems. According to a survey referenced in [42], intelligent Traffic Management Systems (TMS) integrated with smart roads have promoted global awareness among various road components. EC enables vehicles to communicate with each other, ensuring road traffic safety and balance [43]. Vehicle-to-Thing communication allows for more efficient management of special road scenarios, such as accidents or emergencies involving vehicles like ambulances and police cars, by directing vehicles to proactively free up road lanes. Edge Servers (ES) support functions such as vehicle collision detection by utilizing aggregated data from the road environment [44]. ES can provide real-time instructions and adjustments to vehicle speeds, trajectories, and lighting systems to prevent collisions. Moreover, EC enhances the capabilities of road cameras for vehicle detection and tracking [45]. Additionally, security concerns related to autonomous driving, such as Light Detection and Ranging (LiDAR) spoofing attacks, can be managed by edge nodes [46]. The concept of smart roads encompasses a variety of technologies to improve road functionality. For example, 'Electrified Roads' automatically charge electric vehicles through specially designated charging lanes. Various pilot projects, notably in Germany and Sweden, are testing this technology with different approaches. In Sweden, around 2 kilometers of road near Stockholm have been transformed into an electrified road as part of the eRoadArlanda project, recharging the batteries of cars and trucks by connecting them to a rail system embedded in the road. Meanwhile, Germany is exploring wireless induction systems that install coils under the asphalt, transmitting power to the vehicle without direct contact, utilizing the same electromagnetic induction technology used for electric toothbrushes, mobile phones, and other devices [47].

- **E-Health**

By the onset of the current century, the health sector had already witnessed the integration of electronic and information systems, leading to the emergence of what is now widely known as E-Health. This evolving healthcare model leverages electronic devices for diagnosing and treating patients, alongside extensive use of computers for collecting and analyzing health records. EC plays a pivotal role in advancing e-Health by facilitating medical records storage, addressing privacy concerns, and mitigating delays in data retrieval from the cloud [48]. Moreover, utilizing fog/edge computing for continuous remote health monitoring of patients [49] can significantly decrease network bandwidth usage. EC is essential for developing next-generation E-Health applications, such as Telesurgery [50], where doctors remotely control surgical robots from their homes. Such procedures demand ultra-low latency, achievable only through EC. E-health applications have seen a surge in recent years, introducing many innovative solutions that ease the workload of hospital staff and enhance patient care. For instance, Synappz is an E-health application [51] that allows patients to monitor their health at home. Patients can measure their own vitals and input the data, which healthcare providers can then access through a portal and provide tailored information relevant to the patient's care pathway. Another example is Medicus.ai [52], an application designed to help individuals better understand their health by explaining medical reports and health data in a personalized, comprehensible, and visual manner.

1.3 Motivation

As previously mentioned, EC plays a crucial role in realizing the IoT concept, particularly when combined with CC as a continuum. This combination opens up a wide range of services and applications that can greatly enhance our lives, providing increased convenience and flexibility. However, the successful implementation of EC faces various challenges that can affect its performance. One of the main challenges is the efficient, secure, and dynamic allocation of resources, considering

the unique requirements of different application areas [53]. Therefore, numerous efforts have been dedicated to designing and proposing resource allocation approaches that optimize computational and bandwidth expenses, while ensuring timely response, cost-effectiveness, and task prioritization. Nevertheless, these approaches lack a solid foundation for dynamic, secure behavior that can be applied across different domains and handle the predicted exponential growth of data. Hence, the objective of this thesis is to address some of the key shortcomings in resource allocation for EC, specifically focusing on the requirements of the IoT paradigm. Ultimately, this research aims to facilitate widespread deployment of EC applications and services in diverse fields.

1.4 Problem Statement and Research Question

Resource management plays a crucial role in EC, and the subpar performance of resource management significantly impacts the services that can be provided and supported, thereby hindering the future deployment of EC. Despite the substantial progress in research on resource management in EC, there are still significant challenges that can impede its efficiency.

Firstly, the EC environment is complex and dynamic, characterized by heterogeneous devices with varying computation, storage, and communication capabilities. This heterogeneity poses a critical challenge in achieving performance improvement objectives, as limited resources in edge nodes lead to workload imbalances and negatively affect system performance in terms of delays. Additionally, high request rates can cause task queues to lengthen in powerful edge nodes, as limited resource edge nodes may struggle to process the entire input data. Moreover, to minimize reliance on the cloud and maximize the utilization of edge resources, there is a need to explore the concept of task division, which can potentially reduce task execution delays through parallel subtask executions in limited resource edge devices [54].

Secondly, EC resources are subject to dynamic changes due to mobility. The presence of mobile edge nodes like drones [55] and vehicles [56], [57] leads to fluctuating EC resources over time. The joining and leaving of edge devices

further contribute to these changes. Additionally, the resource requirements for executing tasks dynamically change based on various task types, specific applications, time periods, and environmental conditions. Consequently, resource allocation strategies must be adaptive and flexible to cope with such situations. In general, resource allocation in EC systems involves three major computing problems: resource sharing, task scheduling, and task offloading [58]. Resource sharing refers to methods of distributing available resources among edge devices to meet the computational demands of tasks [59], [60]. This is crucial in EC environments where the heterogeneity of edge resources necessitates multiple computing devices to complete a single task. Cooperation among nodes is required to execute computational task demands, and mechanisms for edge-to-edge collaboration must be established to facilitate resource sharing. However, enabling cooperative edge-to-edge interactions is challenging due to the differences in hardware, software, and functionalities of practical devices [61]. Task scheduling is typically performed without sufficient information support, as there are no predictable patterns for the arrival of task profiles in terms of number, size, and arrival rate. Consequently, scheduling algorithms must make prompt decisions without prior experience or prepared information, while also optimizing resource utilization based on changing demands. Task scheduling significantly impacts the computing system's performance, particularly in terms of task execution delays. Balancing workload distribution among computing devices is a challenging problem to ensure stable operation and prevent underutilization of available resources in the limited resource edge. Efficient task offloading algorithms need to consider various factors such as resource states of edge devices and dynamic task requirements.

Thirdly, existing research on resource allocation has not adequately addressed security and privacy concerns. The multi-layer architecture of EC renders the edge system vulnerable to hostile attacks, which can jeopardize reliability and robustness. Trustworthiness of offloaded edge nodes, user authorization for edge services, and privacy protection of data generated by edge services are critical issues. Presently, user data, edge node interaction data, and EC data are often uncritically trusted and accessible [62]. However, in practical applications such as

smart homes and smart health, this data often contains private and commercial secrets that could lead to significant losses if leaked [63], [64]. Consequently, designing authentication and trust mechanisms, along with privacy preservation policies for users covered by specific edge nodes is necessary.

Based on the above observations, the following research questions are to be addressed:

- **DRA in Heterogeneous EC Environment:**

How can we devise efficient algorithms or models for dynamic resource management in EC environments that take into consideration the heterogeneity and variability of edge nodes, thus reducing workload imbalance and optimizing task execution delays? What role can task division and parallel execution play in optimizing the performance of EC environments?

- **EC in Intelligent TMS:**

What strategies and protocols can be established to enhance Vehicle-to-Thing communication using EC, ensuring traffic safety and efficiency while handling special road scenarios? How can we address the security issues related to autonomous driving, such as LiDAR spoofing attacks, using edge nodes?

- **Security and Privacy in EC:**

How can we design and implement robust security and privacy-preserving measures in the multi-layer architecture of EC systems to prevent potential threats and attacks? What mechanisms can be developed to ensure user data and edge node interaction data are trusted and secure, particularly in sensitive applications like smart homes and smart health?

1.5 Aims and Objectives

The principal aim of this thesis is to investigate, model, and propose innovative solutions that enhance the performance, security, and privacy of EC systems in various contexts such as intelligent Traffic Management Systems and other IoT-enabled environments. Our intended research work comprises a series of

objectives that will pave the way toward achieving our overall aim. Each objective, in turn, consists of several sub-goals aimed at achieving the specific objective.

1. Objective 1: Propose and implement an algorithm or model for dynamic resource allocation, taking into account the heterogeneity of edge nodes:

- Analyze existing resource management strategies in EC, identify their limitations, and provide a comprehensive understanding of the need for dynamic and adaptable resource management models.
- Investigate the role of task division and parallel execution in enhancing the performance of EC environments.
- Evaluate the effectiveness and efficiency of the proposed dynamic resource management model.

2. Objective 2: Enhance the security of Intelligent Traffic Management Systems by utilizing edge resources:

- Investigate the role and potential of EC in enhancing Vehicle-to-Thing communication.
- Devise techniques to mitigate security threats, with a specific focus on LiDAR spoofing attacks using edge nodes.
- Validate the proposed strategies and protocols through simulation.

3. Objective 3: Enhance Security and Privacy in EC Systems:

- Conduct an extensive literature review to understand the current security and privacy challenges in EC systems.
- Design and implement robust security measures for the multi-layer architecture of EC systems, aiming to counter potential threats and attacks.
- Create mechanisms that ensure user data privacy and secure interaction of edge nodes, especially in sensitive applications.

Each of these objectives contributes to the overall aim of the Ph.D. research and addresses the research questions posed.

1.6 Contributions

This doctoral thesis focuses on the exploration, modeling, and proposal of novel approaches to improve the performance, security, and privacy of EC systems. It specifically targets diverse scenarios such as intelligent TMS and other environments enabled by the IoT. The goal is to go beyond existing standardized solutions and advance the field of EC by addressing challenges related to efficient, dynamic, and reliable resource management. By doing so, the thesis aims to facilitate the widespread deployment of scalable, reliable, and secure applications within the IoT domain. In addition to an extensive literature review, the main contributions of this thesis can be summarized as follows:

- **A framework for adaptive resource management in EC using Neural Networks**

The first major contribution of this thesis involves the design and implementation of an intelligent system based on NNs. This system is designed to dynamically allocate resources in EC. Its primary objective is to tackle the challenges associated with deploying EC on a large scale. These challenges include the complex and dynamic nature of the edge environment, which is characterized by a variety of devices with differing computation, storage, and communication capabilities. To address these challenges, the proposed framework utilizes dynamic resource allocation to reduce service delays and balance resource utilization at the edge. By doing so, it mitigates workload imbalances caused by limited resources at the edge, which can negatively impact system performance in terms of delays and overloading powerful edge nodes. Moreover, this framework aims to minimize reliance on cloud resources and maximize the utilization of edge resources.

- **An efficient lightweight approach for authenticating edge devices**

The second major contribution of this thesis is the development of an efficient and lightweight authentication method for edge devices. This method addresses security and privacy concerns in EC and aims to enhance the

trustworthiness of edge devices. It facilitates user authorization for edge services in a lightweight manner, conserving computing power and reducing authentication time for a large and dynamic number of IoT devices. Importantly, as most IoT devices have limited battery power, the lightweight approach helps conserve battery life. Additionally, since authentication occurs in close proximity to the IoT device, fewer devices are involved in forwarding authentication packets, saving battery power for all devices not required to be involved. However, the issue of power consumption is beyond the scope of this thesis. The proposed method achieves these goals by delegating some of the intelligence from the SDN-controller to the data plane. This delegation enhances the overall network performance, reduces signaling load, and offloads the controller, thereby meeting real-time requirements. The method takes advantage of programming protocol-independent packet processors (P4) language to ensure that only legitimate nodes can access the network. Notably, this solution delegates the access control task to the data plane in an SDN environment, relieving the controller from potential failures caused by overload. An important feature of this technique is the fast authentication of edge devices, which is performed by a switch in the data plane located near the edge device. Furthermore, this technique exhibits scalability as the controller can deploy additional authentication nodes at any time. This capability extends the coverage area, allowing for the authentication of new edge devices within their immediate proximity.

- **A technique for detecting objects using efficient and lightweight NNs**
The third major contribution of this thesis involves the development and evaluation of a novel real-time detection technique named "Shadow Analyzer." This technique is designed to enhance the safety and efficiency of autonomous driving by effectively identifying LiDAR spoofing attacks. Through its innovative approach, Shadow Analyzer has shown significant improvements in managing special road scenarios and enhancing V2T communication. It accomplishes this by minimizing the amount of data needed for object detection, utilizing a condensed 2D dataset derived from the origi-

nal 3D point cloud. This data reduction allows for the detection process to be executed onboard the vehicle or on edge nodes, facilitating rapid response to potential threats. Furthermore, the method addresses critical security concerns related to autonomous driving, particularly LiDAR spoofing attacks, by allocating the detection and management of such threats to edge nodes. This strategy not only reduces the computational load on the vehicle but also enables efficient edge data processing and optimal utilization of available computing resources. In line with the second objective of this thesis, this technique mitigates security threats, specifically LiDAR spoofing attacks. Additionally, by leveraging a reduced 2D dataset, it facilitates V2T communication, a critical advantage given that many edge devices have limited computing power and memory capacity.

- **A scalable distributed security scheme through homomorphic encryption**

The fourth major contribution of this thesis is the introduction of a novel distributed approach to Fully Homomorphic Encryption (FHE) for processing sensitive and confidential data while ensuring information security and privacy. FHE enables computations to be performed on encrypted data, preserving confidentiality even during data processing. Only authorized users or those with the decryption key can access and interpret the data. This allows secure outsourcing of data processing to powerful public computing resources on the edge, even if they are untrustworthy. However, FHE-based data processing faces scalability challenges due to its high computational complexity. The main goal of this technique is to address the concerns related to security, privacy, and scalability. The proposed distributed FHE-based data processing approach aims to overcome the scalability issues encountered in FHE-based computations by leveraging distributed computing techniques. This approach divides the computational workload among multiple computing instances, thereby reducing the overall computational burden on a single instance. Moreover, the distributed approach enhances privacy by eliminating the possibility of meta-level inferences about the

computations being performed. By distributing the computations across multiple nodes, it becomes challenging for any single entity to comprehend the nature of the computation or gain access to sensitive information.

1.7 Contributions Complementarity and Justification

This thesis presents several contributions that individually enhance the applicability of EC within specific domains. However, these contributions can also be collectively implemented to achieve multiple objectives simultaneously. For instance, our proposed lightweight authentication technique is designed for integration into a SDN environment to enhance security while simultaneously offloading the controller to manage other tasks. This is particularly advantageous in mobile IoT-SDN environments, where mobile devices create highly dynamic topologies. Our method facilitates lightweight network access for these devices for specific durations. Furthermore, our dynamic resource allocation framework is versatile, optimizing resource utilization to reduce costs, conserve energy, decrease response times, and ensure load balance. The Shadow Analyzer, with its innovative 2D technique, offers high accuracy in a lightweight package, making it an ideal tool for use in autonomous vehicles within intelligent transportation systems. Similarly, our distributed Fully Homomorphic Encryption (FHE) scheme can effectively address multiple scenarios, especially where data must be processed on remote, potentially untrustworthy servers to maintain the security and privacy of sensitive information. By integrating all these contributions, the widespread adoption of EC could be facilitated, providing solutions for efficient adaptive resource management in dynamic environments, addressing security and privacy concerns for sensitive data processed at the edge, offering scalable and lightweight network access control, and mitigating specific attacks on smart vehicles, particularly LiDAR spoofing attacks. To demonstrate how different contributions can be collectively implemented to achieve multiple objectives, Chapter 7 of this thesis presents a hardware testbed where both the resource allocation framework and the Shadow Analyzer are complementarily implemented, showcasing how the Shadow Analyzer benefits from the resource allocation technique.

1.8 Thesis Structure

The remainder of the thesis is organized into seven chapters as follows:

- **Chapter 2: Literature Review**

This chapter provides a comprehensive review of major studies that have strong relevance to the central contributions of this thesis. We critically evaluate the pros and cons of these works, emphasizing the pressing need for the novel approaches introduced in our research. The discussion commences by outlining works directly associated with the primary contribution of this thesis, setting the stage for readers to understand the foundational knowledge. Subsequently, we delve into contemporary, state-of-the-art research that aligns with the broader contributions of this thesis, ensuring a well-rounded discussion and reflection on the topic.

- **Chapter 3: Neural Network-Driven Resource Management in IoT Edge Computing (Contribution 1)**

The chapter outlines the creation of an intelligent system for resource allocation in an SDN environment using an AI agent and mininet for testing. Different model types and reward functions were implemented and assessed to find the best approach. Applications were developed to ready network components for the desired goal. The study details the steps for making the necessary dynamic behavior testing models, with thorough evaluations for optimal settings. Notably, the regression model performs the best due to its ability to assess resources individually and choose the highest rated one, suited for dynamic systems. A classification model was also explored but showed performance limitations due to fixed resource properties.

- **Chapter 4: Lightweight edge authentication for SDN (Contribution 2)**

The chapter presents a new approach that uses P4 for authentication and port scan prevention in switches. The technique's adaptability and real-time capabilities are showcased, making it suitable for practical situations, especially in EC. Additionally, a one-time password-based authentication method is introduced. These methods collectively tackle security and privacy worries in EC, enhancing the reliability of edge devices. They enable efficient user authorization for edge services, saving computing resources and decreasing authentication time for a diverse range of IoT devices.

- **Chapter 5: Scalable Distributed Homomorphic Encryption for Complex Computational Models (Contribution 3)**

This chapter introduces a novel approach for distributed FHE designed to process sensitive data securely. The approach's performance was evaluated using virtual instances on an external server, employing three widely used FHE schemes (CKKS, BGV, and BFV). The experiments involved arithmetic operations on datasets up to 2^{16} in size. By conducting fifty test runs for each scheme and data size, comparing centralized and distributed methods, the distributed approach exhibited time savings of up to 54%. These results highlight the potential of the proposed approach in efficiently handling confidential data in public environments while maintaining security.

- **Chapter 6: Shadow-Analyzer: Efficient Detection of Ghost Objects in Autonomous Driving using Neural Networks (Contribution 4)**

This chapter focuses on improving V2T communication by using EC to enhance traffic safety and efficiency, specifically addressing the threat of LiDAR spoofing attacks. The main goals are to reduce data requirements, maintain accurate detection, and enable real-time responses. The approach highlights efficient edge-based data processing near vehicles, crucial for quick responses in time-sensitive situations. The chapter introduces a lightweight LiDAR spoofing-attack detection method called the Shadow-Analyzer, using 2D bird's-eye view images. This method achieves a remarkable average accuracy of 97.8% in detecting ghost objects within 6ms, proving its potential for real-time detection without compromising efficiency.

- **Chapter 7: Empirical Validation of Usability: Extensive Evaluation of Our Approach in an Edge Computing Hardware Testbed**

The chapter investigates the practicality of the proposed solution using a custom testbed with varied computing devices, representing diverse edge IoT resources. The solution's resource allocation framework is integrated into the SDN controller, focused on managing these resources, especially for the shadow analyzer task discussed in Chapter 6. The conducted tests, involving different computing power resources, yield promising results. These findings indicate that the resource allocation framework effectively optimizes computing resource allocation, enhancing task completion efficiency.

- **Chapter 8: Conclusions and future work**

This chapter provides a concluding overview of the thesis, outlining the primary research gaps that have been explored and discussing the principal methodologies employed to address these gaps. Furthermore, it expounds upon the principal findings made throughout the research and sheds light on the key lessons learned highlighting future research prospects and directions.

2 Literature Review

2.1 Introduction

Resource management is a cornerstone in the realm of CC and EC, directly influencing the efficiency and effectiveness of these technologies. In this chapter, we delve into the multifaceted world of existing resource management strategies within these domains. Our journey begins with a comprehensive exploration of the techniques and methods currently in play, followed by a close scrutiny of their characteristics and inherent limitations. One of the salient aspects that will receive special attention is the dynamic nature of edge resources. To provide a holistic perspective, the state-of-the-art works closely related to the central contributions of this thesis will be evaluated. Through a critical analysis of their strengths and weaknesses, we underscore the pressing need for our pioneering contributions, all of which aim to bolster the efficiency of edge computing.

2.2 Resource Management in Cloud Computing

Resource management stands out as a major issue in CC, as highlighted in [65]. Another crucial problem is devising resource allocation techniques, particularly for determining the optimal placement of modules [66] alongside other devices to enhance throughput and reduce latency. Data centers in CC face challenges in implementing methods for resource migration and allocation [67]. Inadequate resource allocation leads to resource wastage and subpar service delivery within data centers. Effective allocation and recognition of resources such as servers, memory, and CPUs are essential. Various methods are employed to achieve optimal resource allocation in CC. These can be categorized as follows [68]:

- Numerical methods: These mathematical techniques are utilized to solve numerical problems. For instance, in [69], the authors utilized the Exact VM Allocation algorithm, an extended Bin-Packing approach, to address power consumption concerns. To achieve efficient VM consolidation, they employed a migration module to manage the state of unused hosts (such as sleep or shutdown) and determine the placement of VMs.
- Numerical approximate methods: These methods rely on probability or operational search techniques to estimate excellent solutions for task scheduling problems. In [70], Sandhu et al. proposed the KNN algorithm to calculate the distance between data center locations. The distribution of queries within the data center is adaptively performed based on the available resources and the resource consumption anticipated for big data requests.
- Linear programming methods: This traditional analytical approach involves mathematical formulation. Chaisiri et al. introduced the Optimal VMs Placement algorithm (OVMP) in [71] to deploy VMs on hosts. Their architectural system consisted of a user, VMs, cloud providers, and a cloud broker. The OVMP aimed to select an optimal solution by solving stochastic integer programming through two main steps: determining the number of provisioned VMs in the first step and the number of VMs assigned for utilization in the second step.
- Learning methods: Some researchers have utilized learning techniques to address DRA issues. For example, in [72], a learning strategy-based dynamic resource allocation was proposed for a virtual streaming media server cluster in a multi-version VoD system. Similarly, [73] employed a ML algorithm to determine optimal resource allocation. Moreover, [74] presented a solution based on learning methods to optimize resource allocation. Learning methods have emerged as a dominant approach in various research fields and hold promise for enhancing the performance of cloud environments. Consequently, there has been an increased interest in developing new learning techniques for DRA [75], [76].

Numerous studies in the scientific literature have explored cloud technology, specifically focusing on its key characteristics, features, and unresolved matters. These studies have also investigated various approaches employed to optimize resource allocation within this environment. In [77], the authors proposed a classification of cost optimization methods for scheduling scientific workflows in cloud and grid computing. Similarly, Challita et al. discussed different techniques utilized for Virtual Machine (VM) placement, citing energy management, resource utilization optimization, and traffic engineering as the driving factors behind such scheduling practices [78]. Kalra et al. [79] deliberated on task scheduling techniques based on Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithm (GA), as well as two novel approaches, the League Championship Algorithm (LCA) and BAT algorithm. Masdar et al. [80] conducted an analysis of workflow scheduling approaches in CC, additionally highlighting works involving Simulated Annealing (SA) and Cat Swarm Optimization (CSO). Furthermore, they examined Heuristic-based methods and Hybrid Algorithms. Another study investigated job scheduling methods such as first-come first-served, Round Robin, Min-Min, and Max-Min algorithms [81]. Nevertheless, these studies fail to consider the dynamic nature of resource allocation within the cloud. Although there are significant advantages associated with this new technology, there exist significant worries and obstacles. The primary concern revolves around security and privacy matters in cloud-based settings. Moreover, resource allocation, load balancing, data management, data availability, scalability, compatibility, and interoperability pose additional challenges in cloud-based environments, which ultimately impact the effectiveness and dependability of this technology.

2.3 Resource Management in Edge Computing

The previous chapter highlighted the significance of resource management in EC, emphasizing its role in unifying various dynamic, diverse, and dispersed edge nodes [15]. Resource management can be conceptualized as a sequence of interconnected actions. This begins with the formation and distribution of resources, monitoring them, and strategizing for their future deployment. This chapter delves into these steps, aiming for a deeper understanding of effective resource management within EC.

2.3.1 Resource Allocation

The task of resource allocation or planning involves assigning each forthcoming task to the most suitable edge server, either physical or virtual, that can host it. Given the importance of Quality of Service (QoS) in differentiating EC from CC, determining the precise amount and caliber of resources needed for a pending request is crucial to the success of EC [82]. Resource allocation in EC is complex due to the need to take into account numerous variables and circumstances. The following are a few of these considerations: Resource scheduling aims to minimize computational and bandwidth expenses, while ensuring the requisite response time is maintained [83], [84], making it cost-driven. In terms of crowd management, it is important for a workload that is assigned to nearby-edge resources to recognize the load percentage of adjacent edge devices, thus preventing overloading of a particular host [85]. As for dynamic demand, apps and services at the edge frequently exhibit varying needs in terms of quantity and quality of resources over time and location. Therefore, the scheduler must take these fluctuations into account, dynamically revising and adjusting the allocated resources to avoid degradation in QoS or under-utilization of resources [86]. The aspect of priorities plays a role in EC environments, where numerous users are competing for a place in an edge node. Here, the scheduler must strive to treat all user requests equitably. The research in [87] examines various scheduling algorithms based on different priorities, such as the order of arrival, client type, task nature, and so forth. Finally, fault tolerance is important since edge nodes can lose power or connectivity at any moment. Providing backup versions of any scheduled application would enhance the overall reliability of EC services [88].

2.3.2 Resource Placement

The allocation of resources involves designing and optimizing the strategy for distributing physical and virtual edge servers. As edge/fog computing is still in its early stages of real-world implementation, numerous current studies are aiming to determine the most effective placement strategy for edge nodes. These studies consider various criteria such as latency, reliability, and user preferences, while also taking into account different scenarios like metropolitan networks and

vehicular networks. In metropolitan areas, individuals and businesses can utilize EC to meet their computational and memory requirements. In this context, edge nodes are distributed similar to antennas, meaning that as the population density increases, more edge servers are needed to meet the demand [89]. However, a more precise approach involves considering not only the number of users in a region but also their interest in using applications with low latency [90]. Additionally, in Multi-access Edge Computing (MEC) servers are distributed across cells. It is recommended that when multiple users from different cells interact, such as playing virtual reality games or streaming together, they should be served by the same MEC server to avoid relying on a third-party aggregation server in the cloud [91]. Placing edge servers as close as possible to access points is also suggested to minimize latency, although this approach raises concerns about the associated capital and operational costs. Researchers have explored finding the optimal balance between offering QoS and reducing costs in the placement of 5G-MEC servers [92]. Furthermore, when distributing edge resources, it is crucial to consider robustness as a criterion [93]. Robustness refers to the system's ability to continue functioning normally even if multiple edge nodes fail or come under attack. The distribution of resources should be robust enough that if an edge node fails, there is another one in that region capable of replacing it.

2.3.3 Resource Provisioning

Resource provisioning involves aligning the quantity and quality of resources with the desired level of service for users. In the context of EC, provisioning resources entails the tasks of planning, estimating, and pooling the necessary physical and VMs. These resources need to be precisely customized in terms of processor, memory, and network interfaces before being handed over to the scheduler for utilization by upcoming requests [94]. However, unlike in CC, where resource costs tend to remain stable, the edge environment experiences spatial-temporal fluctuations in prices. Therefore, resource provisioning actions in this context must adapt to recent changes and strike a better balance between QoS and costs [95]. A notable example of spatial changes can be observed in vehicular EC, where resource provisioning should be based on traffic conditions [96].

2.3.4 Resource Pooling

The process of establishing a collection of resources that can be allocated or planned based on upcoming demands is referred to as resource pooling. The objective of resource pooling is to assemble diverse edge nodes together and organize them into a unified community, enabling them to interact and utilize one another's computational and networking resources [97].

2.4 Security and Privacy

The preceding chapter emphasized the unique challenges posed by the distributed and diverse nature of physical edge devices, making them particularly vulnerable to physical attacks. Additionally, the high volume of data transfer and circulation at the network edge increases the susceptibility of edge servers to cyber threats. In this chapter, we will explore specific security concerns associated with EC.

2.4.1 Attack Detection

The field of attack detection involves the examination of various adversarial attacks and vulnerabilities that specifically target edge devices and servers, particularly those that are embedded. Since certain lightweight edge devices lack the capability to utilize defensive tools like anti-viruses or firewalls, it is necessary to develop software detectors for unusual behavior and testing techniques specifically tailored for these edge nodes [98]. One of the most widely recognized threats to edge servers is Distributed Denial of Service (DDOS) attacks. These attacks aim to overload the server's capacity by bombarding it with requests from numerous zombie machines. Such attacks are notably more effective when targeted at the Edge as opposed to the Cloud. Furthermore, given that the majority of edge devices are fog/mist devices, which have limited energy resources, certain attacks exploit this vulnerability by focusing on draining the batteries of these edge devices, causing them to consume a significant amount of energy. An example of such an attack is the Stretch attack [99], which involves sending data packets with headers that contain extended routing and looping paths, compelling these edge devices to consume energy on unnecessary data transmission routes.

2.4.2 Data Integrity

Data integrity refers to the process of confirming the accuracy and consistency of data that is spread across a network. It is crucial to ensure data integrity in EC due to the possibility of intentional manipulation or corruption by adversaries, as well as accidental errors caused by sensor malfunctions or transmission issues [100]. To verify data integrity, a protocol called EDI-V was introduced by the authors of [101]. This approach involves assigning a unique tag to each data block before storing it in an edge node. Subsequently, a reliable and trusted third-party server is responsible for auditing data changes by comparing the initial tags with the latest ones. However, incorporating a third party alone is insufficient to address privacy concerns. To address this limitation, [102] proposed an approach for distributed and lightweight auditing using Merkle Hash Trees.

2.4.3 Access Control

Access control is the process of permitting only authorized individuals to access a specific edge server. To minimize latency caused by authentication, the challenge in EC lies in ensuring lightweight access control [103]. Additionally, the authentication process becomes more difficult in EC due to user mobility and the widespread distribution of edge servers. Almaini et al. [104] propose an authentication mechanism that utilizes a port knocking service on the switch. Port knocking is generally performed by firewalls to authenticate hosts by sending a predefined sequence of ports (TCP SYN packets) before establishing a connection. The suggested approach involves storing authenticated and unauthenticated nodes (connections), represented by IP source and destination addresses, using match-action table rules inserted by the SDN controller. The authors further expand their method in [105] to safeguard against replay attacks by implementing One-time Password (OTP) authentication, where a password is valid for only one transaction. [104] also introduce a timeout for the stored connections to manage available resources and prevent memory-exhaustion attacks. To enhance security and customization, the approach allows network operators to determine the length of the sequence, which refers to the number of ports that need to be knocked. The OTP implementation relies on cryptographic hash functions (e.g., SHA3) to calculate the next expected password and is based on the Leslie Lamport algorithm.

2.4.4 Privacy

In today's context, the issue of user privacy has become a subject of controversy, particularly due to the increasing prevalence of camera surveillance systems and the exploitation of confidential user data by social media platforms. The collection of information by connected edge devices for service provision can potentially compromise individuals' privacy. In the field of data processing, specifically in the domains of CC and EC, there are three primary stages: data cleaning, data aggregation, and data analysis. The cleaned data often ends up being less representative, with fewer attributes compared to the original dataset. To address privacy concerns, a distributed data cleaning algorithm was proposed by [106], which solely requires users to provide data representation without transferring the actual data to the Cloud. Subsequently, for data analysis, it becomes necessary to transfer the data to a centralized server to run analytic models. Numerous approaches have been presented in existing literature to safeguard privacy during this process. For instance, [107] introduced a novel distributed Fully Homomorphic Encryption (FHE) approach, ensuring the security and privacy of sensitive and confidential data. Another approach involves applying lightweight encryption to the data before transferring it, as suggested by [108]. Furthermore, [109] proposes the addition of noise to the data and training ML models with this noisy data. Similarly, [110] suggests multiplying the data with projection matrices before transmitting it to the Edge/Cloud for training purposes. When it comes to data analysis, Federated Learning (FL) stands out as a promising solution that allows the extraction of information from data without centralizing it [111].

2.5 Developments in Resource Management for Edge Computing

The work by Nguyen et al. [112] presents a deep learning-based auction system for allocating EC resources in mobile blockchain networks, aiming to maximize revenue for service providers. This approach is innovative, leveraging an optimal auction analytical solution to design a multi-layer neural network, setting a new standard in resource allocation efficiency and financial optimization. However,

the complexity and computational demands of the model raise concerns about its scalability and transparency. The deep learning aspect, while powerful, could obscure the decision-making process, potentially affecting fairness and participant confidence. Additionally, the model's adaptability to the rapidly evolving landscape of blockchain and mobile technologies is yet to be fully tested. The paper [113] introduces a solution for optimizing task offloading and bandwidth allocation in Mobile Edge Computing (MEC) using Deep-Q Network (DQN). The approach effectively tackles the mixed integer nonlinear programming challenge inherent in such problems, achieving near-optimal performance in resource allocation and cost minimization. This DQN-based method stands out for its efficiency in managing complex scenarios and adapting to varying requirements, such as task sizes and energy consumption patterns. However, it does come with its set of challenges, including the computational intensity required for training the DQN model, the complexity in setting up the model including the intricacies of defining state and action spaces, and a sensitivity to the tuning of hyperparameters like learning rates and batch sizes. Despite these drawbacks, the paper showcases the promising application of deep reinforcement learning in enhancing mobile computing environments by improving efficiency and reducing operational costs. It is worth noting that three prior investigations, partially authored by the same group of researchers and addressing related issues, have contributed to this area of research. These studies collectively introduce a system designed for task management across a diverse user base. They also focus on identifying the optimal timing and type of tasks to offload to the network. To achieve these objectives, Q-Learning is employed to devise an optimal task processing strategy, with a particular emphasis on minimizing energy consumption [114]. The second study [115] focused on leveraging multi-agent reinforcement learning (MARL) to tackle the challenges of computation offloading in the IoT networks through edge computing. The proposed Independent Learners based Multi-Agent Q-learning algorithm facilitates distributed decision-making, enabling each agent or user to independently learn optimal decisions based on local observations without global network state information. This approach not only emphasizes energy efficiency by aiming to minimize the long-term system cost, which encompasses

energy consumption and task execution delay, but also enhances the robustness and scalability of the system, making it well-suited for environments with numerous IoT devices. However, the implementation of such MARL algorithms can be complex due to the necessity for each agent to learn and adapt based on partial network state information, leading to a significant challenge in achieving optimal solutions. Since agents operate independently without comprehensive knowledge of others' actions or the overall system state, there's a risk of converging to suboptimal solutions compared to centralized approaches that globally optimize decisions. Additionally, the decentralized nature of computation offloading could introduce considerable communication overheads for exchanging state, action, and reward information among agents and the central gateway, especially as the number of IoT devices increases. Despite these challenges, the paper offers a promising solution for enhancing energy efficiency and supporting distributed operation in IoT networks. The third study [116] tackles the issue of efficiently managing computation tasks within IoT networks enhanced by EC, using machine learning approaches for optimal resource allocation. This innovative framework, leveraging centralized user clustering and distributed task offloading algorithms, aims to optimize computational resource distribution between local IoT devices and edge servers. By employing machine learning techniques, specifically deep reinforcement learning, the system is designed to dynamically adapt to changing network conditions and user demands, improving its efficiency without manual intervention. This approach ensures minimized system costs, incorporating energy consumption and task execution latency, thereby enhancing energy efficiency and reducing operational expenses. However, the implementation of such a machine learning-based resource allocation system introduces complexity due to the intricate nature of training models and ensuring real-time responsiveness. Additionally, there's a risk of suboptimal performance if the models fail to capture all nuances of the network environment or user behavior. Moreover, the effectiveness of the algorithms heavily relies on the quality and quantity of training data, where inaccuracies or data insufficiency can adversely affect model performance and, subsequently, the efficiency of resource allocation. The authors in [117] introduce a collaborative Q-Learning framework known as the Multi-Agent Reinforcement

Learning network (MARL). This framework aims to reduce energy consumption efficiently. The study employs reinforcement learning to empower an SDN controller in devising a policy that optimizes the overall system's energy usage. Through the MARL framework, agents cooperate and exchange information to enhance their training. In a related work, the authors in [118] propose a novel algorithm called Deep Reinforcement Learning based Resource Allocation (DRLA). This algorithm is designed to ensure the efficient utilization of network and computational resources, aiming to minimize the service time for processing tasks. This service time includes both routing and computing durations. In this proposed approach, time-sensitive or computationally demanding tasks generated by various devices are directed towards the nearest MEC nodes. If a task cannot be processed by the current MEC due to unsuitability, it is rerouted to the next available MEC node until a compatible one is found. The routing decisions are guided by an SDN controller that has been trained through reinforcement learning to make optimal allocation choices. The system exhibits intelligent routing behavior based on network conditions, following a training period. In [119] a novel approach to enhance the efficiency of vehicular ad hoc networks through the integration of EC and software-defined networking for optimized resource allocation. By proposing a three-tier EC framework and utilizing a reinforcement learning algorithm, the model aims to dynamically allocate computational resources and efficiently route to edge servers for real-time vehicle monitoring. This integration significantly reduces latency and maximizes system utility, offering an adaptive solution to resource management in IoT networks for vehicular applications. However, implementing such a system introduces complexities, notably in the continuous learning and adaptation required by the reinforcement learning algorithm, which demands significant computational resources and expert knowledge in networking and machine learning. The reliance on centralized SDN control also raises concerns about scalability and potential bottlenecks in large-scale deployments. Despite these challenges, the model's promise of improved latency and resource utilization presents a compelling case for further exploration in smart vehicular network management. The paper [120] presents a novel approach to dynamic resource allocation for cloud computing environments. This approach is designed

to offer QoS guarantees across multiple classes of clients, addressing the challenge of efficiently provisioning virtual machine (VM) instances with varying service requirements and cost considerations. By employing a reinforcement learning (RL) algorithm, the proposed model adapts the allocation of VMs to meet QoS demands for all client classes while maximizing cloud provider (CP) profits under changing conditions, such as service costs, system capacity, and demand for services. The primary advantage of this approach lies in its ability to dynamically adjust resource allocation in response to fluctuating market conditions and system demands, ensuring QoS guarantees and optimizing CP profits simultaneously. The model's use of an RL algorithm allows for a sophisticated and adaptive strategy that can navigate the complex trade-offs between client satisfaction and cost efficiency. However, the implementation of this model brings with it several challenges. Firstly, the complexity of the RL algorithm and the need for continuous adjustment based on real-time data may require significant computational resources and expertise in machine learning. Secondly, the model's reliance on accurate predictions of service demand and cost can introduce uncertainties, particularly in rapidly changing or unpredictable market environments. Lastly, while the model aims to balance the interests of both clients and CPs, there is a risk of suboptimal outcomes if the RL algorithm does not adequately learn from the environment or if the model parameters are not appropriately tuned. In the study [121], a novel multiagent deep reinforcement learning resource allocation algorithm is introduced to address the challenges of decentralized radio resource management within 5G vehicular networks. This approach leverages an actor-critic technique, employing centralized training to facilitate information sharing among agents, thereby enhancing their ability to learn each other's policies. During decentralized execution, agents utilize their actor networks in conjunction with local observations to make optimal decisions regarding transport block selection and transmission power. The proposed method showcases a notable improvement, with an 18% higher packet reception ratio when compared to a spectrum allocation scheme based on double DQN, and a remarkable 33% higher reward when compared to prior state-of-the-art Multi-Agent Reinforcement Learning (MARL) techniques. In another study [122], deep learning techniques are explored for predicting vehicle mobility patterns in Vehicle-to-Everything (V2X) networks,

with the ultimate goal of optimizing channel allocation and enhancing network performance. The proposed architecture combines centralized decision-making with distributed channel allocation strategies. This is achieved by employing DQN and Advantage Actor-Critic (A2C) techniques in conjunction with Long Short-Term Memory (LSTM) networks to effectively account for the dynamic nature of user mobility. To validate the efficacy of the resulting LSTM-DQN and LSTM-A2C algorithms, extensive simulations were conducted using real data sourced from the California State Transportation Department. The researchers in [123] have tackled the challenges associated with cloud computing when it comes to applications that require low latency and energy efficiency. They have done so by introducing a novel approach, which involves the deployment of a distributed deep neural network (DNN) within an Intelligent Software Defined Networking (ISDN) framework. This ISDN framework effectively manages network resources, including bandwidth and computational capacity, using the SDN paradigm. To ensure high-quality communication and efficient DNN task offloading, the team has devised a dynamic routing technique for optimizing the quality of service. Additionally, they have developed a task offloading model based on the Markov Decision Process, aiming for the optimal distribution of DNN tasks. In general, prior efforts were centered around devices and tasks, with a primary focus on whether a task should be offloaded from a client and how much computing power a task should receive from a resource [113]–[117]. In contrast to prior works, this thesis emphasizes the efficient distribution of tasks from different clients to available resources. Specifically, we address the challenges of resource volatility, where resources may appear or disappear from the network at any time. While previous works have considered managing multiple resources, these were often controlled by a central or small intelligent controllers that managed local areas [114]–[116], [118], [119], we aim to organize AI controllers in a hierarchical manner. This approach enables a controller to manage other controllers or devices at lower levels. This hierarchy provides the ability to acquire additional resources from other networks by requesting assistance from higher-level controllers. Moreover, in the event of a controller failure, another controller can assume management responsibilities.

2.6 Resource Management through Machine

Learning

In the evolving landscape of EC, Machine Learning and Deep Learning technologies are revolutionizing the approach to resource management, optimizing resource allocation, enhancing task offloading strategies, and improving computational efficiency across EC environments. EC aims to minimize latency and boost processing power at the network's edge, closer to the sources of data generation, the end-users and IoT devices. However, this advancement introduces complex challenges in efficiently managing resources. Key issues include the need for dynamic task offloading, computational resource allocation, and adapting to fluctuating application demands under variable network conditions and user mobility. ML and DL emerge as transformative tools within EC, capable of learning from historical data, predicting future system demands, and making informed decisions. These technologies enable EC systems to dynamically adjust to changing conditions, optimizing resource use, minimizing energy consumption, and reducing latency effectively. ML/DL models excel in making informed offloading decisions, choosing between local processing and offloading to the edge or cloud based on current network conditions, device energy constraints, and application requirements. This dynamic decision-making is crucial for maintaining operational efficiency and user satisfaction. Through analyzing data traffic patterns and resource demands, ML/DL algorithms guide the strategic distribution of computational resources across EC servers, enhancing system efficiency and ensuring that user experiences remain uncompromised due to potential resource shortages. The dynamic nature of EC, characterized by changing network topologies and user mobility, necessitates adaptive models. Reinforcement learning models, in particular, allow systems to evolve based on continuous feedback, maintaining high performance and reliability amidst environmental shifts. Practical applications of ML/DL in EC, including smart city operations, IoT device management, and real-time analytics, showcase the significant benefits of these technologies. They demonstrate ML/DL's capability to enhance resource management, reduce latency, and improve user experiences in edge computing contexts. The convergence of ML/DL with EC

marks a new era in computing, where systems not only respond more efficiently but also predict and adapt to future scenarios. As EC supports an expanding range of applications, from autonomous vehicles to augmented reality, ML/DL's role in developing intelligent, efficient, and adaptive resource management strategies becomes ever more critical. Insights from extensive surveys and case studies highlight ML/DL's pivotal role in advancing the EC ecosystem, paving the way for future innovations in edge computing. This integrated approach to ML/DL applications in EC resource management underscores the potential of these technologies to revolutionize edge computing, offering a roadmap for navigating the challenges and leveraging the opportunities within the EC landscape [124] [82]. Tables 2.1 to 2.3 present various studies that explore machine learning methods and their potential applications in resource management. Additionally, these tables outline the advantages and disadvantages of each method in relation to the specific objectives of the study. Furthermore, table 2.4, followed by table 2.5, provides a comprehensive summary of recent studies on Machine Learning-Based Resource Management, detailing algorithms, simulation tools, and optimization objectives.

2.7 Developments in Edge Security for SDN

In recent years, the field of SDN has seen significant advancements in exploring programmable data planes to overcome the limitations inherent in traditional SDN paradigms, such as OpenFlow. These traditional paradigms often require controller involvement in every decision-making process. Moreover, these innovations aim not only to address latency issues but also to enhance security functions within the data plane. One of the notable advancements in this field is the development of the P4 language, a domain-specific programming language designed for describing how packets are processed by the network data plane. P4 allows programmers to specify the packet processing behaviors of switches, routers, and other network devices, independent of the specific hardware or software platform. This capability enables the customization of network forwarding behavior, facilitating the implementation of high-level policies for traffic manage-

Table 2.1.: Machine Learning Methods and Their Potential Applications in Resource Allocation [124].

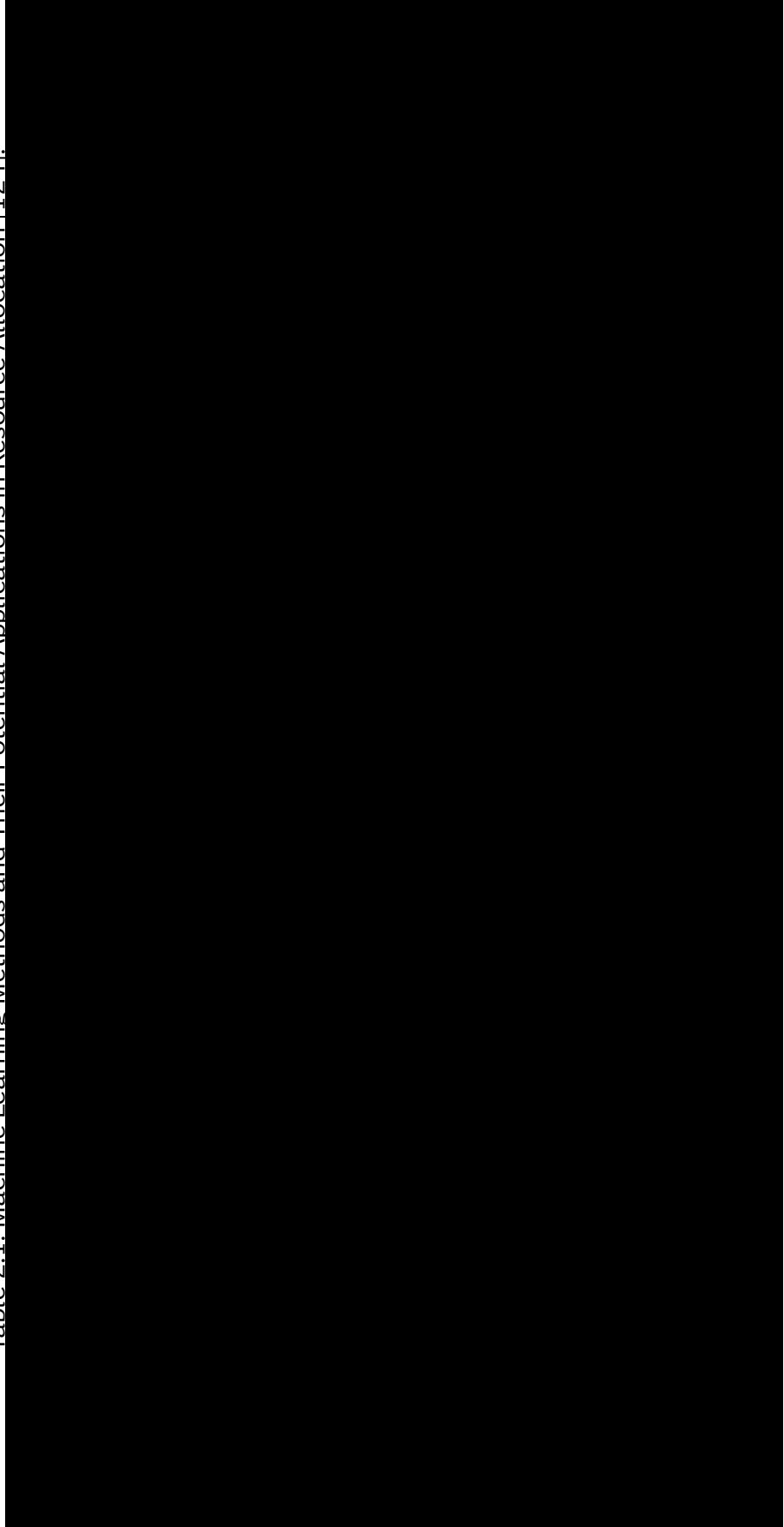


Table 2.2: [Continued] Machine Learning Methods and Their Potential Applications in Resource Allocation

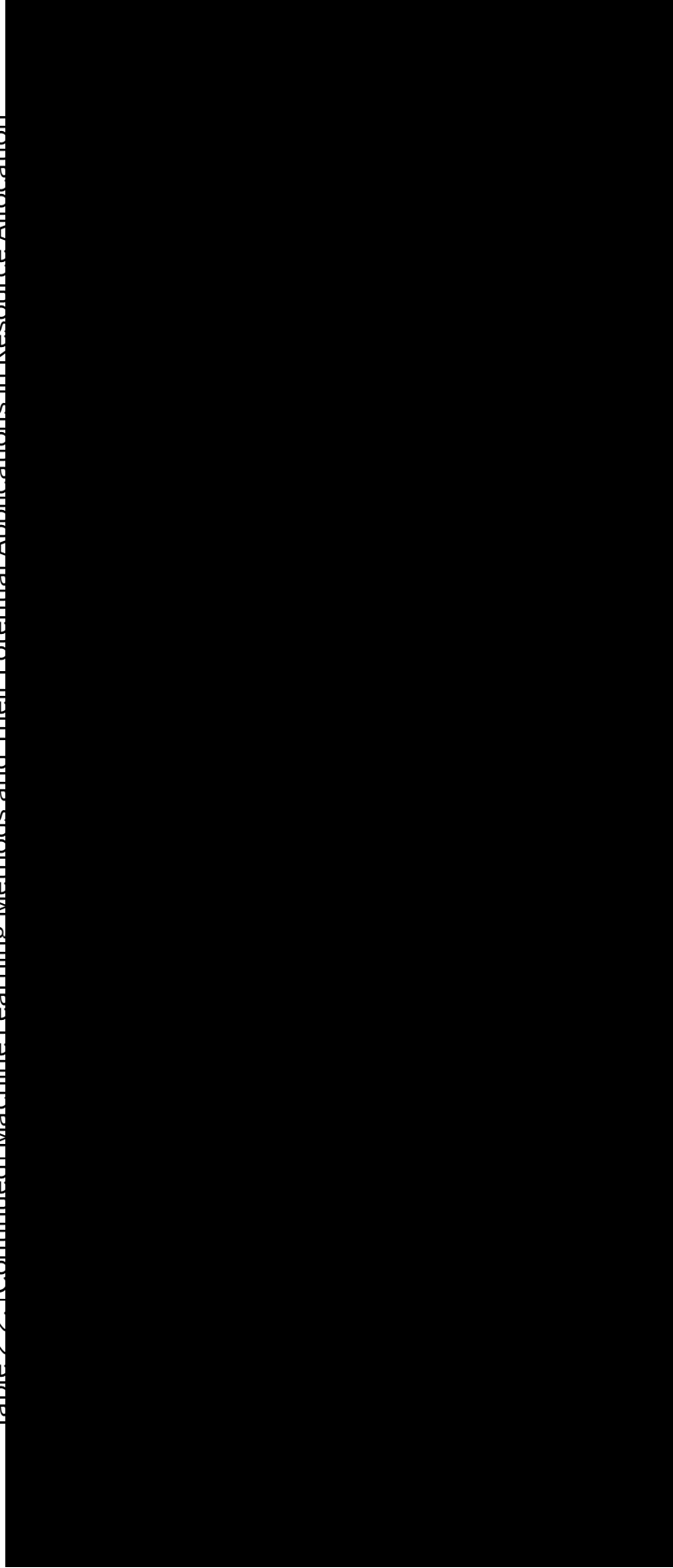


Table 2.3: [Continued] Machine Learning Methods and Their Potential Applications in Resource Allocation.

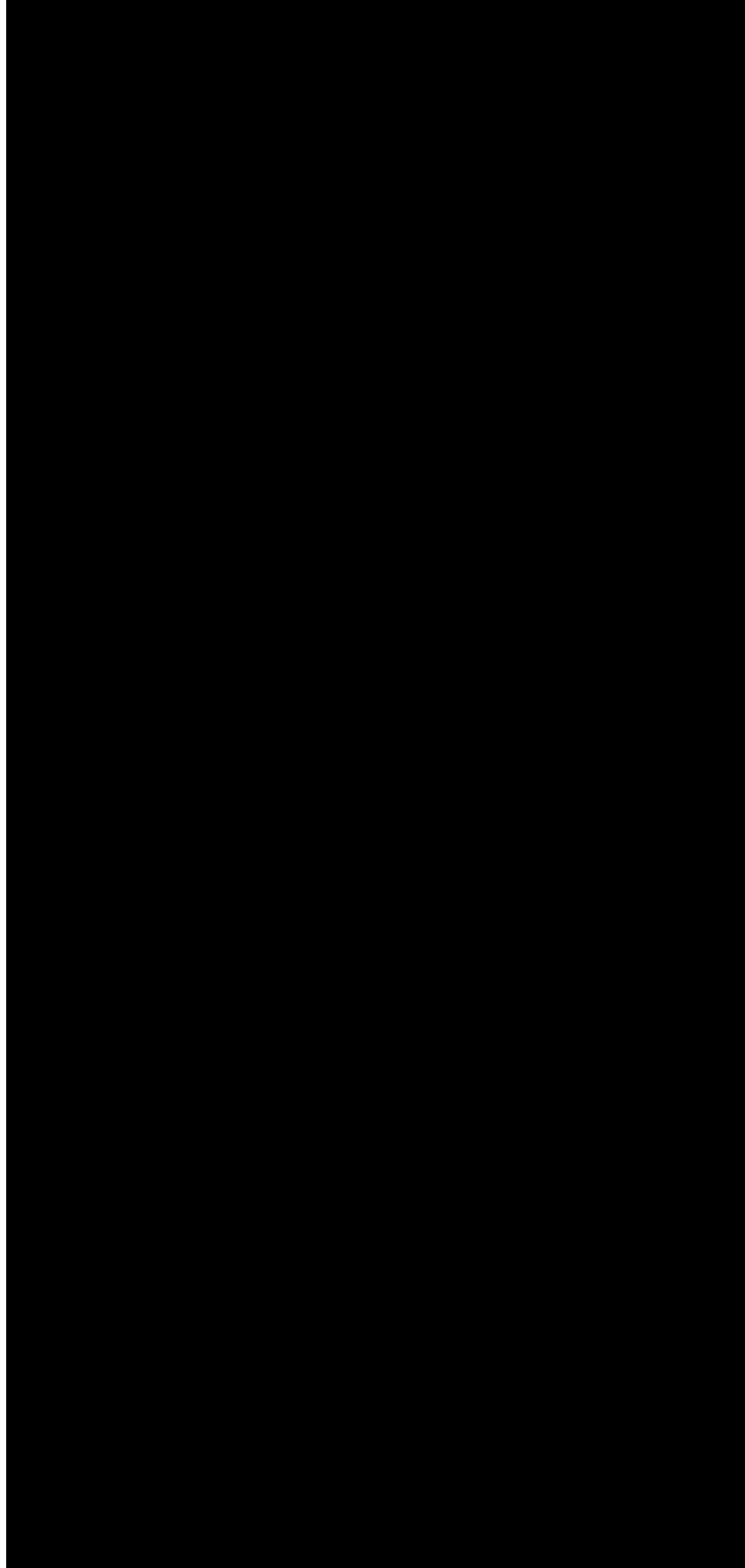


Table 2.4: Summary of Research on Machine Learning-Based Resource Management: Detailing Algorithms, Simulation Tools, and Optimization Objectives [124].

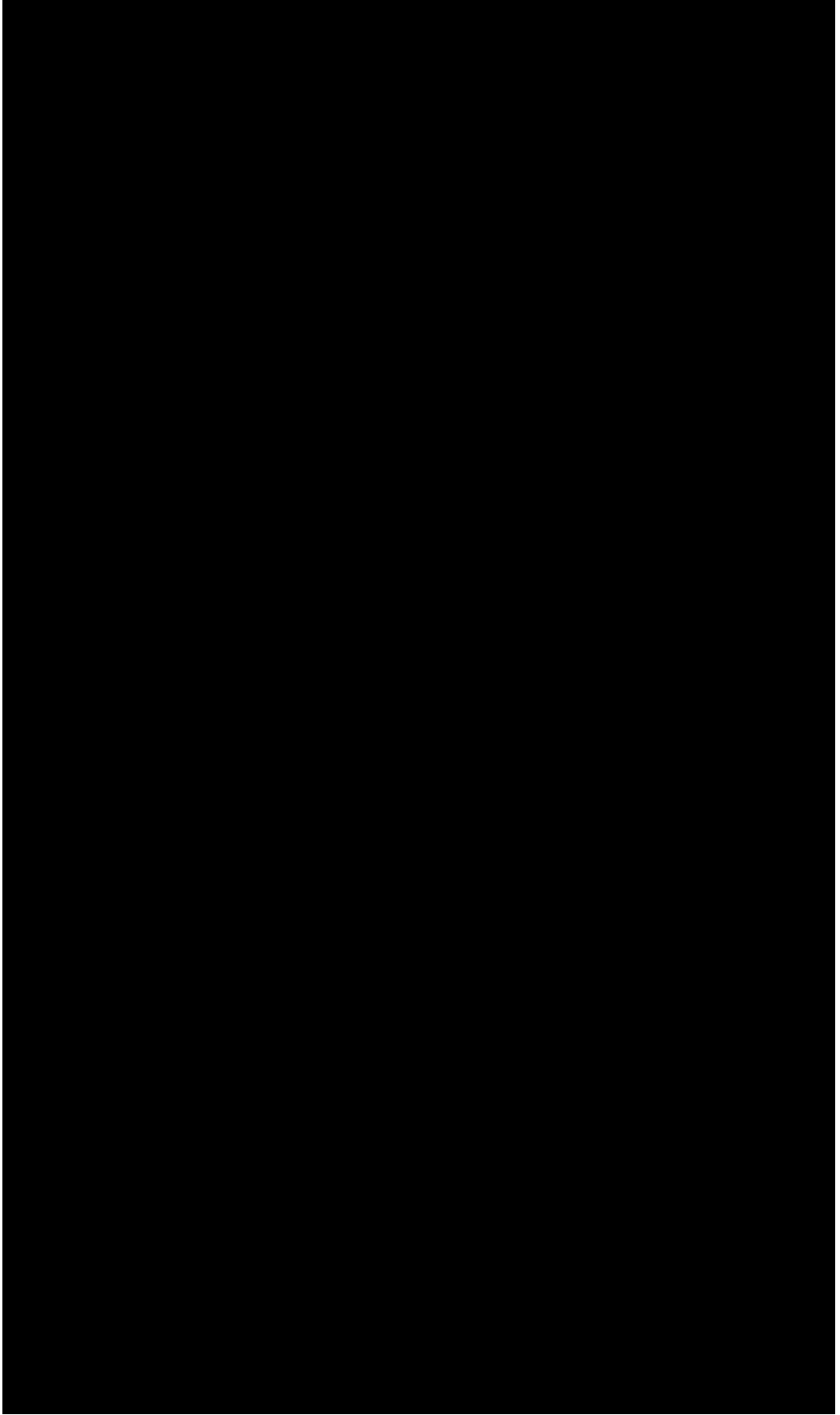
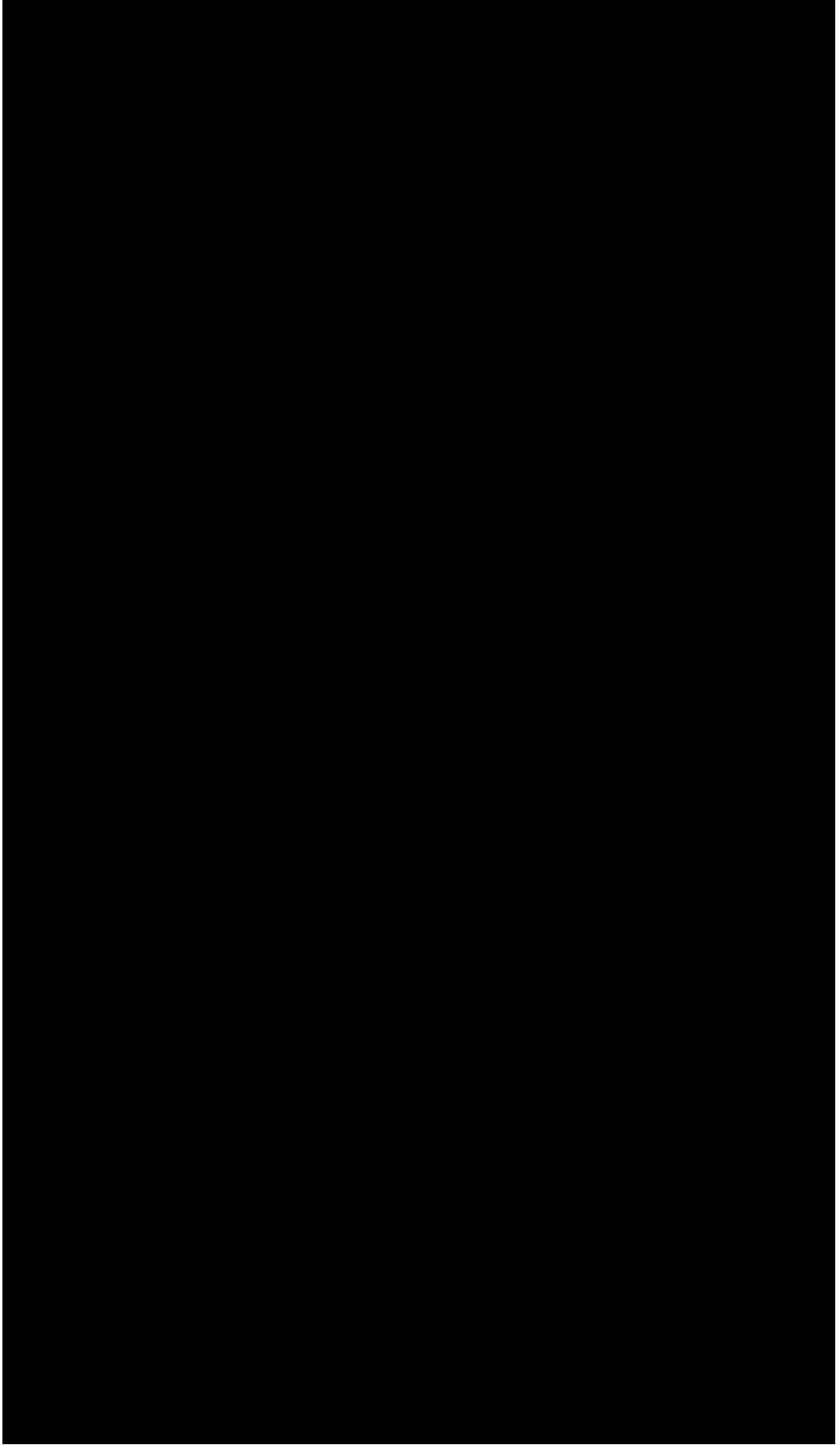
A large black rectangular area representing a redacted table. The table content is completely obscured by a solid black fill.

Table 2.5: [Continued] Summary of Research on Machine Learning-Based Resource Management: Detailing Algorithms, Simulation Tools, and Optimization Objectives

A large black rectangular area covering the majority of the page, indicating that the content of Table 2.5 has been redacted.

ment, network monitoring, and security functions. P4 strives to offer a flexible and efficient means to control and adapt the behavior of network devices in real-time, optimizing network performance and functionality to meet specific needs and applications. One notable contribution to this evolving landscape is the work of F. Paolucci et al. [171], who proposed the integration of P4 technology into SDN multilayer packet-over-optical networks, thereby facilitating advanced data plane programmability. Their architectural framework introduced edge nodes equipped with P4 switches boasting deep packet inspection capabilities. This P4-enabled node operates autonomously, harnessing stateful processing at wire speed without controller intervention. Furthermore, the research introduced dynamic P4-based traffic engineering solutions, such as traffic offloading, within a multilayer network context. Notably, the incorporation of a P4-based DDoS mitigation proof-of-concept offered augmented firewall capabilities for safeguarding internal edge resources without necessitating dedicated hardware. Experimental results underscored impressive scalability, with minimal switch latency, showcasing the feasibility and performance of P4 implementations. The work in [172] tackled the issue of computation-resource constrained controllers, particularly in scenarios characterized by heavy flows leading to substantial delays. Their proposed solution, a hierarchical edge-cloud SDN controller system, sought to enhance network scalability while meeting QoS requirements. By distributing computational tasks across edge and cloud resources based on traffic loads, their architecture provided an adaptable approach. Demonstrating the system's effectiveness, even at large scales, without compromising overall performance, it maintained remarkable stability even under fluctuating traffic loads. Sivaraman et al. [173] introduced HashPipe, a heavy hitter detection algorithm leveraging programmable data planes with P4. Their approach advocated continuous heavy-hitter monitoring at all network switches to promptly respond to transient traffic fluctuations. The identification of flows contributing substantial traffic to a link was deemed valuable for a variety of network applications. In [174], authors proposed security enhancements in the data plane employing P4, manifesting as a second-generation firewall. This firewall operated as a packet filter with stateful memory, contributing to improved security within SDN environments. Addressing

the critical concern of denial of service attacks, especially anti-spoofing techniques in the SDN data plane, [175] introduced an SDN-based system designed to mitigate such attacks effectively. Li et al. [176] presented LOSSRADAR, a lightweight packet loss detection service capable of swiftly identifying the locations and 5-tuple flow information of lost packets. Bianchi et al. [177] explored the utilization of eXtended Finite State Machines (XFSM) in their OpenState study. Their research contended that programming should allow specifying how states on a switch are managed, directly executing these specifications without controller interaction. Proposing modifications to OpenFlow 1.1, they enabled stateful processing of flows within switches. Kabasele et al. [178] contributed a two-level network-based Intrusion Detection System (IDS) tailored for Industrial Control Systems (ICS) utilizing SDN. The first level, implemented in P4 on network switches, operated as a whitelist-based filter. If no matching whitelist entry was found, the packet was forwarded to the second level of IDS, a security engine running on a dedicated host. This engine determined the packet's malicious or benign nature, instructing the controller to update switch tables accordingly. In [179], authors introduced an approach to secure access control systems by combining SDN capabilities with RADIUS-based user authentication. Their approach initially centralized security tasks within the controller, potentially creating bottlenecks and vulnerabilities. However, in our work presented in chapter 4 of this thesis, we proposed an innovative approach. This approach delegated authentication tasks typically requiring sophisticated equipment to SDN switches, using port knocking as a practical example. Demonstrating how a single SDN switch programmed with P4 could efficiently perform port knocking and authentication at the network ingress, this work represented a pioneering effort in delegating authentication entirely to the data plane using P4 technology.

2.8 Developments in Homomorphic Encryption

Over the past decade, the landscape of cryptography has witnessed a remarkable surge in research endeavors dedicated to exploring the diverse applications of FHE technology. FHE, a groundbreaking encryption paradigm, offers an al-

luring solution for the secure processing of sensitive user data within public domains. It accomplishes this feat by enabling computations on encrypted data without divulging its contents, thus preserving the utmost confidentiality and safeguarding user privacy. Nevertheless, the widespread adoption of FHE has been hampered by its formidable computational complexity and the substantial time investment required for data encryption. These limitations have prompted researchers to embark on a quest to unravel innovative strategies aimed at augmenting the efficiency and practicality of FHE. One of the most promising avenues toward enhancing the efficiency of FHE revolves around the utilization of specialized hardware accelerators, such as Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), to expedite the execution of homomorphic encryption schemes. A pioneering work in this direction, as demonstrated by the authors in [180], involved the implementation of an FHE scheme on an NVIDIA C2050 GPU, harnessing the power of the Fast Fourier Transform (FFT) algorithm to create a compact parameterized version of the lattice-based FHE system originally conceived by Gentry and Halevi [181]. This pioneering effort yielded substantial speed improvements, with encryption, decryption, and re-encryption operations achieving acceleration factors of 7.68, 7.4, and 6.59, respectively. By effectively leveraging the parallelization capabilities of GPUs and harnessing the FFT technique to address the computational bottleneck associated with large modular multiplications, this work marked significant progress in FHE acceleration. However, it still confronted practicality challenges, primarily attributed to high latency during encryption and re-encryption operations. To alleviate the computational burden of FHE on hardware with limited processing capabilities, such as IoT devices, the authors of [182] introduced two innovative techniques. The first, termed SHE+FHE, entailed encrypting plaintext using Somewhat Homomorphic Encryption (SHE) on the IoT device itself, which is computationally less demanding than FHE. Subsequently, the SHE ciphertext was transformed into FHE ciphertext on the more capable cloud service side, where computational constraints were less of a concern. The second technique, dubbed SHE+TRIVIUM+FHE, incorporated the TRIVIUM secret key cryptosystem to encrypt the plaintext and SHE public key cryptography to encrypt the TRIVIUM key. According to the results

of their experiment, the authors confirmed that this technique can significantly reduce the load on IoT devices. In [183], an extensive comparative analysis of two prominent FHE schemes, namely BFV and CKKS, was undertaken using the widely adopted Microsoft SEAL FHE library. This meticulous investigation revolved around evaluating the performance of these schemes concerning the time required to execute specific arithmetic operations. Through an exhaustive examination of the time consumed by each scheme during these operations, the study provided invaluable insights into their relative efficiencies, serving as a foundational reference for future research in the field. Addressing the persistent challenge of high computational overhead associated with FHE, [184] presented a novel solution: F1, an FHE programmable accelerator. F1's uniqueness according to the authors lies in its design philosophy, which distinguishes it from other FHE acceleration approaches. This accelerator capitalizes on high-throughput functional units tailored to expedite fundamental computations commonly encountered in higher-level operations. Moreover, the co-design of the compiler and hardware components minimizes data movement, a notorious bottleneck in traditional FHE acceleration schemes. In the pursuit of accelerating the encryption and decryption operations of the BFV homomorphic encryption scheme, [185] unveiled two innovative hardware architectures. By harnessing high-performance polynomial multipliers, the authors achieved remarkable speed enhancements for these critical operations. Employing a hardware/software co-design approach, encryption and decryption tasks were offloaded onto an FPGA, while remaining operations were executed within software on a conventional desktop computer. This approach yielded a substantial reduction in encryption and decryption times, with encryption time plummeting by a factor of approximately 12 and decryption time by approximately 7. These outcomes underscore the immense potential of hardware-accelerated homomorphic encryption schemes in substantially elevating the performance of encryption and decryption operations. The collective efforts of researchers to amplify the efficacy of FHE schemes are palpable in the plethora of hardware architectures proposed in works such as [186] and [187]. These architectures aim to optimize the efficiency and speed of FHE operations, showcasing the dedication of the scientific community to advance secure data

processing. Furthermore, the integration of GPU accelerators, as evidenced in [188], [189], and [190], has emerged as a potent strategy for enhancing FHE's performance, signaling a growing interest and substantial investment in the development of more efficient and secure FHE solutions. In the context of this thesis, we propose a novel distributed FHE approach [107] presented in chapter 5 tailored to process sensitive and confidential data while steadfastly upholding the tenets of security and privacy. Our innovative distributed FHE framework holds great promise for handling extensive volumes of confidential data within untrustworthy public environments, providing an exemplary solution that bridges the gap between advanced encryption techniques and real-world application scenarios.

2.9 Developments in the Detection of Spoofing

Attacks in Autonomous Driving

Numerous studies have been conducted focusing on defenses against LiDAR spoofing. The countermeasures proposed in these studies are based on a diverse range of strategies, each with distinct design principles. The effectiveness of these strategies varies, presenting certain advantages and disadvantages. In this section, we will provide a concise overview of some relevant prior research, organized into four primary categories:

Adjustment of the LiDAR sensor: LiDAR sensors commonly employ a broad receiving angle on the receiver. However, a reduction in the receiving angle could potentially minimize the LiDAR sensor's attack surface. This alteration compels an attacker to align more accurately with the LiDAR sensor's line of sight to successfully execute an attack. If the attacker's pulse is delivered from an excessively deviated angle, it will not be detected. For instance, the receiving angle of the Velodyne VLP-16 LiDAR could be reduced to as low as 0.0048° . Given the rapid speed of the light pulse, the rotational speed of the LiDAR sensor becomes negligible, thereby allowing its own signal to be conveniently captured even with such a reduced angle. However, this approach imposes constraints related to diminished sensitivity and maximum range reduction. Furthermore, it does not provide absolute protection since a strategically placed attacker could still conduct a spoofing attack [191], [192].

Redundancy and Sensor Fusion: The objective of redundancy is to enhance the robustness of the entire system by increasing the count of identical sensors, which monitor the same region. If an anomaly is detected by one sensor but not by the others, it could be indicative of a malfunction or an attack. To execute a successful attack, the intruder would need to compromise multiple sensors in order to deceive the system [193]–[195]. In a parallel strategy, Sensor Fusion integrates different types of sensors, such as cameras, radar, LiDAR, and others, to contrast data from varied sources. This strategy elevates the complexity of a potential attack, as the attacker would need to have extensive knowledge across diverse sensor types and employ a wide range of attack tools to exploit them effectively. However, both strategies carry the disadvantage of increased costs due to the need for a larger sensor array [192], [193], [195]–[198]. These methods can also be extrapolated to a Vehicle-to-Vehicle (V2V) paradigm, wherein vehicles within the same environment exchange information. If there are discrepancies between the data shared by two vehicles, this might suggest an ongoing attack [193], [199]. For the effective operation of the aforementioned three methods, it is essential that the fields of view of the sensors or vehicles overlap, as non-overlapping fields of view create opportunities for attacks.

Randomness: Several strategies can be employed to incorporate randomness into a LiDAR sensor, thereby increasing the challenge for an attacker attempting to synchronize with a target system. The most straightforward technique involves randomly switching the sensor off and on, creating a variable number of pulse signals sent within a specific time frame. The success of an attack depends on the attacker's ability to predict when a pulse is emitted from the victim's transmitter, as the deceptive pulse must reach the victim receiver within a defined time window to be perceived as legitimate. However, pulse omission can negatively impact the recording quality of the system [191]. Alternatively, the unpredictability of the system can be enhanced by randomizing the rotor direction, complicating an attacker's ability to foresee the direction of the next pulse and thereby spoof it effectively. Nevertheless, this strategy increases system complexity due to the need for internal compensation of the introduced randomness [192]. Another proposed strategy is waveform randomization of the laser pulses, with the receiver

configured to only accept reflected signals sharing the same waveform. Although this method can increase security, it does not provide comprehensive protection against all spoofing attacks, leaving vulnerabilities that permit attackers to inject points that are further away than the attacker's actual position [191], [192].

Machine Learning (ML): Recent advancements in artificial neural networks have popularized a novel method, particularly due to its outstanding performance in image processing and object recognition. This ML approach has seen increasing application in the autonomous driving sector of the automotive industry, specifically for detecting active system attacks [191], [200]. In a study by [201], the authors employ a 3D shadow effect to verify the authenticity of an object and discern if it was inserted via a spoofing attack. The 3D shadow effect is described as the void space behind an object when it's scanned using a LiDAR sensor, as the light pulses, unable to penetrate the object, fail to illuminate the area directly behind it. These shadow regions, combined with the detected object, help verify the object's authenticity. A shadow-less object may indicate a spoofed insertion, whereas a shadowed one likely points to a real entity. The entire procedure comprises two stages: preparation and subsequent shadow analysis. During the preparation phase, the LiDAR-generated point cloud is transmitted to an object detector, which identifies the presence of objects within the scene. Using a dedicated algorithm, the detector then envelops these objects in bounding boxes, offering information regarding their width, depth, and height. Subsequent shadow analysis examines the 2D shadows of these detected objects. A bird's-eye perspective aids in identifying whether an object casts a shadow or not. Shadow contamination, often due to noise such as a larger vehicle obscuring the shadow of another, renders the 2D shadow analysis approach ineffective. Consequently, this approach was rejected in favor of a 3D shadow method. This method calculates the volume of the unlit space behind a vehicle using the bounding boxes, allowing for noise elimination within the shadow and achieving a more accurate classification [201]. In this work, the same preparation phase is employed, but with an alternate shadow analysis technique that uses the 2D shadow approach. To mitigate the problem of noise contamination within the shadow, we introduced a methodology that eliminates any object taller than the detected object. Con-

sequently, if a taller object is located behind the detected object, it is removed due to its size, thus revealing the shadow again. This procedure is repeated for each object detected within a critical range. In our work presented in chapter 6 of this thesis, we proposed a novel technique that leverages a reduced 2D dataset derived from the 3D point cloud to detect ghost object attacks. The reduction in data volume is a vital aspect, greatly impacting the optimal utilization of available resources at the edge [202], [203]. Diverse computing resources possessing varying computing power can more readily be allocated to process the reduced data.

2.10 Summary

In this chapter, we conducted an in-depth examination of relevant literature closely aligned with the primary objectives of this thesis. Our research focuses on several key contributions, including the design and implementation of an intelligent system for adaptive resource management in EC. Additionally, we present an efficient lightweight approach for authenticating edge devices and introduce a scalable distributed security scheme using HE. These latter contributions directly address critical security concerns in resource allocation within the context of EC environments. The chapter underscores the pressing need for our innovative contributions, as we evaluate prior research in these areas. Our work not only aims to enhance the overall efficiency of EC but also presents a novel real-time detection technique. This technique is dedicated to improving the safety and efficiency of autonomous driving by effectively identifying LiDAR spoofing attacks. The synthesis of previous research findings and the introduction of our novel contributions underscore the significance of our work in advancing the field of EC.

3 Neural Network-Driven Resource Management in Edge Computing

3.1 Introduction

The core focus of this thesis is presented in this chapter. We have previously explored the criticality of effective resource management and identified the unique challenges in the context of EC, characterized by a dynamic environment of diverse devices with varying capacities. Additionally, we touched upon the fluid nature of EC resources, resulting from factors such as device mobility, joining and exiting of devices, and the dynamic resource requirements of different tasks. Considering these complexities, we highlighted the need for adaptable and versatile resource allocation strategies. In response, we propose a reinforcement learning-based framework for resource allocation capable of dynamically assigning resources to reduce service delays and ensure balanced resource utilization at the edge. Reinforcement learning (RL) is a branch of machine learning where an agent learns decision-making by taking actions in an environment to achieve specific goals. The agent learns from the consequences of its actions, instead of being explicitly instructed on what to do. Essentially, through trial and error, the agent identifies actions that yield the greatest rewards by receiving feedback in the form of rewards or penalties. Unlike supervised learning, which relies on labeled data, RL learns through interaction with the environment. This approach is particularly well-suited for scenarios where explicit examples of correct behavior are difficult to obtain. Reinforcement learning agents are capable of adapting to changes in their environment, making them ideal for dynamic and uncertain contexts. Our approach stands in contrast with previous works [113]–[117], which focused primarily on the offloading of tasks and allotment of computing power from resources. Our work accentuates the efficient distribution of tasks from

multiple clients to available resources, with a particular attention to resource volatility. We differ from past literature in the manner of organizing AI controllers, we propose a hierarchical structure as opposed to a centralized or localized one [114]–[116], [118], [119]. This hierarchical approach allows a controller to govern other controllers, ensuring effective management of dynamic changes, and enabling the acquisition of additional resources when needed. The end goal is to create a hierarchical organization of AI controllers, with this study presenting the main component of the system, linking user space devices to an AI controller in the EC layer. The proposed system is visualized in figure 3.1, designed to operate on three levels: userspace, EC, and CC. Our framework aims to optimize task allocation and resource utilization, with the ability to leverage the cloud’s power when required. In this chapter, we delve deeper into the details of our proposed framework, providing comprehensive insights into its structure, functionalities, and potential benefits, thereby substantiating our solution as a pragmatic answer to the challenges encountered in EC resource management.

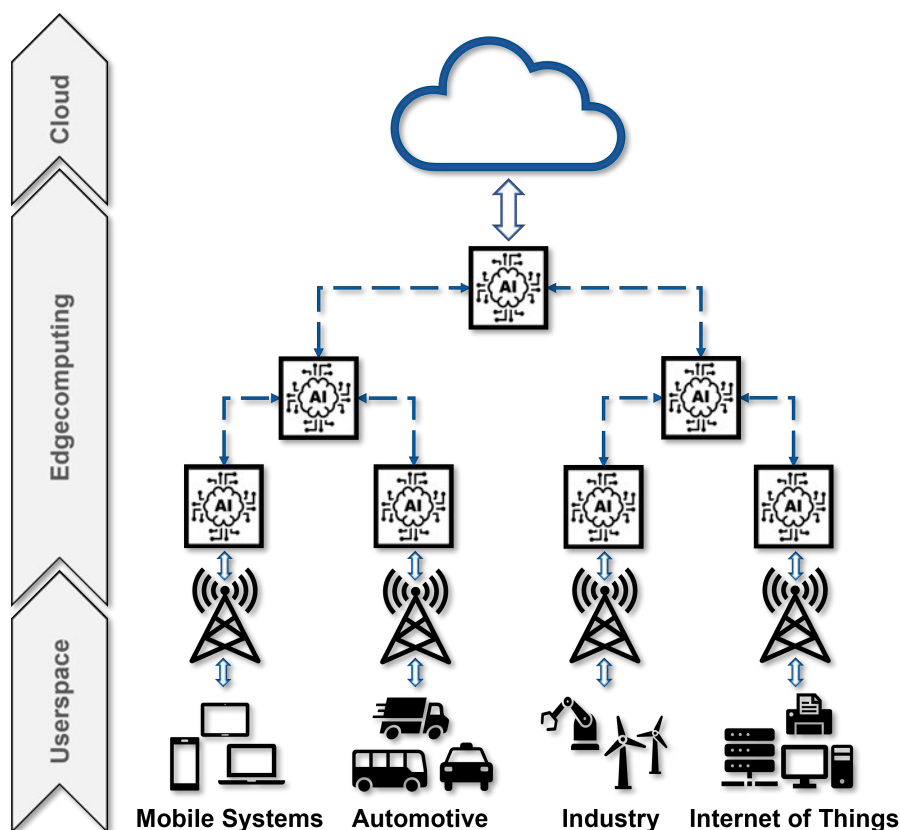


Figure 3.1: Hierarchical Structure for Resource Allocation in the Edge – An Overview

3.2 Traditional Techniques vs. ML Techniques for Resource Allocation

Traditional resource allocation techniques are broadly divided into four categories: approximation-based, heuristic-based, metaheuristic-based, and game-theoretic-based. Approximation methods aim to find a quasi-optimal solution to NP-hard problems within polynomial time [204], with a guarantee of proximity to the optimal solution [205]. Heuristic techniques, described as approaches to problem-solving that utilize a practical method not necessarily optimal, perfect, or rational, but sufficient for achieving immediate, short-term goals, are tailored for specific problems and can identify reasonably good solutions within an acceptable timeframe [206]. Unlike heuristics, which are problem-specific, metaheuristic approaches are designed for a broad range of optimization problems [206] and offer the advantage of exploring a larger solution space, potentially leading to better performance in terms of total execution times for applications. However, as the number of tasks increases, the runtime of metaheuristic algorithms tends to escalate quickly, making them less suited for large-scale IoT applications [207]. Game theory, an applied mathematics branch, explores interactive decision-making scenarios where each player's outcome is influenced by the collective actions of all involved [208].

3.2.1 Challenges with Conventional Resource Allocation Techniques

The complexity of resource allocation issues, classified as NP-hard, means that solutions derived from traditional methods do not reach global optimality. These conventional techniques struggle with meeting varied QoS demands and with adaptation in dynamic settings. Broadly, the limitations of traditional resource allocation methods include [124]:

- **Computationally expensive:** Traditional approaches see their execution time increase in proportion to the application's size, such as the number of tasks, leading to significant computational time overheads. This makes them unsuitable for applications sensitive to delays or those that are data-intensive.

- **Slow convergence:** Conventional resource allocation strategies exhibit slow convergence rates as they are unable to leverage learning from past sub-optimal solutions.
- **Limited adaptability:** Solutions yielded through traditional methods are prone to sensitivity towards changes in the environment. These methods often assume a static and known computing environment by mobile users. Changes in environmental parameters necessitate the reformulation of the optimization problem to incorporate the new parameters and achieve the intended outcomes. Consequently, traditional resource allocation frameworks lack the flexibility needed for time-varying and dynamic environments.

3.2.2 Why Do We Need ML/NN

The increasing number of IoT devices connected to the Internet is producing vast amounts of data, including pictures, audios, and videos. Given that this data originates at the network edge, it is more advantageous to process it there. ML/NN techniques are crucial in this scenario because of their capability to efficiently analyze and extract features from large volumes of data swiftly. Furthermore, for effective execution and analysis of the generated data, it is essential to correctly allocate (offload and schedule) the data to edge computational resources that can meet the data requirements, such as latency, privacy, and Quality of Experience (QoE). ML/NN plays a key role in data prediction, enabling accurate forecasts of both the data requirements and the EC nodes that will process the data [124]. Particularly, NNs have several advantages over other machine learning models due to their architecture and operational principles. NNs excel at capturing and modeling complex, non-linear relationships within data. This is due to their layered structure, where each layer can learn different aspects of the data and combine them in complex ways. Unlike traditional machine learning models that often require manual feature engineering or selection, NNs are capable of automatically discovering and learning the features directly from raw data. Additionally, NNs are highly scalable to large datasets, thanks to their capacity to learn from vast amounts of data. This is complemented by their adaptability to various types of data (e.g., images, text, and sound) and tasks (e.g., classification, regression, and sequence prediction) [209].

3.3 AI Resource Allocation Structure

This research study introduces the development of a comprehensive system model, as depicted in Figure 3.2, formulated based on the defined objectives, insights from previous research, and problem statement. The model consists of three main components: resources, an AI agent, and clients. The clients are represented as a variety of devices within the network, denoted as $Client_1, Client_2, \dots, Client_n$. Each client generates a series of tasks, $Task_1, Task_2, \dots, Task_m$, depicted by the squares above the respective clients in the diagram. These tasks are distinguished by their unique features, as indicated by different numbers and colors. To ensure efficient processing, tasks are assigned only to resources equipped to handle their specific requirements. Upon creation, a task is immediately sent to the AI agent, which comprises two modules: the agent logic, forming the core framework, and a NN responsible for selecting the appropriate resource for each task based on task data and resource attributes. The agent logic compiles the task and resource information into a state, which is then processed by the NN. Task information, contained within the task frame, includes application type, priority, time constraints, and data to be processed. Resource information, on the other hand, is gathered for each task, represented in the diagram by a green line flowing from the resources to the agent logic. This includes details such as power capacity, supported applications, and current task load. Utilizing the provided input, the NN identifies the most suitable resource for each task and relays this decision to the agent algorithm, which then routes the tasks to the designated resources for processing. In this thesis, the agent was developed to operate within the Mininet environment, employing TensorFlow for the predictive component and Python for additional implementation aspects. However, the agent is a self-contained application capable of functioning in any suitable environment. The overarching goal is to establish a hierarchical system of several agents distributed across different levels between the edge and cloud. In this system, an agent can request assistance from a higher-level agent, representing a type of super resource, when no suitable resources are available to meet the clients' demands. This thesis demonstrates the functionality of a single agent, with the extension to a cross-level multi-agent structure being straightforward and envisioned for future work.

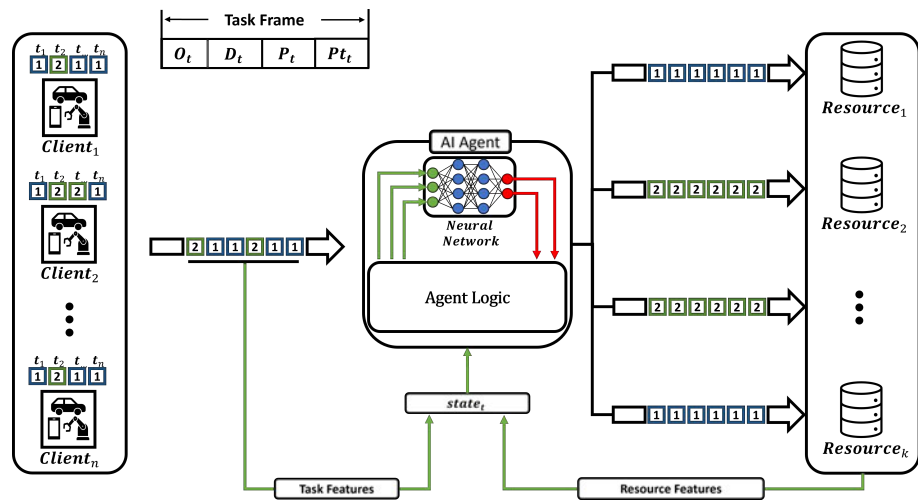


Figure 3.2: Schematic representation of the AI-based resource allocation system model

3.3.1 State

The decision-making process of the NN is based on a specific set of information, referred to as the state. This set includes a finite list of individual elements, each corresponding to data processed by the neurons in the input layer. Specifically, the state at any given time is represented as

$$state_t = \{O_t, D_t, P_t, P_t_t, U, Pw, Rt\}$$

, where the subscript t indicates the task to which the information pertains. As demonstrated by the use case chosen in this thesis, the goal is to increase the number of successfully completed tasks within a predefined timeframe (the fulfillment of priorities). A priority is considered fulfilled if the processing time for a given task is less than or equal to a specified time limit for each task. Consequently, this chosen set of characteristics represents the most relevant and suitable features to train the model for optimal priority fulfillment. Detailed explanations of the remaining variables are provided in the subsequent paragraphs.

Operation (O_t): This term specifies the arithmetic operation required for a given task as defined by the client. Given the diverse resource capabilities, the AI agent must select the most appropriate resource for executing the task. For the sake of simplicity in this study, the focus is on two distinct operations, though it is important to note that the types and numbers of operations in practical scenarios

can vary. These operations are used here to demonstrate the proof of concept. The first operation is the computation of the factorial of the transmitted data (n), defined by the formula $n! = \prod_{k=1}^n k$. The second operation involves exponentiation, represented by $n^m = \prod_{k=1}^m n$.

Data (D_t): This term refers to the information that requires processing by available resources. It is also used by NN to estimate the duration of task processing, which allows for the assessment of prioritization efficacy. In this study, the data values are assumed to be integers within the range of 1 to 10,000.

Priority (P_t): This term refers to the level of importance assigned to a given task. The assignment of priority is random and does not depend on the size of the task. In this study, we define three distinct priority levels: '1' denotes the highest priority, '2' indicates medium priority, and '3' signifies the lowest priority. It is important to note that these designations are arbitrary and do not influence the proof of concept demonstrated herein. Additionally, the priority given to a task is inherently connected to the 'priority time'.

Priority Time (Pt_t): This is a measure of the time duration allocated to a resource to complete a task with a specified priority level. Should this duration be exceeded, the corresponding priority is considered unmet. In this study, durations of 2, 5, and 8 seconds were assigned to priorities 1, 2, and 3, respectively.

Usage (U): This term refers to the extent to which a resource is utilized. To ensure optimal usage, an AI agent must evaluate the utilization of each network resource before task allocation. Initially, we estimated CPU utilization as a percentage by employing the Psutil library's 'cpu_percent()' function [210]. However, this metric does not invariably reflect the actual workload of a resource. To circumvent this limitation, we devised a custom implementation that monitors the number of tasks currently being processed by the resource. The AI agent gathers this data and conveys it to the appropriate model, thereby facilitating more precise and efficient resource allocation.

Power (P_w): This term refers to the power associated with a given resource that is transferred to the AI agent. To ensure consistency during test runs, the power setting must be established prior to the start of testing and remain constant throughout. The Linux kernel's control groups (cgroups) are employed to manage power effectively. Specifically, the CPU subsystem uses 'cpu.cfs_period_us' to define the frequency at which a CPU resource can be accessed, while 'cpu.cfs_quota_us' determines the maximum amount of time that a task is permitted to run in a given period. For example, a 'cpu.cfs_period_us' value of 1000000 coupled with a 'cpu.cfs_quota_us' value of 700000 implies that a task may run for a maximum of 0.7 seconds within any 1-second period, which correlates to a performance rate of 0.7 in our implementation.

Response Time (R_t): This is an important parameter in evaluating network performance. It facilitates the simulation of network latency by introducing a predefined delay prior to the commencement of task processing. Specifically, an R_t value of 2 denotes that the resource will wait for two seconds before starting to process each task. It is important to note that the settings for each resource can be individually configured prior to a test run and will remain constant throughout the testing period. Moreover, it should be emphasized that the defined delay does not necessarily represent the actual latency of the network but should serve merely as a proof of concept in a test environment. This helps demonstrate how the system might accommodate network latency in making the most appropriate decisions under varying network conditions.

3.3.2 Action Space

Considering the input features delineated previously, which contribute to the derivation of the current state, a suitable action may be elected from the action space at hand. The action space, S_a , contemplated in this inquiry is crafted to fulfill the goal of resource allocation and thus encompasses an array of selectable resources. The action space is formally depicted as $S_a = \{Res_1, Res_2, \dots, Res_k\}$.

3.3.3 Network Setup

To establish a suitable environment for AI agents, we designed and implemented a network, as depicted in Figure 3.3. This network is composed of clients capable of transmitting tasks and resources allocated for processing these tasks. This approach enables AI agents to collect real data, which can be used for both training and evaluation purposes. Within this network, several devices run various programs: clients execute a program that facilitates the creation of tasks and maintain a switch-based connection to the AI agent. The AI agent operates a NN-based resource allocation application and connects to the resources via switches, acting as a proxy for the clients. Clients are unaware of the resource locations and must contact the AI agent for directions. The resources, virtualized for these experiments, are self-contained, encapsulated applications, each capable of processing a specific task. Each resource was assigned specific properties, such as power and usage, as mentioned in section 3.3.1. An interface links both clients and resources, used to configure network devices or send start signals to them. The network routing is managed by the RYU controller.

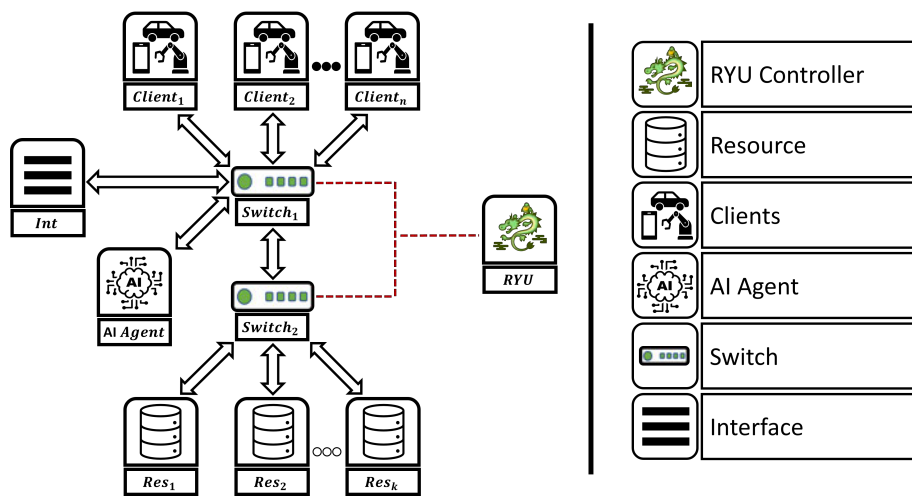


Figure 3.3: Schematic of Network Architecture and Connectivity Setup

3.4 AI-Models

Two distinct models were developed, differing in their design but sharing fundamental principles. The first model employs a classification approach, while the second model adopts a regression approach. Both models utilized a fully

connected feedforward Neural Network, implemented using TensorFlow. Before introducing the two models, a brief background on the different techniques is provided.

- **Fully connected feedforward neural networks:** Also known as multilayer perceptrons (MLPs), are a class of artificial neural networks where all neurons in one layer are connected to all neurons in the next layer. There are no connections within the same layer or backward connections. These networks consist of an input layer, one or more hidden layers, and an output layer. Each connection between neurons carries a weight, and each neuron typically applies a non-linear activation function to its weighted input. Feedforward neural networks process information in a forward direction, from the input layer through the hidden layers to the output layer, making them well-suited for a wide range of tasks, including classification, regression, and pattern recognition [211].
- **Classification:** In reinforcement learning, the classification approach involves categorizing the possible actions or decisions an agent can take in a given state into classes. The goal is to learn a policy that maps states to actions by identifying which action (or class of actions) leads to the most favorable outcome, based on the rewards received from the environment. This approach contrasts with value-based methods, where the focus is on estimating the value of each action in a state. In classification-based reinforcement learning, the agent uses experience to learn which actions are best in different situations, essentially classifying states into action categories to maximize the cumulative reward [212].
- **Regression:** The regression approach is a statistical methodology used for modeling and analyzing the relationships between a dependent variable and one or more independent variables. The primary purpose of regression is to predict the value of the dependent variable based on the values of the independent variables. This is accomplished by estimating the coefficients of the regression equation that minimizes the difference between the actual and predicted values of the dependent variable. Regression analysis can be

used for forecasting, estimating outcomes, and determining the strength and nature of relationships between variables. Common types of regression include linear regression, where the relationship is modeled as a straight line, and nonlinear regression, where the relationship is modeled using more complex equations [213].

- **TensorFlow:** Is an open-source software library for dataflow and differentiable programming across a range of tasks. It's designed for machine learning and deep learning applications, providing a comprehensive, flexible ecosystem of tools, libraries, and community resources that allows researchers to push the state-of-the-art in ML, and developers to easily build and deploy ML-powered applications. TensorFlow was developed by the Google Brain team and is used for both research and production at Google. It supports computations on multiple CPUs or GPUs, as well as mobile and edge devices, making it a versatile library for developing and training ML models.

3.4.1 Classifier Model

Architecture: The proposed model introduces a classification approach that integrates input features from a single task and multiple resources. This model comprises multiple output neurons, each corresponding to an input resource. These output neurons are responsible for calculating the probability that indicates the most suitable choice, based on the input features. Figure 3.4 illustrates the model's architecture, which is structured into three layers. The topmost layer is the input layer, responsible for receiving the earlier mentioned input features. Specifically, the first four neurons, highlighted in orange, process the client's input features: O_t , D_t , P_t , and the time request of the priority (Pt_t). Each of these features is assigned to an individual neuron. The neurons colored green are designated for the input features of the resources. Each resource is allocated three neurons, corresponding to utilization (U), performance (Pw), and (Rt). For enhanced clarity, these resource input features are collectively labeled as 'resource features' in Figure 3.4. The middle layer, depicted in blue, is the Hidden Layer. Its composition in terms of the number of neurons and layers is flexible,

governed by hyperparameters. These hyperparameters are predefined before training and can be adjusted in subsequent sessions. The section on training will offer more details regarding the selection and adjustment of hyperparameters.

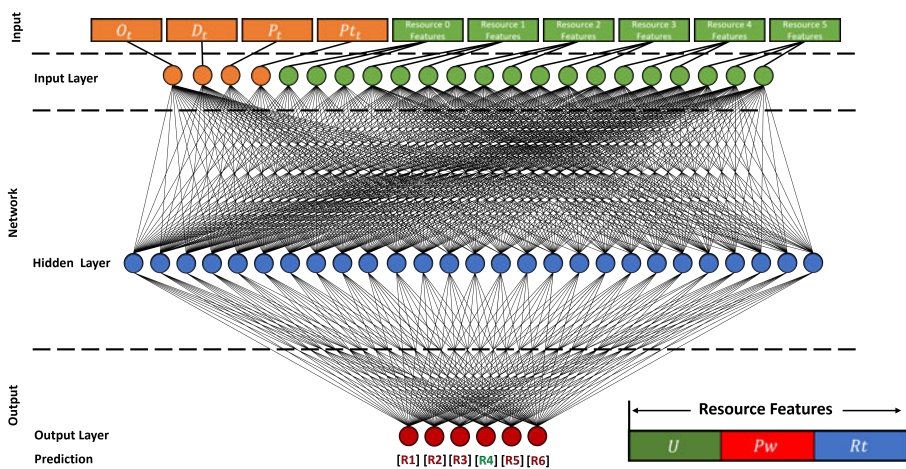


Figure 3.4: Structure of the classification model, NN architecture and input/output layers

Functionality: Before training, it is essential to establish the model’s architecture, including the number of layers and neurons. Initially, we designed the model to accommodate six resources, making the integration of a seventh resource challenging without architectural modifications. Each resource is represented by three input neurons and one output neuron. Attempting to introduce a seventh resource without adjusting the architecture would result in an error, as there would not be enough neurons to process the additional data. Conversely, if fewer than six resources are used, the model encounters issues as all neurons require input. To address this, placeholders are used when fewer than six resources are available. These placeholders are defined with fixed features: $Pw = 0$, $Rt = 1000$, and $U = 1000$. These deliberately suboptimal features are ignored by the model, which instead focuses on the actual resources. In scenarios with more than six resources, such as nine, the model is invoked multiple times, each time with the same task information. In the first invocation with six resources, the model operates as usual. The second invocation incorporates the three additional resources, along with the best resource selected from the first call, and two placeholder resources. The four actual resources then compete, with one being selected for task processing. This strategy allows for the utilization of the same

model, differing only in the frequency of invocation. Figure 3.5 illustrates the model’s prediction flow with nine resources. Should more resources be available, this process is repeated, allowing the optimal outcome from previous rounds to compete against the new resources, until all have been evaluated.

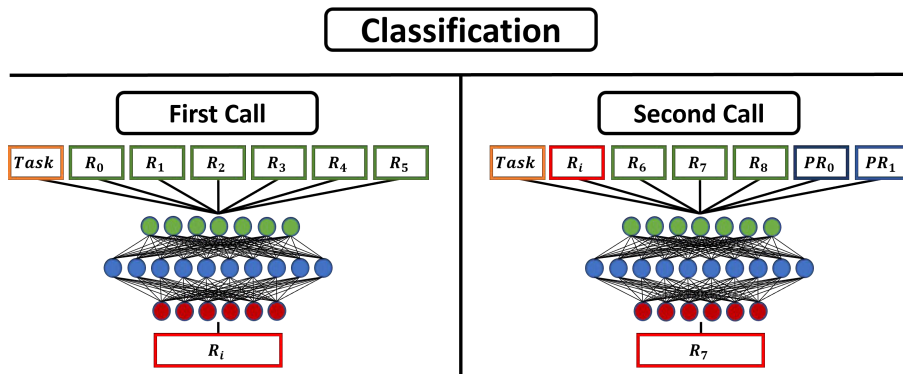


Figure 3.5: Illustration of the classification model workflow utilizing nine resources

3.4.2 Regression Model

Architecture: The model presented herein, akin to the classification model, utilizes input features from the client. However, it diverges in that it processes input features from only a single resource at any given time, rather than handling multiple resources simultaneously. The output is determined by a single neuron responsible for selecting the appropriate resource. This model generates a score that assesses the performance of the input features. The architecture of the model comprises three layers: input, network, and output, as illustrated in Figure 3.6. The input layer displays various input features, with those from the client shown in orange and features from resources in green. However, features from only one resource are input into the model at a time, consisting of four values: resource operation (O_r), utilization (U), performance (Pw), and response time (Rt). Unlike the classification model, the regression model also needs to recognize the supported operation (O_r) by the currently considered resource. This is because the regression model evaluates the resources one at a time, in an arbitrary order, making it unable to know the operations supported by different resources beforehand. In contrast, the classification model is set up in a way that specific resources are assigned to specific neurons. This setup allows the model to learn the operations supported by different resources over training iterations

and through the reward received. Since features from only one resource are input into the model at a time, the network architecture requires a fewer number of neurons and connections. Adding a new feature for a resource only requires the inclusion of a single neuron, as opposed to the classification model, which would need six input neurons and possibly several additional neurons in the hidden layer. The output layer consists of a single neuron that is typical for regression models and generates a score reflecting the quality of the input features.

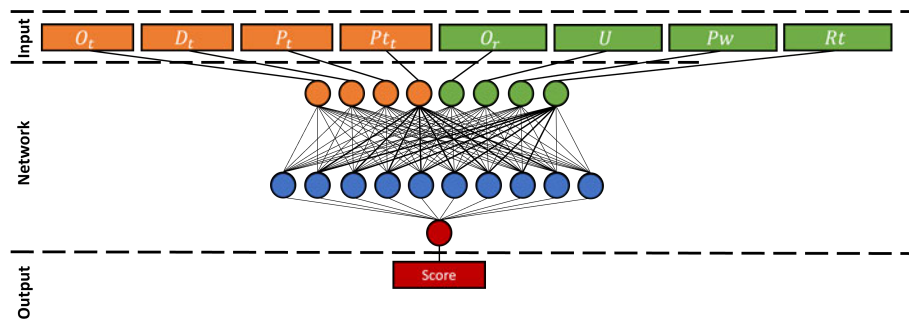


Figure 3.6: Structure of the regression model, NN architecture and input/output layers

Functionality: The regression model described in this study is constructed to analyze one resource at a time. When multiple resources are employed, the model is invoked for each resource individually, with the corresponding features being provided for each instance. In contrast to the classification model, this method does not necessitate the use of any placeholder resources. However, it requires the more frequent updating of client-specific features. The scores from the individual models are aggregated using a *max* function, which selects the highest value as the ultimate decision, denoted by $\max(rs_1, rs_2, \dots, rs_k)$. The application of the regression model to nine resources is depicted in Figure 3.7.

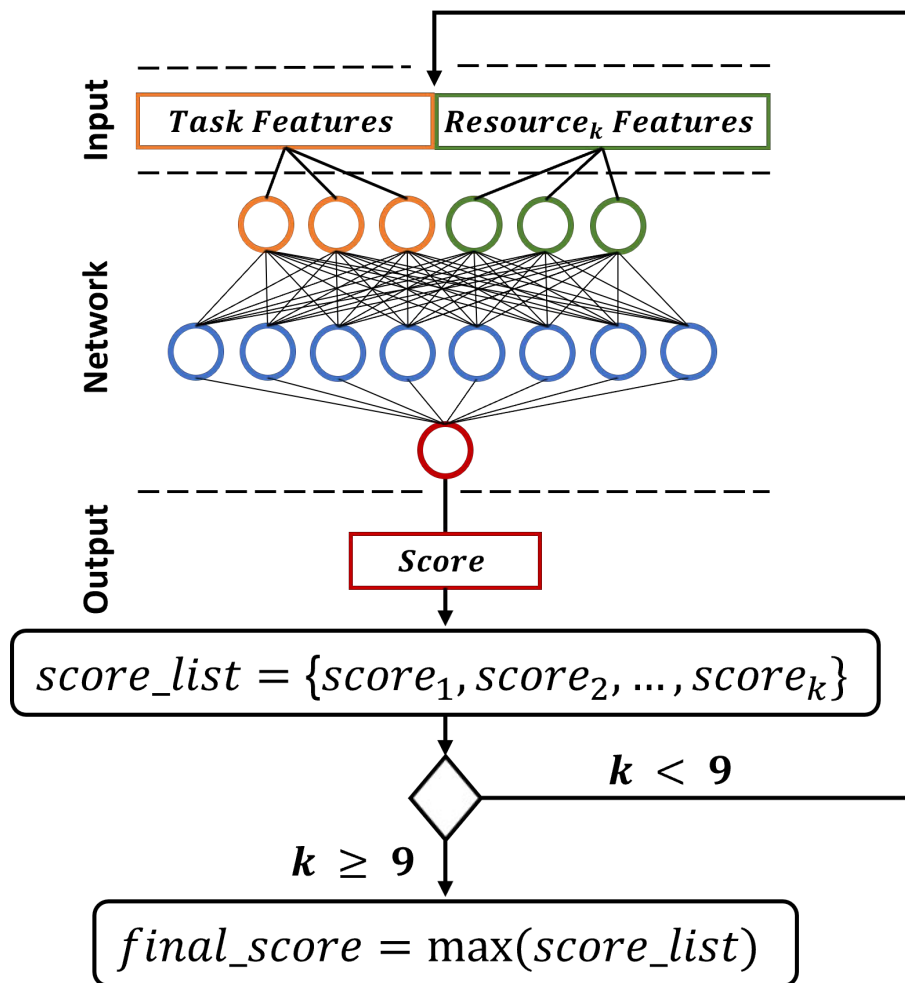


Figure 3.7: Illustration of the regression model workflow utilizing nine resources

3.5 Primary Metrics

In this study, various metrics were used to assess the efficacy of the models. In addition to conventional metrics like loss, which are commonly used during model training, we developed and utilized customized metrics such as reward functions and the fulfillment of priorities. To ensure the findings are easily comprehensible and reproducible, this section provides a detailed explanation of these metrics.

3.5.1 Reward

Rewards serve as a critical measure for evaluating the training process and overall performance. The magnitude of the reward is determined by the efficacy of an action. Actions with better outcomes receive higher rewards, while undesirable

actions may result in negative rewards or penalties. The value and calculation of rewards are flexible and can be customized as needed. Consequently, we have designed and implemented three reward functions. This section presents an overview of the design and implementation of these functions, including their structure and key concepts. We also discuss the impact of these functions on the training process and final evaluation.

Reward A: The reward function was developed to achieve a compact and simple design. In instances where a task is assigned to a resource incapable of performing the required operation, the resource responds with a value of 0, signifying a failed computation and incurring a penalty of -5 . Conversely, successful completion of a task within the specified time (Ct) and meeting a priority yields a reward of 3. Failure to meet a priority results in a penalty of -3 .

$$r_a = \begin{cases} -5 & result = 0 \\ 3 & Ct \leq Pt \\ -3 & Ct > Pt \end{cases}$$

Reward B: Offers more refined granularity than its predecessor, Reward A. It accounts for the speed at which a priority is fulfilled, thereby providing a more comprehensive assessment of task completion quality. In Reward B, a -5 penalty is applied when a task is sent to a resource incapable of performing the required operation. However, the reward is not solely determined by task fulfillment but also by the quality of completion. Specifically, the reward is proportional to the task completion speed. A task completed in 50% or less of the allotted time receives the highest reward. If completion time falls between 50% and 75%, the reward is 2 points, reducing to 1 point for times between 75% and 100%. Penalties, similarly, are assigned based on the severity of misdemeanors. A minor misdemeanor, nearly exceeding the time limit, incurs a -1 penalty, while misdemeanors exceeding the allotted time by more than 200% receive a more substantial -3 penalty.

$$r_b = \begin{cases} -5 & result = 0 \\ 3 & Ct \leq Pt \times 0.5 \\ 2 & Pt \times 0.5 < Ct \leq Pt \times 0.75 \\ 1 & Pt \times 0.75 < Ct \leq Pt \\ -3 & Ct > Pt \times 2 \\ -2 & Pt \times 1.5 < Ct \leq Pt \times 2 \\ -1 & else \end{cases}$$

Reward C: Has a structure and granulation similar to Reward B but differs in reward distribution. Instead of offering a high reward for the best times, Reward C provides better rewards for processing times that are closer to the priority time. Specifically, the maximum reward, among three tiers, is attained for times between 75% and 100%. Conversely, the lowest reward tier applies to times at 50% or below. The penalty structure remains unchanged from Reward B. The primary objective of this reward function is to train the network to select resources that are adequately sufficient for a task, rather than consistently opting for the best available resources. This approach is intended to prevent the network from unnecessarily claiming resources that may be required for more demanding tasks.

$$r_c = \begin{cases} -5 & result = 0 \\ 1 & Ct \leq Pt \times 0.5 \\ 2 & Pt \times 0.5 < Ct \leq Pt \times 0.75 \\ 3 & Pt \times 0.75 < Ct \leq Pt \\ -2 & Pt \times 1.5 < Ct \leq Pt \times 2 \\ -3 & Ct > Pt \times 2 \\ -1 & else \end{cases}$$

3.5.2 Loss

Loss is a key metric for evaluating the accuracy of a model's predictions. Each time the training function is called, it processes a set of records that include both the inputs and the desired outcomes. The model processes these inputs, generates its outputs, and then compares these outputs with the desired outcomes. The difference between the model's output and the desired outcome is known as the loss. This loss value is calculated and recorded with every training iteration, allowing for monitoring of the model's progress throughout the training process. Ideally, the loss should decrease rapidly at the beginning of training and continue to decline steadily. A slow decrease in loss might indicate a low learning rate, leading to unnecessarily prolonged training durations. Conversely, if the loss oscillates around a certain value without further reduction, it could suggest either an inadequate model architecture or a high learning rate that causes excessively large update steps. For effective decision-making, the loss needs to reach a sufficiently low value. Monitoring the loss value during training enables the process to be halted when further reductions become unlikely.

3.5.3 Priority Fulfilled

This metric evaluates the performance of a trained model. To this end, the model is deployed in a production environment and tasked with processing a predefined number of tasks. Performance is measured by the frequency with which the model successfully meets the assigned priorities. The metric is calculated as $P_{result} = P_{fulfilled} \cdot 100 / task_{total}$, where P_{result} is the percentage of fulfilled priorities, $P_{fulfilled}$ is the number of priorities successfully met, and $task_{total}$ is the total number of tasks. A priority is considered fulfilled if the processing time for a given task ($task_{proc}$) is less than or equal to the assigned priority time (Pt_t). Thus, $P_{fulfilled}$ is the count of tasks for which $task_{proc} \leq Pt_t$. This metric focuses only on the time during which a task actively utilizes resources, reflecting the period the model can influence. The emphasis on resource processing time is justified by the model's role in selecting resources for task delegation, acknowledging that different resources may have varying processing times.

3.6 The experimental setup

The experimental setup depicted is utilized for both training and evaluating our proposed method.

Table 3.1: System Configuration

Hardware			
graphic card	Nvidia RTX 2080 TI, 11GB GDDR6		
processor	AMD Ryzen 7 1800x, 8 CPU Kerne, 3,6 GHz		
RAM	G.Skill Trident Z DDR4 32 GB (4x8)		

Software	Version	Software	Version
Tensorflow	2.4.0	Numpy	1.19.4
Mininet	2.3.0d6	Pandas	1.0.5
Ryu Controller	4.34	Dash	1.13.4
Flask	1.1.2	Plotly	5.1.0
Flask-Restful	0.3.8		

3.7 Training

To equip an AI agent with the capability to perform resource allocation based on logical decisions, it is essential to train the NN beforehand. Supervised learning is not feasible in this context, as it requires the availability of labeled data. However, pre-determining the optimal resource choice based on input features is not possible. To overcome this challenge, we developed a reinforcement learning system. Figure 3.8 illustrates the training process flow, which involves three key components: an agent, an environment, and a buffer. During training, the agent’s task involves taking an action based on a given state, receiving a reward, and subsequently improving. The environment provides the agent with a state and determines a reward contingent on the action executed. The buffer stores all pertinent information, furnishing the agent with a series of experiences throughout a training session. The training process commences with the environment presenting a state to the agent, which is then distributed among the respective neurons. Each neuron processes the requisite element, enabling the agent to make a decision based on a policy. This decision is relayed back to the environment. Following this, the environment calculates the reward based on the action and its outcome,

transmitting this information to the buffer. The buffer compiles all data into an experience, denoted as $Exp_n = \{state, action, reward\}$, but does not immediately transfer it to the agent. Instead, the agent completes a predetermined number of tasks while continuing to populate the buffer with experiences. Upon reaching this predefined number, the agent initiates a training session, selects a set number of random experiences, and uses these to train the NN. The policy is refined through this training, leading to more informed subsequent actions by the agent. This cycle is repeated continuously, resulting in progressive enhancements to the policy with each training iteration.

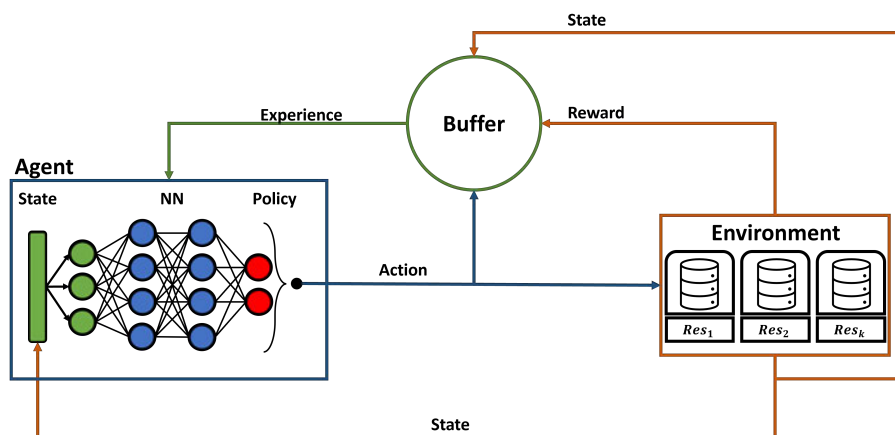


Figure 3.8: Workflow for training the AI agent in the proposed system

3.7.1 Training setup

To optimize the models, a test setup was designed to assess the impact of various parameters on the results and to identify the optimal parameters for the specific use case. The best-performing models were then employed to evaluate the dynamic behavior. The initial part of this section outlines the test setup and procedure, while the subsequent part presents the characteristics of the training and testing datasets.

Dataset Structure: To ensure the reliability and generalizability of neural network models, it is widely accepted practice to maintain separate training and test datasets [214]. Test sets provide data not previously encountered by the model, enabling verification that the network has accurately learned underlying patterns, rather than merely memorizing the training data. In this study, we utilized three training datasets, each containing 20,000 tasks, and three test datasets, each

comprising 2,000 tasks. Each training iteration varied due to the initialization of the model’s weights, leading to a gradual convergence towards the final result. Employing multiple training sets of varying complexities enhanced the accuracy and robustness of the model’s performance. Table 3.2 summarizes the properties of each dataset, including the number of tasks with different priorities in each dataset.

Setname	Prio 1	Prio 2	Prio 3	Value range
train set 1	6580	6789	6631	1–10000
train set 2	6706	6687	6607	1–10000
train set 3	6740	6669	6591	1–10000
test set 1	657	645	698	1–10000
test set 2	833	775	392	2000–10000
test set 3	1007	802	191	4000–10000

Table 3.2: Task Prioritization and Value Range Assignment Overview

Test procedure: As previously mentioned, three reward functions are utilized in both classification and regression models, resulting in six models that require training and evaluation. Additionally, each model has a unique set of hyperparameters that must be specifically selected and evaluated. To ensure consistency in the experimental procedure, the approach presented in Figure 3.9 is employed. The first step involves hyperparameter tuning, where default values are initially assigned to all hyperparameters. One hyperparameter is selected and adjusted while keeping the other parameters constant. The resulting model is trained on the three designated training sets, generating three models based on the respective training sets. Each model is then tested against the three designated test sets to determine the optimal model. Training and testing results are documented, and the process is repeated with the same hyperparameters but varying values. The most promising values are selected after sufficient evaluations are conducted.

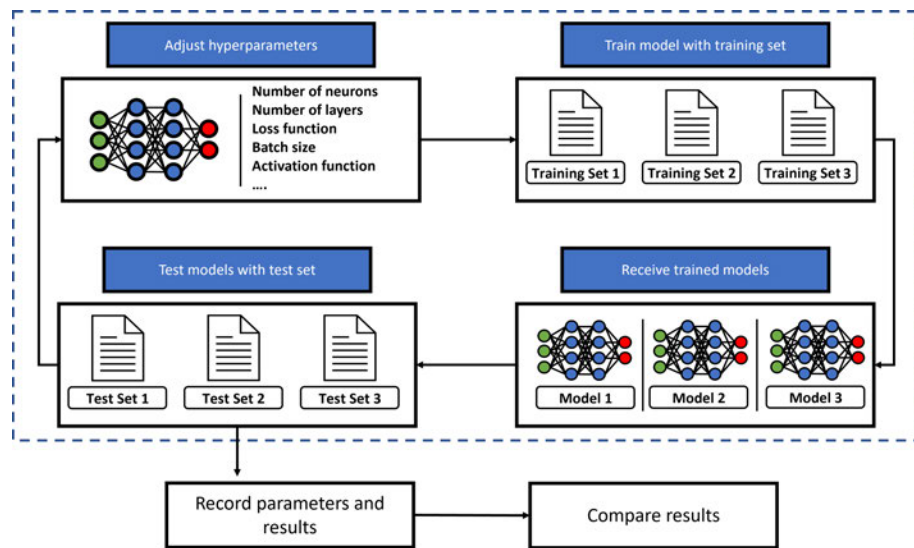


Figure 3.9: Hyperparameter Tuning Process: A Flowchart Illustrating the Steps for Determining Optimal Hyperparameters

3.7.2 Fine-Tuning Model Parameters

This section presents the methods employed and the various hyperparameters that were fine-tuned to enhance the performance of each model. The effect of the modifications on the models is reported here for better clarity and understanding.

Normalization: The process of normalization involves transforming features to maintain them within a similar range of values. This results in improved model performance and training stability [215]. In the initial tests, it was observed that satisfactory rewards could not be achieved, regardless of the number of hidden neurons used. In the regression model, positive rewards could not be obtained without normalization. Analysis of the inputs revealed that all features, except the data input feature, fell within a similar value range (see Table 3.3).

Input feature	Value range
operation	1,2
prio	1,2,3
prio_time	2,4,8
res_operation	1,2,3
usage	0,1,2,3,...,40
power	0.3,0.5,0.8,1.0
resp_time	0.5,1.0
data	1,2,...,10000

Table 3.3: Input Features and Corresponding Value Ranges

However, the size of the data input feature caused it to receive a higher priority when multiplied with weights, leading to insufficient weighting of other features. To address this issue, a normalization function:

$$data_{normalized} = \frac{data - data_{min}}{data_{max} - data_{min}}$$

was applied, which mapped the data value range to (0, 0.0001, 0.0002, ..., 1) based on known value ranges in the network. The smallest and largest values corresponded to 1 and 10000, respectively. The same equation was also applied to (*usage*), but it did not affect the results and was therefore discarded. Two graphs were generated to demonstrate the effects of normalization on training, as shown in figure 3.10. The classification and regression models were initially trained using different numbers of hidden neurons and then retrained with normalization, with other parameters unchanged to solely display the effects of normalization. The graph shows the average reward (left y-axis) and average loss (right y-axis) achieved before (dashed line) and after (solid line) normalization. Tests were performed using different models to show the effects on different model architectures. The classification model showed a decrease in loss by 57% on average, with loss ranging from 0.5256 to 0.2944 after the normalization. Rewards, except for the model with zero hidden neurons, improved by an average of 130%, with

negative rewards shifted to a positive range. A similar pattern emerged for the regression model, with an average decrease in loss by 59% and improvement from having value ranges between 2.9 and 27.6 to having value ranges between 1.1 and 13.58. Normalization significantly improved rewards in the regression model by an average of 211%, making it possible to obtain positive rewards. It is important to note that useful results in the regression model require more than three hidden neurons.

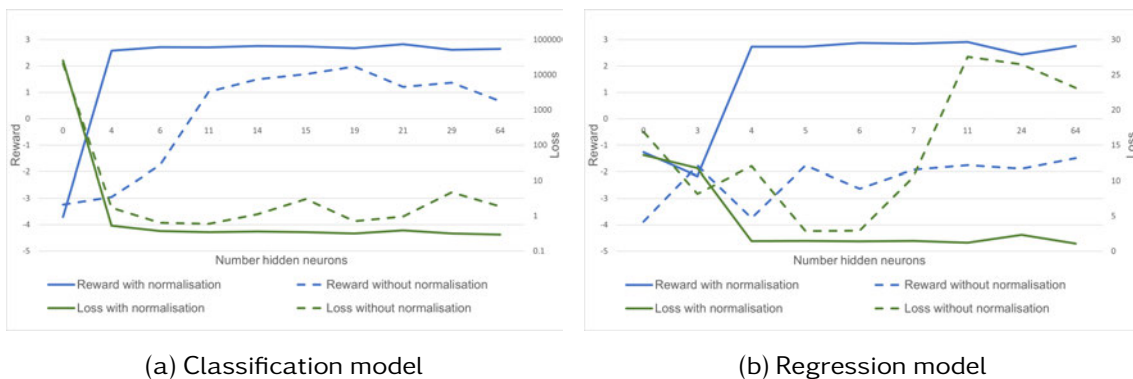


Figure 3.10: Impact of Normalization on Reward and Loss in Model Training

Hidden Neurons: This section presents an overview of the impact of hidden neurons (N_h) on reward and the fulfillment percentage of priorities. The determination of the optimal number of hidden neurons in the hidden layer of a model is a time-consuming task. The sizes of the input and output layers are determined by the input features and the desired output, while the size of the hidden layer is flexible and varies according to the problem being addressed. Although there is no definitive formula for determining the optimal number of hidden neurons, various methods have been proposed, as indicated in Table 3.4, to provide guidance. In this work, we applied each formula to determine the optimal number of neurons for classification (N_c) and regression (N_r) models. The Rule of Thumb method and Prototyping formula yielded satisfactory results in our study.

Method	Equation	N_c	N_r
Sheela, Deepa [216]	$N_h = \frac{4N_i^2+3}{N_i^2-8}$	4	5
Li et al. [217]	$N_h = \frac{\sqrt{1+8N_i}-1}{2}$	6	4
Tamura, Tateishi [218]	$N_h = N_i - 1$	21	7
Xu, Chen [219]	$N_h = C\left(\frac{N_i}{N_i \log N_i}\right)^{\frac{1}{2}}$	15	24
Rule of Thumb	$N_h = (N_i + N_o) * \frac{2}{3}$	19	6
Shibata, Ikeda [220]	$N_h = \sqrt{N_i * N_o}$	11	3
Hunter et al. [221]	$N_h = \log_2(N_i + 1) - N_o$	-	3
Comp. Int. [222]	$N_h = \frac{N_i+N_o}{2}$	14	5
Luis Serrano [223]	$N_h = 2^x$	2^4-2^{13}	2^4-2^{13}
Prototyping	$N_h = N_i * 130\%$	29	11

Table 3.4: Overview of Formulas for Determining the Number of Hidden Neurons in Neural Networks

This thesis not only compares classification and regression models with varying numbers of hidden neurons but also investigates the effects of reward functions A through C to showcase their influence (refer to figure 3.11).



(a) Reward with classification model

(b) Fulfilled priorities with classification model

Figure 3.11: Relationship between the Number of Neurons, Reward, and Percentage of Fulfilled Priorities in the Classification Model

The figures presented in this study provide an overview of the average reward achieved across all test sets for each reward function (shown in the left figure), as well as the average percentage of priorities fulfilled (depicted in the right figure).

The horizontal axis in both figures represents the number of neurons tested, while the colors yellow, blue, and green correspond to reward functions A, B, and C, respectively. The vertical axis in the left figure represents the reward score, ranging from -4 to 3 , whereas the right figure displays the percentage of fulfilled priorities, ranging from 0% to 100% . The findings indicate that the inclusion of at least one hidden layer is crucial for obtaining satisfactory results. With four neurons, Reward B exhibited positive outcomes with a 69.09% satisfiability rate and a reward score of 1.14 . Similarly, Reward C achieved a 69.5% satisfiability rate, albeit with a lower reward score of 0.5 . However, Reward A did not yield satisfactory results, displaying only a 27.73% satisfiability rate and a negative reward score of -1.85 . When the number of neurons was increased to six, the performance of Reward B and C slightly improved, while Reward A started to show promising results for the first time, with a 77.87% satisfiability rate and a positive reward score of 1.72 . Beyond fifteen neurons, all reward functions demonstrated favorable outcomes, and additional neurons did not necessarily contribute to improved results. Surprisingly, the results obtained with 8192 neurons were comparable to those achieved with only fourteen neurons, but the architecture becomes more complex with a larger neuron count.



Figure 3.12: Relationship between the Number of Neurons, Reward, and Percentage of Fulfilled Priorities in the regression Model

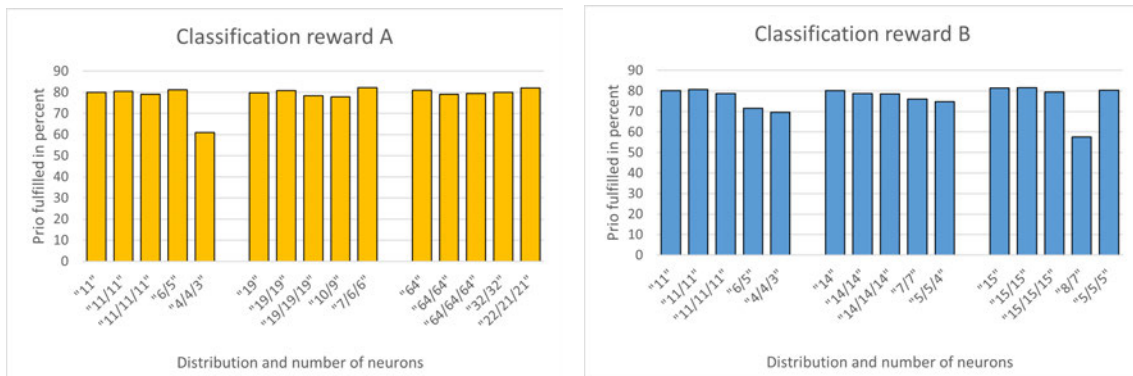
Figure 3.12 illustrates the test results obtained from the regression model, indicating that a minimum of three neurons is necessary to achieve satisfactory priority satisfiability. Among the reward functions, Reward Function B exhibits the poorest performance in this regard, falling below the 70% threshold. Similarly, the re-

sults obtained from five neurons exhibit comparable behavior to the classification model, with values falling within a relatively similar range. A comparison between the classification and regression models is presented in Table 3.5, which summarizes the results of separate tests conducted for the respective reward functions. The values obtained are comparable, with the regression model demonstrating slightly superior performance in most cases. Notably, the classification model attains the highest values, reaching 87.5% for Reward B.

	Class. A	Reg. A	Class. B	Reg. B	Class. C	Reg. C
Test 1	84.73	86.43	87.16	86.53	86.29	86.25
Test 2	73.23	77.04	74.97	75.44	74.58	75.95
Test 3	59.38	64.24	60.66	62.05	59.44	61.72
Average	72.11	75.90	74.26	74.67	73.44	74.64

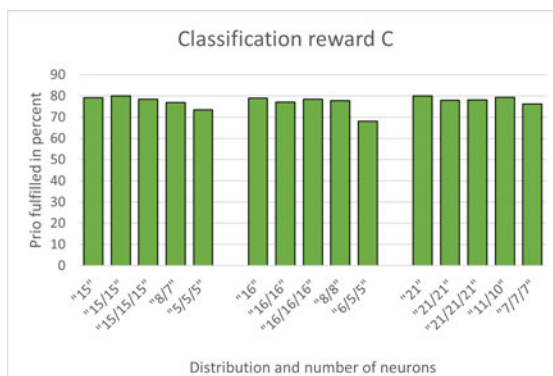
Table 3.5: Percentage of Priority Fulfillment for Classification and Regression Tests

Hidden Layers: In addition to considering the number of hidden neurons, the arrangement of neurons across multiple layers can also be a crucial factor. As mentioned previously, it is generally recommended to have between one and three hidden layers. To streamline the testing process, this section of the study adheres to this guideline and focuses on evaluating a specific number of arrangements. From each model and reward function, the top three arrangements are selected, resulting in a total of nine arrangements for classification and nine for regression tasks. This approach aims to further enhance the already achieved results. The classification model results are illustrated in figure 3.13.



(a) Classification model with reward A

(b) Classification model with reward B

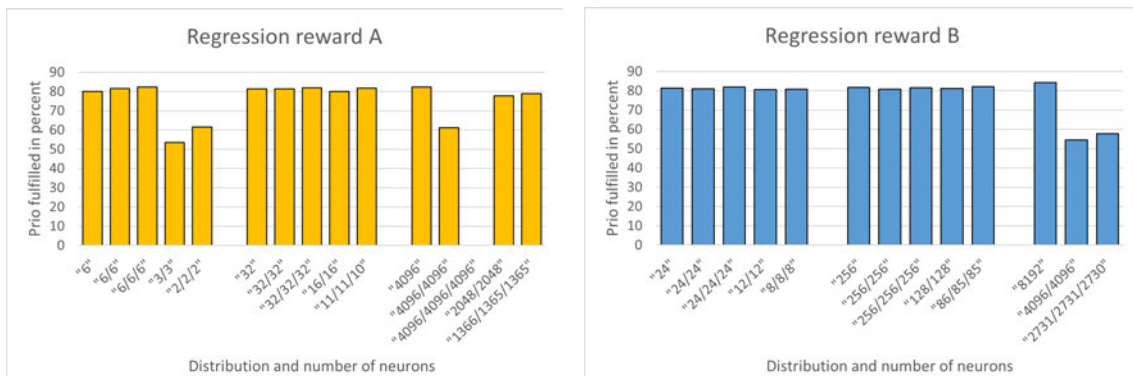


(c) Classification model with reward C

Figure 3.13: Impact of Neuron Distribution on Priority Fulfillment in Classification Models

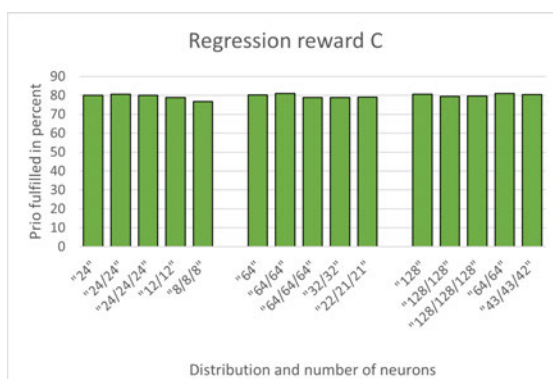
All figures, except for the color of the columns, depict the various reward functions and models using the same color scheme as in the hidden neurons section. The graphs illustrate the percentage of priority satisfaction on the vertical axis and the number and arrangement of tested neurons on the horizontal axis. A single number, such as „11“ represents a hidden layer with eleven neurons, while „6/5“ signifies six neurons in the first hidden layer and five neurons in the second layer. The combination „4/4/3“ indicates three hidden layers with four neurons in the first and second layers and three neurons in the last layer. Each graph is divided into three groups, each containing the best-selected neurons from the previous section. These neurons are then tested in different combinations, employing the same approach for each experiment. In the first test, the same number of neurons is transferred to multiple layers, for example, „11“ becomes „11/11“. In the second experiment, the neurons are distributed across multiple layers, while maintaining the total number of neurons. For instance, „11“ is

divided into „6/5“. Figure 3.13 presents the results for the classification models. The values fall within a similar range, and it is not possible to conclude directly that a higher number of layers leads to better results. However, observations indicate that if the number of neurons is already in a low range, further spreading them across multiple layers may result in poorer outcomes. This is evident in the middle figure (b), where distributing the „11“ neurons among „6/5“ or „4/4/3“ leads to a decrease in satisfiability from approximately 80% to 70%. The extent of this decrease may vary, as seen in the results for „15“ neurons. When distributed among „8/7“, the result drops from 81.4% to 57.53%, but remains almost the same (80.31%) when distributed among „5/5/5“. Similar trends are observed for the reward function C in the right figure (c). However, exceptions are observed in the left figure (a) with reward function A, where the outcome declines from 79.89% to 60.98% when switching from „11“ neurons to a „4/4/3“ distribution. Conversely, it increases to 81.12% for a „6/5“ distribution and to 82.20% and 82.04% for „7/6/6“ and „22/21/21“ distributions, respectively. The results for the regression model, along with the corresponding reward functions, are summarized and displayed in figure 3.14.



(a) Regression model with reward A

(b) Regression model with reward B



(c) Regression model with reward C

Figure 3.14: Impact of Neuron Distribution on Priority Fulfillment in Regression Models

The presentation structure for the regression models follows a similar approach to that of the classification models. Many of the previous observations can also be applied to these models. However, unlike the classification models, we conducted tests with higher numbers of neurons for the regression models. This can be seen in figure (a) on the left, which represents the results for reward function A, and figure (b) in the middle, which represents the results for reward function B. In the left figure, the combination „4096/4096/4096“ on the right side presented difficulties in the current implementation, leading to missing results due to the extensive training time required for a model of this size. Consequently, this test had to be canceled. Similarly, in the middle figure (b), the combinations „8192/8192“ and „8192/8192/8192“ were not tested, as it was assumed that they would yield similar results. Furthermore, the results demonstrate that a higher number of neurons does not always result in better outcomes. This can be observed in the left figure (a), where the models with „4096“ neurons

and „6/6/6“ distribution yielded nearly identical values of 82.33% and 82.36%, respectively. As we move to a higher range of neurons, the results essentially converge. This can be seen in the right figure (c) for the reward function C.

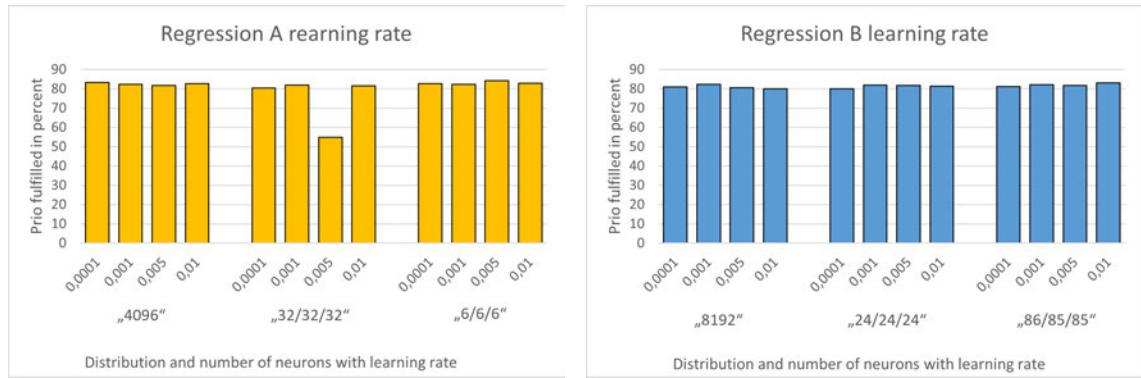
Learning Rate: This section explores the influence of the learning rate, the last of the three hyperparameters examined in this study. Learning rate controls the size of the steps the algorithm takes in updating the model’s parameters during training. It determines how much the weights in the network are adjusted in response to the estimated error each time the model weights are updated. Choosing a proper learning rate is crucial as it affects the speed and quality of the learning process: too small a rate can lead to a very slow convergence, while too large a rate can cause the training process to oscillate or even diverge, preventing the model from learning effectively. This study explores nine combinations per model, selected from previous tests that have yielded positive results. The learning rates of 0.0001, 0.005, and 0.01 are tested to assess the impact of smaller or larger weight adjustments on the results. Additionally, the default learning rate used in previous tests (0.001) is included for comparison purposes. Three graphs are presented, corresponding to specific models. Figure 3.15 illustrates the outcomes of the classification model, with each diagram representing one of the reward functions (A, B, or C) utilized in the respective test. These reward functions are assigned the colors yellow, blue, and green. Each diagram showcases the evaluation of three neuron combinations with four different learning rates.



Figure 3.15: Impact of Learning Rate on Priority Fulfillment in Classification Models

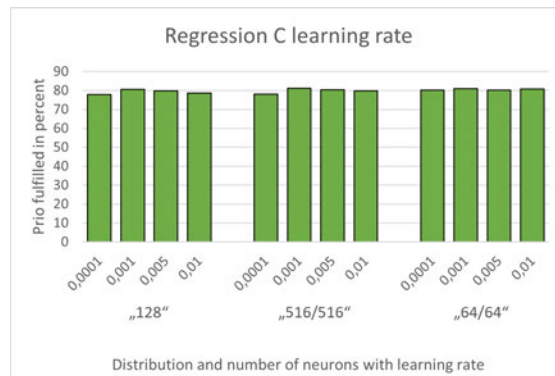
The neuron combination is presented on the x-axis, using the same representation as in the previous section (e.g., „7/6/6“ indicates seven hidden neurons in the first layer, six in the second, and six in the third). Above the neuron combination, the learning rates are displayed from left to right, ranging from 0.0001 (slowest) to 0.01 (fastest). The y-axis represents the satisfiability of the priority as a percentage, with each column corresponding to a tested learning rate. The influence of the reward function on the results was found to vary. In particular, the left figure (a) illustrates that Reward Function A has a more significant impact, producing the best results when using a learning rate of 0.001, which was previously used as a starting value. This learning rate achieved results above 80% for all three neuron combinations, outperforming other learning rates within the same range. It was observed that a value in the middle range worked best for the classification in combination with Reward Function A. Even a slightly faster learning rate of 0.005 resulted in a loss of at least 5%. Increasing the learning rate beyond this point led to higher losses, rendering the model unusable, as demonstrated by

the „6/5“ distribution, which achieved only 21.71% accuracy. Further reduction of the learning rate did not improve the results. The response to variations in the learning rate differs in the middle figure (b) and the right figure (c). In these cases, the results tend to be more stable, with the best performance typically achieved using a learning rate of 0.001. The variation in results with different learning rates is generally smaller compared to reward function A. For a network of 15 neurons, the difference between the best and worst results is only 2.69%. Reward functions B and C demonstrate better performance across different learning rates than reward function A. However, appropriate tuning of the learning rate leads to the best results for reward function A, with satisfiability exceeding 82% in two instances. Nonetheless, this improvement is minimal compared to the other reward functions, and in some cases, the difference is less than one percent. The regression model with reward function A exhibits distinctive behavior, as depicted in figure 3.16. Notably, the regression model results are more consistent regardless of the reward function. Learning rates have minimal effects on the results, except for the left figure (a) in the „32/32/32“ distribution, where the result declines to 54.86% at a learning rate of 0.005. However, this occurrence is unique. The regression models achieve the best results in the „6/6/6“ distribution, with a maximum value of 84.28%. Remarkably, the regression models consistently achieve satisfiability values of over 80%.



(a) Classification model with reward A

(b) Classification model with reward B



(c) Classification model with reward C

Figure 3.16: Impact of Learning Rate on Priority Fulfillment in Regression Models

3.8 Performance Comparison of Models in Dynamic Environments

This section focuses on the performance comparison of classification and regression models in the context of a dynamic environment characterized by a fluctuating number of resources that can be added or removed over time. Specifically, the best-performing models identified during the training phase are chosen and subjected to a comprehensive evaluation against each other. The ability to manage a dynamic environment with varying resources is one of the primary contributions of this thesis. Consequently, this section presents the experimental setup, including the test conditions, and provides an analysis of the results obtained from testing the models within the dynamic environment.

3.8.1 Network Adaptation for Optimized Dynamic Testing

In order to adapt the network to the new test environment, the first essential step is to address hardware limitations. As part of this process, adjustments were made to the number of clients, which was reduced from 40 to 30, while the number of resources was increased from 6 to 15. The configuration details for the old and new resources are provided in Table 3.6. The settings for the old resources remained unchanged and were utilized in the same manner as in the previous tests. Conversely, the new resources were specifically configured to be more appealing to the model, featuring zero response time and high performance. This approach aimed to incentivize the model to assign tasks to the recently introduced resources.

Resource	Response time	Performance	Operation
0	0,0	0,3	1
1	0,0	0,3	2
2	0,0	0,5	1
3	0,0	0,5	2
4	0,5	0,8	1
5	1,0	1,0	2
6	0,0	1,0	1
7	0,0	0,4	2
8	0,0	0,6	1
9	0,0	0,8	2
10	0,0	0,4	1
11	0,0	0,6	2
12	0,0	0,8	1
13	0,0	1,0	2
14	0,0	1,0	1

Table 3.6: Resource Settings for Dynamic Testing Configuration

The previously trained models were utilized in order to save time on retraining and ensure the application of the most effective classification and regression models. Table 3.7 presents a summary of the chosen models and their respective

outcomes. Throughout the test procedure, as explained in section 3.7, three distinct test datasets were employed for each model.

Model no.	Model type	Reward function	Fulfillment priority
1059	Classification	A	82,20%
1284	Classification	B	81,53%
1196	Classification	C	80,41%
1108	Regression	A	84,28%
1286	Regression	B	83,14%
1236	Regression	C	81,17%

Table 3.7: Selected Models for Dynamic Testing and Associated Outcomes

In order to evaluate each step under varying resource availability, a total of 4500 tasks were processed during each test. The following predefined steps were executed based on the varying number of resources:

- After 500 tasks, resource 5 was taken down.
- After 1000 tasks, resources 4 and 3 were taken down.
- After 1500 tasks, resources 3, 4, and 5 were brought back up.
- After 2000 tasks, resource 6 was added.
- After 2500 tasks, resources 7 and 8 were added.
- After 3000 tasks, resources 9, 10, and 11 were added.
- After 3500 tasks, resources 12, 13, and 14 were added.
- After 4500 tasks, the test was finished.

The testing methodology employed in this thesis involved a progressive increase in the resource load until a threshold was reached, triggering a subsequent decrease and return to the initial state. It was anticipated that the introduction of additional resources would lead to enhanced outcomes and improvements over time.

3.8.2 Performance Analysis and Evaluation of Dynamic Behavior

When assessing model efficiency, it is crucial to take into account the role of rewards. The reward function plays a significant role by assigning a positive reward ranging from 1 to 3 when a priority is successfully fulfilled. Conversely, a negative reward ranging from -1 to -3 is given when a task takes an excessive amount of time to complete. Moreover, if a requested operation is assigned to a resource incapable of handling it, a negative reward of -5 is assigned. The achieved reward during the testing period serves as the basis for the initial evaluation and is presented for both the classification and regression models (refer to figure 3.17).

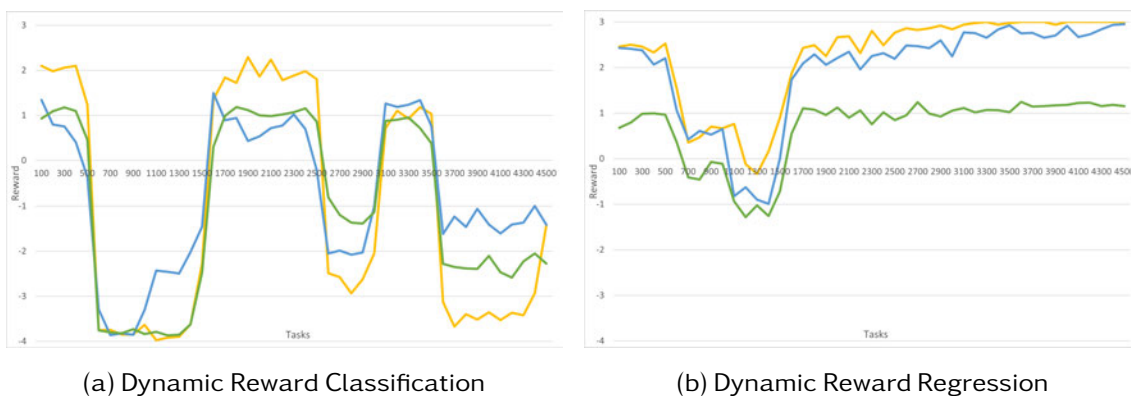


Figure 3.17: Comparison of Reward Functions in Dynamic Environment: Time Evolution of Rewards in Two Models

The classification model demonstrates effective results with a positive reward distribution for up to 500 tasks. However, beyond the 500 task threshold, when the most powerful resource for operation two (resource number five) is no longer available, the results start to decline significantly. Negative results in the range of -4 indicate that the models frequently make incorrect decisions rather than just being overloaded. The observation that all three reward functions exhibit poor results in this range implies that the reward functions alone cannot be solely held responsible. By the 1000 task mark, resource four, the most powerful resource for operation one, also fails, resulting in no discernible change and further supporting the occurrence of frequent incorrect decisions. However, when resources four and five resume operation after 1500 tasks, the results promptly recover and settle into a similar range of values as at the beginning of the test. The addition of the

first additional resource, resource six, after the 2000 task threshold does not have the expected effect, as reward function B shows a negative trend. Similarly, the addition of two more resources, resources seven and eight, after the 2500 task threshold should have led to an improvement, but instead, it causes a notable decline. Although the decline is not as severe as at the 500 task threshold, it is unexpected considering that more resources are available, and a better reward distribution was anticipated. Subsequently, every 500 tasks, three more resources were added until the 3500 task threshold, but no further improvements were observed up to the end of the test at 4500 tasks, and deterioration continued. In contrast, the regression model on the right initially exhibits higher rewards. However, a decline can be observed at the 500 task threshold, although smaller in comparison to the classification model. It is noteworthy that reward functions A and B can still yield positive rewards, while reward function C declines to a slightly negative value of -0.3 . The graph also illustrates the results of the second failure, which led to the loss of resource number four, resulting in further degradation of the model. Reward function B also slipped into the negative range, similar to function C, while reward function A remained predominantly positive. Notably, when the failed resources were restored, the results normalized, as depicted in the graph. Reward functions A and B showed an upward trend starting around task 2000, explained by the increasing number of resources added in 500 task intervals until task 3500 when the last three resources were added. This trend demonstrated a continuous increase in rewards until the end of the test. The expected results of the regression model aligned with the actual results. The classification model yielded unexpectedly poor results, even in regions where improvement was anticipated. To understand the reasons behind this decline in reward despite an increasing number of resources, an analysis was conducted to determine if tasks were being sent to resources capable of performing the appropriate operations. Figure 3.18 illustrates the applicability of tasks to the resources, and it is evident that the classification model made a significant number of incorrect decisions.

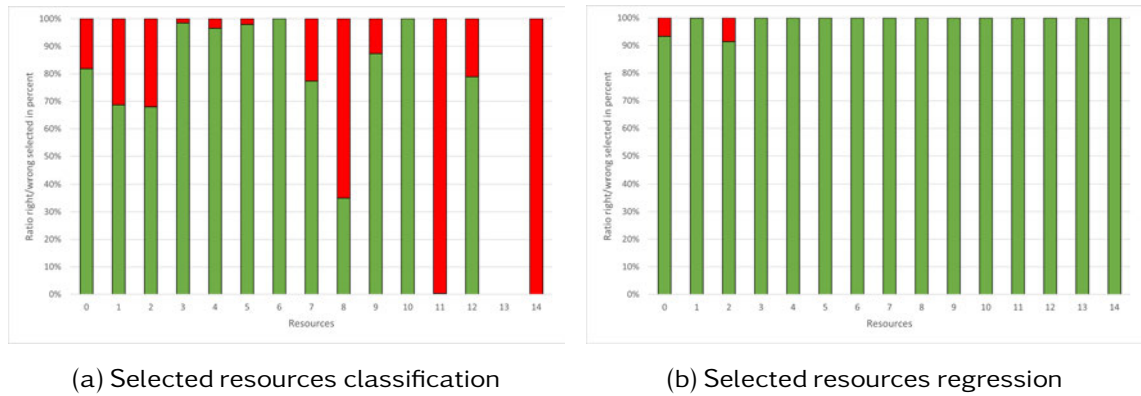


Figure 3.18: Evaluation of the model's ability to select resources matching the task's requirements

Notably, after 2500 tasks, there were frequent incorrect selections of specific resources, namely 8, 11, and 14. These incorrect selections contributed to the negative range of rewards shown in figure 3.18(a). Resource 13 was not selected at all by the classification model, indicating that despite its strong performance, it did not contribute significantly to the system's value. Similar observations were made for resources 0 and 1. The high penalty within the range of 500 tasks suggests that the results were adversely affected by these two resources. To determine the optimal model with the most effective reward function, a final graph was generated. Figure 3.19 presents the models that performed the best in a dynamic environment, based on the percentage of priority fulfillment.

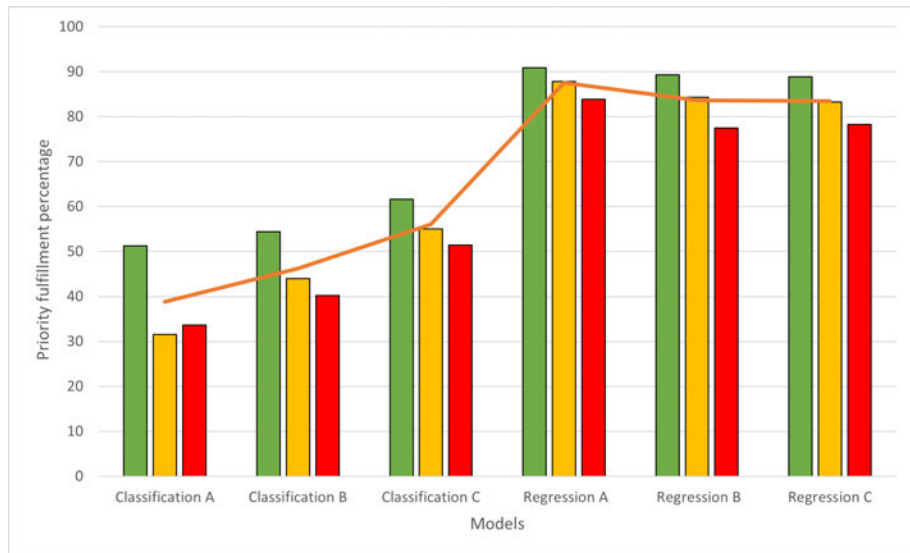


Figure 3.19: Comparative Analysis of Model and Reward Function Performance (Green denotes first test with the lowest difficulty level, yellow is the second with moderate difficulty level, and red represents the third and most challenging test. The average performance across all tests is represented by the orange curve)

The horizontal axis in the graph represents the six models utilized for the dynamic test, while the vertical axis represents the percentage of priority fulfillment. Each bar corresponds to a specific set of test data, enabling the evaluation of a model's effectiveness on a particular test set. The color coding denotes the associated test and provides insights into the difficulty level of each test. The results clearly demonstrate the impact of issues with the classification model, as it exhibits significantly poorer performance compared to the regression models. The most effective regression model achieved an efficiency of 87.5%, which is 56.25% higher than the best-performing classification model, with an efficiency of 56%. Among the classification models, the reward function C demonstrated the highest performance, followed by A and B. The three regression models displayed similar levels of efficiency.

3.9 Summary

This chapter presents the development of an intelligent system for resource allocation in an EC environment. The system utilizes an AI agent and a testing environment created using mininet. Various model types and reward functions were implemented and evaluated to determine the optimal approach. The study outlines the necessary steps for creating the models required for dynamic behavior testing. Each step was carefully evaluated to determine the best settings for the intended use case. Several hyperparameters were introduced to influence the model architecture and behavior, with three of them identified as having a significant impact. The analysis reveals that the regression model outperforms other models in terms of dynamic behavior. Its ability to evaluate one resource at a time and select the highest evaluated resource makes it well-suited for dynamic systems. Furthermore, this model requires minimal data preparation, resulting in concise code. It is relatively small, with a limited number of inputs, outputs, and hidden neurons, reducing the computational and storage requirements. However, it is noted that the classification model can be optimized for better performance. In the section on "Fine-Tuning Model Parameters," comparable results to the regression model were obtained. The main difference lies in the use of a static number of resources in the classification model, which may contribute to its lower performance. The regression model, on the other hand, employs a rotation of resources, allowing the model to learn the values of various resources. The classification model employs specific input neurons that consistently represent the properties of a particular resource. This approach leads to static properties, such as power or response time, which remain unchanged during the training process. However, when a new resource replaces an existing one in a given location, the model struggles to accurately interpret the new values. As a result, the model makes irrational decisions, as evidenced by the observations made during dynamic testing. In conclusion, this chapter highlights the development and evaluation of an intelligent resource allocation system in an EC environment.

4 Efficient Edge Authentication for Software-Defined Networks

4.1 Introduction

In previous discussions, it was highlighted that EC extends both computational and storage resources to the edge of a network to cater to applications with stringent latency requirements and high demands. However, edge network environments are susceptible to a heightened risk of malicious attacks. The dependability of communication in networks is usually contingent upon the successful verification of content and identity authenticity. Increasing the security level of authentication, although beneficial, invariably demands more computational resources, time, and energy. This creates a paradoxical situation where there's a trade-off between the augmentation of security levels and the optimization of resources during computational offloading. In this chapter, we introduce the implementation of lightweight edge authentication technique, designed to ensure only verified, legitimate nodes gain access to the network. This method exploits the opportunities provided by the programming protocol-independent packet processors (P4) language support in SDN. In this arrangement, we delegate certain control logic elements to the edge; more specifically, switches offload the controllers from making local state-based decisions that do not necessitate a comprehensive knowledge of the entire network. This approach specifically involves the delegation of conventional security functions from specialized middleboxes to the data plane. The lightweight nature of this approach stems from its operational methodology. Unlike centralized authentication processes managed by the SDN controller, this method operates in a distributed and scalable manner across

multiple switches. This effectively offloads the controller and distributes the workload among several authentication nodes deployed on the switches. Consequently, each switch becomes responsible for authenticating a subset of the total client base, allowing multiple clients to undergo authentication simultaneously. Moreover, the system's scalability is enhanced through the adaptable number of authentication nodes; new nodes can be easily integrated by activating the functionality on additional switches to meet growing demand, thus maintaining a manageable load on each switch. Additionally, by conducting authentication directly on the switch, which is in closer proximity to the clients, latency is minimized, leading to quicker authentication processes.

4.2 Delegating Authentication to SDN Switches

SDN is an emerging network paradigm that separates the control of the network from the forwarding plane. In SDN, centralized controllers manage the data plane and make packet forwarding decisions for the switches. The main interface used to communicate with switches in SDN is OpenFlow [24], [224], which offers a uniform abstraction for configuring different network devices. This allows controllers to insert and update forwarding rules in flow tables, regardless of the switch vendor. The primary goal of OpenFlow was to enable network administrators to remotely reconfigure forwarding tables, collect network statistics, and handle packets that do not match any rules by redirecting them to the controller for further processing. However, this reactive approach introduces unwanted latency. In OpenFlow, any stateful processing is typically handled by the centralized controller, while switches are limited to installing forwarding rules provided by the controller. Although this centralized decision-making process is advantageous when global network visibility is required and time constraints are not critical, it can become a bottleneck [177]. In recent years, extensive research has focused on offloading stateful processing to switches in order to alleviate the burden on the controller and reduce signaling overhead between them. One significant outcome of this research effort is P4 [225]–[227], a high-level language that enhances the flexibility, extensibility, and power of software running on

SDN switches. P4 enables SDN switches to perform advanced packet processing, including stateful operations. It provides stateful memories such as counters, meters, and registers that maintain state information across multiple packets. By separating the control and forwarding planes and centralizing intelligence in the controller, opportunities arise to reduce network control and management complexity. However, in the context of security, there is no need to centralize tasks that solely rely on local state, and therefore, there is no benefit from the controller's network-wide knowledge [173], [228], [229]. On the contrary, involving the controller explicitly for each stateful processing operation and match/action table update introduces extra signaling and overloads, which may lead to controller failure. One such task that does not require the controller's network-wide knowledge is port knocking, an authentication technique used by network administrators [230]–[232]. Port knocking involves a specific sequence of closed port connection attempts to designated IP addresses, referred to as a knock sequence. When the exact sequence of packets is received, the firewall opens a specific port for the requesting host. Before this event, all packets are dropped. As argued earlier, we believe that equipping the data plane with intelligence can enhance network performance, reduce signaling load, offload the controller, and meet real-time requirements for certain applications. Consequently, many applications that require sophisticated hardware can run inexpensively and in a distributed manner on the data plane. In this thesis, we propose delegating security tasks that typically rely on sophisticated equipment to SDN switches, using the port knocking application as a practical example. For our use case, we generalize the port knocking method to function as an authentication mechanism for hosts or subnetworks intending to establish a connection with a specific server. We demonstrate how a single SDN switch can be programmed with P4 to efficiently handle the port knocking application and serve as an authentication unit at the network ingress.

4.3 Centralized vs. distributed approach

As previously mentioned, the centralized decision-making provided by the controller in SDN offers advantages and can enhance system security, particularly when the controller's comprehensive network visibility is necessary and time is not a critical factor. However, the explicit involvement of the controller in any stateful processing poses challenges. In the most favorable scenarios, this could lead to increased signaling load and processing delay. Should we examine the scenario where our port knocking application is integrated directly into the logically centralized controller, it becomes apparent that a potentially significant amount of signaling information (theoretically, all packets directed to all n ports in the sequence) would need to be sent to the controller. Furthermore, the controller would have to provide timely instructions for packet forwarding following a correct knocking sequence to prevent the loss of the initial legitimate packet. Additionally, the controller must allocate memory resources to monitor the state of all nodes attempts to initiate a connection. Both the surplus signaling and memory consumption are proportional to the length of the sequence and the number of nodes attempting to establish a connection. Moreover, embedding this application within the controller provides no significant advantage, as it does not exploit the controller's global network visibility or high-level security policies [233], but merely utilizes local states associated with specific flows on a single device. Another argument for delegating some control to the data plane, especially when applications rely solely on local flow/port states, is as follows: while stateful SDN may move some states to the data plane, the core logic is still maintained by the logically centralized control plane. Consequently, it faces potential issues arising from an inconsistent global view of the network, such as in a physically distributed controller setting due to controller crashes or disconnections [234]. However, by moving states to the data plane, the vulnerability for specific applications can be eliminated, and our port knocking application serves as an example. In this case, the operations performed by the switch are entirely local and independent of the potentially inconsistent global state maintained by the control plane.

4.4 Modeling Deterministic Port Knocking

Authentication using Finite State Machine

We utilize the Finite State Machine (FSM) to model the intended deterministic behavior of port knocking, functioning as an authentication unit within a switch. An FSM consists of a limited number of states and can only exist in one state at any specific time. Upon receiving external inputs, the FSM can transition from one state to another. It is referred to as deterministic when, given an input at a specific state, the machine will unambiguously transition to the next state [235]. This abstract model aligns perfectly with the intended behavior we aim to achieve. Our primary objective was to design this feature and implement it in P4, thereby facilitating switch authentication without controller involvement. To enhance the design's reconfigurability, we maintained a flexible port-sequence (both in length and order) that must be knocked by the initiating node. This sequence can be altered post-switch deployment, should it be cracked by a malicious node. We further introduced configurability to two elements:

1. The set of untrustworthy nodes: This set consists of nodes that must knock the correct secret sequence of ports before initiating a connection to the server.
2. The set of nodes requiring authentication: These nodes mandate that all connecting nodes be authenticated before establishing a connection with them.

We adopted the FSM as an abstract model comprising various states, inclusive of default and final states. The transition between states is triggered by two types of events. The first occurs when the connection initiator, requiring authentication to be considered a trustworthy node, knocks the correct port in sequence. This leads to a transition to the subsequent state, as illustrated in figure 4.1. A transition from the default to the first state is possible only if the connection initiator has knocked the initial correct port in the sequence. Consequently, a transition from the first to the second state can only occur if the connection initiator has knocked the second correct port in sequence, and so forth. The second event transpires

when the initiator knocks an incorrect port while in any state other than the default or final. This triggers a roll back action, reverting the state to default, as depicted in figure 4.1. If the initiator remains in the default state and has not knocked the first correct port in the sequence, no transitions occur. The final state can only be achieved when the connection initiator has correctly knocked the sequence of ports.

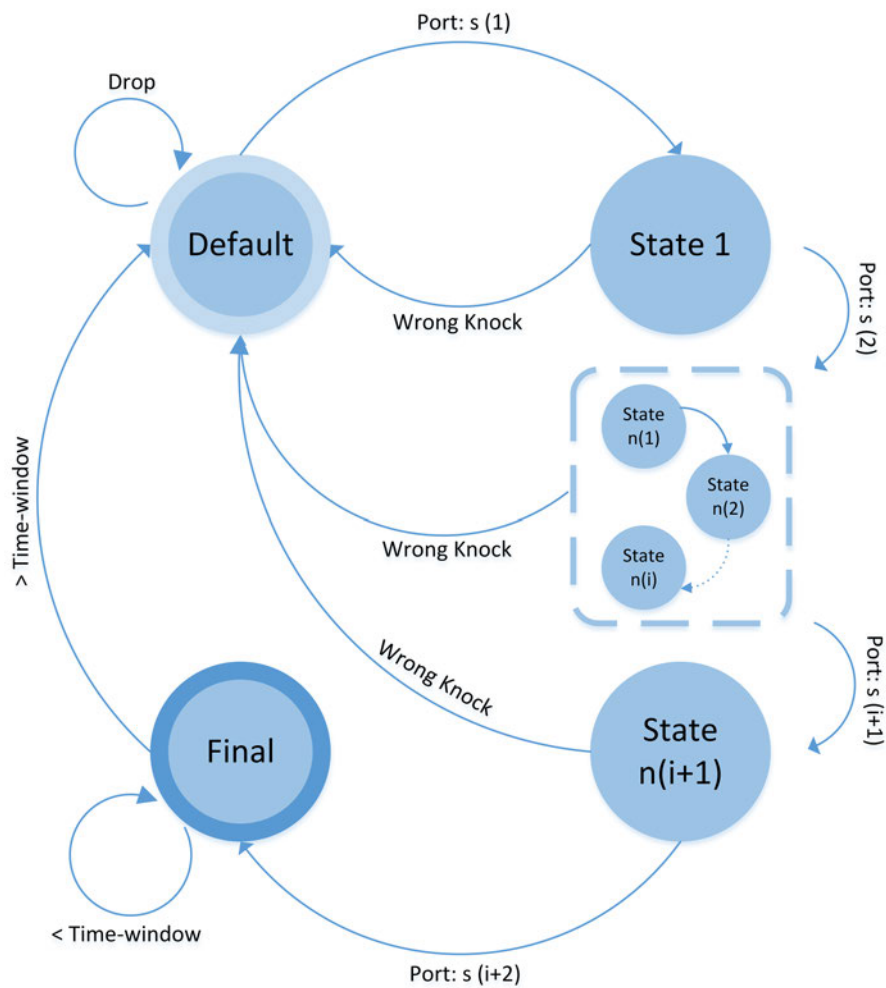


Figure 4.1: State Transition Diagram of Deterministic Port Knocking Authentication Using FSM

4.5 Authentication and Port Knocking

Implementation for Secure Network Access

This thesis demonstrates the proposed ideas using the behavioral model (bmv2) and mininet. A simple network topology consisting of one switch and four hosts

was simulated, as shown in figure 4.2. This topology was chosen for two primary reasons. First, we focused on developing an authentication node that operates within a single switch. While a larger topology could accommodate multiple authentication nodes on different switches, each responsible for a different sub-network, it would not contribute significantly to the clarification of the concept. Second, a larger topology becomes necessary to demonstrate switch failure. In such a case, another switch with the pre-installed program can be dynamically activated by the controller to take over the authentication task. However, this particular scenario falls outside the scope of this work and is aimed at showcasing the network's resilience.

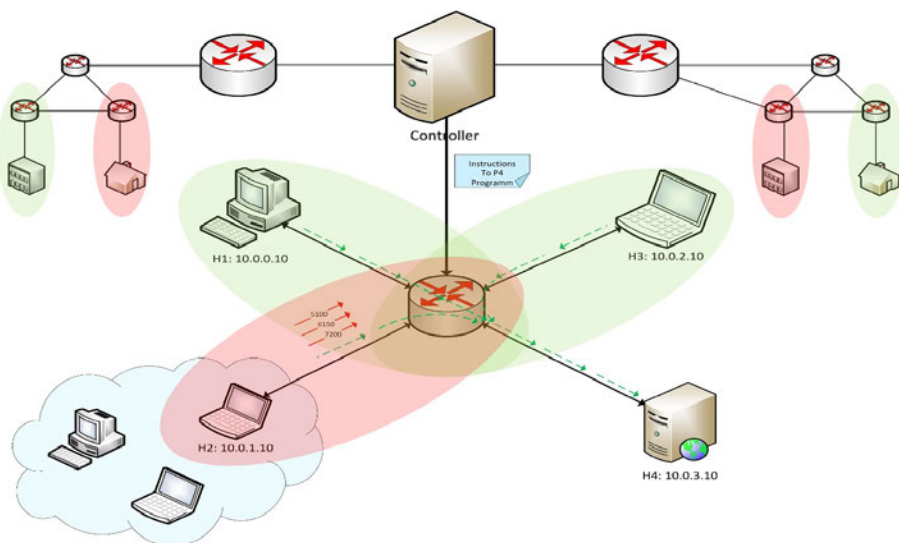


Figure 4.2: A Visual Representation of the Reference Topology

In our implementation, we designed a scenario where each host, except H2, can initiate connections to any other host, with the switch merely forwarding the packets to their intended destinations. However, H2 is considered an untrustworthy node and needs to authenticate itself before initiating a connection to the server H4. To achieve this, we employed an authentication application based on the principle of tickets. Nodes possessing a valid ticket are permitted to initiate connections with other nodes. We distinguish between benign nodes and those suspected of being untrustworthy. Initially, benign nodes are granted valid tickets, enabling them to establish connections with any other node without authentication. In contrast, nodes suspected of being untrustworthy, potentially due to past abnormal activities, are not granted valid tickets. This ensures that they must authenticate

themselves through a port knocking process. Only if they successfully knock the correct secret sequence of ports will they be granted a valid ticket, allowing them to initiate a connection. In our scenario, once H2 successfully completes the port knocking process, it acquires a valid ticket (permission) to establish a connection with H4 for a specific predefined period of time. After this time elapses, the ticket is invalidated, and H2 must authenticate itself again. This measure prevents tampered nodes from using old tickets to initiate unauthorized connections to the server. Motivated by the FSM, we have implemented an additional security measure to prevent nodes from accidentally or intentionally bypassing the correct sequence of ports by knocking arbitrarily long sequences. To achieve this, we have defined a specific condition to trigger a transition to the subsequent state as the pair [Port, state]. In our implementation, a knocked port is considered correct only if it is the next expected port number in the sequence and the assigned state in the P4 program for the knocking node matches the correct state in the [Port, state] pair. This ensures that only nodes that knock the port sequence in the predefined order can be successfully authenticated. Otherwise, authentication fails, and the node needs to restart the port knocking process from the first port in the sequence. Unless the correct port-sequence is provided and the ticket granted, all packets dispatched by the node are discarded at the switch without any feedback relayed to the node. Hence, packets are only forwarded to their intended destination after successful provision of the correct port-sequence and acquisition of the ticket. This strategy eliminates the possibility of a malicious node deducing any correct sequences. To demonstrate the process, let's consider a secret port sequence of (5100, 6150, 7200). Initially, each suspected node that needs to authenticate itself and obtain a ticket is assigned state 0 (Default). The switch expects the initial pair [5100, 0]. When the node knocks on port 5100, this results in the correct pair, triggering a transition to state 1. In the next step, the switch expects the pair [6150, 1]. Therefore, only if the node knocks on port 6150, the condition is satisfied, and a transition to state 2 occurs. If the node knocks on a different port, a rollback to the pair [5100, 0] is triggered, and state 0 is reassigned to the node as depicted in Table 4.1. If the rollback transition is omitted, the switch retains state 1 assigned to that node, even if it knocks

incorrect ports after knocking port 5100. Thus, if the node accidentally knocks port 6150 at any time, it mistakenly transitions to state 2, which can erroneously permit a node to manipulate the correct sequence via an arbitrary long sequence of ports that may accidentally encompass the correct sequence. On the other hand, implementing this functionality on the logically centralized controller would result in a new packet-in message for each incoming packet to the switch, as the switch would not have any installed rules to handle these packets. Roughly, this would generate up to l (the length of the port sequence) additional messages towards the controller for each new connection attempt by a node to the server. Moreover, the controller must retain the state of each port knocking operation in order to ultimately install a new rule in the switch, which then either forwards or blocks packets from that node. An attacker could exploit such an implementation to execute a DoS attack by consuming computational resources on the controller through the use of a set of spoofed IP addresses.

Knocked port	Assigned state	Expected pair	Resulting pair	Resulting state	Action	Ticket granted
5100	0	[5100, 0]	[5100, 0]	1	Drop	False
6150	1	[6150, 1]	[6150, 1]	2	Drop	False
6200	2	[7200, 2]	[6200, 2]	0	Drop	False
7200	0	[5100, 0]	[7200, 0]	-	Drop	False
*	0	[5100, 0]	[*, 0]	-	Drop	False
5100	0	[5100, 0]]5100, 0]	1	Drop	False
6150	1	[6150, 1]]6150, 1]	2	Drop	False
7200	2	[7200, 2]	[7200, 2]	3	Drop	True
Any	3	-	-	-	Forward	True

Table 4.1: Port Knocking Authentication Process
 (*) All ports except the correct port 5100.

4.5.1 Dynamic Switch Reconfiguration for Enhanced Authentication and Network Security

The implementation of an authentication application with P4 in the switch offers several advantages. One significant benefit is the reduction of malicious traffic in

the network. By filtering out traffic from suspicious nodes at the network ingress, there is no need to forward this traffic to specialized middleware within the network for filtering. Once the controller injects the authentication rules into the switch's designated table, it is no longer involved in the decision-making process or the addition of new forwarding rules for successive incoming packets. This reduces the load on the controller and improves overall network performance. Furthermore, if the switch goes down, the authentication application can be activated at runtime on any other switch in the network without disrupting network availability. Despite the numerous advantages, it is important to note that this authentication method does not offer comprehensive protection. To demonstrate this, we present an attack example that exploits the vulnerability of the authentication function in the switch to a memory saturation attack. In this scenario, an attacker with knowledge of the correct sequence of ports launches a large number of connection attempts (packets) from spoofed IP addresses to the first port in the sequence (e.g., port 5100). The switch, upon receiving these packets, checks its state table to determine the incoming flow's state. If there is no existing record for the source IP address, the switch assigns the next state to that IP. Since all incoming packets are destined for the correct port, the state of all the corresponding IPs will be updated to the next state (e.g., State 1). Consequently, the state table could be overloaded with entries, enabling an attacker to intentionally create thousands of records and exhaust the switch's memory [236]. The impact of this vulnerability can be mitigated by configuring the switch with a timeout mechanism. If the switch does not receive a second packet from the same flow that progresses to the next port, it will reset the flow's state and remove it from the state table. Additionally, distributing the authentication functionality across multiple switches in the network, with efficient load balancing between them, can significantly reduce the impact of such an attack. To ensure our implementation is realistic, scalable, and capable of handling network changes seamlessly, we designed it to be dynamic and reconfigurable at runtime. Nodes suspected of being untrustworthy and requiring authentication before initiating a connection are defined as pairs of source and destination with IPv4 addresses. The source represents the suspected node or subnetwork, while the destination represents

the server or protected node. Hence, a node may be deemed suspicious when initiating a connection to a specific node/server but may not be suspected when connecting to another node/server. To add a node (connection initiation) to the list of suspicious nodes, the controller merely needs to add a rule to the responsible table. Similarly, removing this rule effectively eliminates the node from the list. This process can be dynamically managed at runtime, thereby enabling swift reconfiguration of the switch to handle any new network incidents. To implement this behavior, we have employed the Hit/Miss construct in P4 [24]. If a specific node's record exists in the table (Hit), it is considered suspicious and must authenticate first to obtain a ticket. Conversely, if a node's record does not exist in the table (Miss), it is deemed benign and, by default, possesses a ticket (see figure 4.3).

```

apply(ticket_deny_table) {
    hit {
        host_need_processing();
    }
    miss {
        apply(grant_ticket_for_not_suspicious_table);
    }
}

```

Figure 4.3: Refining the Hit/Miss construct: Identifying trustworthiness in node selection

The proposed approach allows the controller to configure the secret sequence of ports that need to be knocked in a specific order to obtain a valid ticket. This configuration can be modified at runtime, providing flexibility and enhancing security. The controller has the capability to delete the current sequence or a subsequence at any given time, and replace it with a new sequence of the same length or a longer one, thus increasing the difficulty of cracking the sequence. Consequently, an attacker would need to correctly guess the sequence out of a vast number of possible permutations, specifically 65535^l (where "l" denotes the length of the sequence). To demonstrate this capability, we utilized a Command Line Interface (CLI). In the initial configuration, we set the sequence to be (5100, 6150, 7200) using the following commands (refer to figure 4.4).

```

table_add update_state_table update_state 5100 0 => 1 0
table_add update_state_table update_state 6150 1 => 2 0
table_add update_state_table update_state 7200 2 => 3 1

```

Figure 4.4: CLI commands to set the initial knock-sequence

In this sequence, 5100 represents the first port in the sequence, with 0 being the initial state, 1 denotes the state after the transition, and the last 0 in the first line represents the ticket validity (0 = not valid, 1 = valid). To modify this sequence, the controller must delete these records and introduce new rules incorporating the new sequence, depicted in figure 4.5. This results in a runtime sequence modification, and only the potential nodes capable of knocking the new sequence in the correct order will gain a valid ticket.

```

table_delete update_state_table 0
table_delete update_state_table 1
table_delete update_state_table 2

table_add update_state_table update_state 6100 0 => 1 0
table_add update_state_table update_state 8150 1 => 2 0
table_add update_state_table update_state 9500 2 => 3 0
table_add update_state_table update_state 5100 3 => 4 1

```

Figure 4.5: CLI commands to change the knock-sequence

In the primary configuration, node 10.0.1.10 (H2) was identified as potentially suspicious within the context of connection attempts to 10.0.3.10 (H4). Subsequently, we dispatched a TCP stream using iperf from H2 towards H4. As illustrated in figure 4.6, all packets were initially dropped since H2 had not yet authenticated itself. Using a second terminal, we sent UDP packets from H2 to H4 with destination ports 5100, 6150, and 7200 to obtain a valid ticket. The first attempt failed due to the arrival of another packet with an incorrect port in between. This triggered a rollback action, forcing the host to restart the sequence from the beginning. Only when the sequence was correctly knocked in the proper order, the host was authenticated, and the ticket was granted. Subsequent packets were then forwarded normally to the destination. The issued ticket is only valid for a predefined duration. Hence, the switch examines the arrival time of all subsequent packets and invalidates the ticket once the validity period expires. All packets are then dropped until the node secures a new valid ticket.

No.	Source	Destination	Protocol	Length	Source Port	Destination Port	Comment
1	10.0.1.10	10.0.3.10	TCP	74	58778	5001	
2	10.0.1.10	10.0.3.10	TCP	74	58778	5001	not authenticated >> drop
3	10.0.1.10	10.0.3.10	TCP	74	58778	5001	
4	10.0.1.10	10.0.3.10	TCP	74	58778	5001	
6	10.0.1.10	10.0.3.10	UDP	42	53	5100	
9	10.0.1.10	10.0.3.10	UDP	42	53	6150	
11	10.0.1.10	10.0.3.10	UDP	42	53	7500	authentication started but packet 11 leads to rollback action
12	10.0.1.10	10.0.3.10	UDP	42	53	7200	
13	10.0.1.10	10.0.3.10	TCP	74	58780	5001	
14	10.0.1.10	10.0.3.10	TCP	74	58780	5001	no ticket >> drop
15	10.0.1.10	10.0.3.10	TCP	74	58780	5001	
17	10.0.1.10	10.0.3.10	UDP	42	53	5100	
19	10.0.1.10	10.0.3.10	UDP	42	53	6150	
21	10.0.1.10	10.0.3.10	UDP	42	53	7200	successful authentication
22	10.0.1.10	10.0.3.10	TCP	74	58782	5001	
23	10.0.3.10	10.0.1.10	TCP	74	5001	58782	
24	10.0.1.10	10.0.3.10	TCP	66	58782	5001	ticket gained >> forwarding
25	10.0.1.10	10.0.3.10	TCP	74	58782	5001	
26	10.0.3.10	10.0.1.10	TCP	74	5001	58782	
27	10.0.1.10	10.0.3.10	TCP	66	58782	5001	
28	10.0.1.10	10.0.3.10	TCP	90	58782	5001	
29	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
30	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
31	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
32	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
33	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
34	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
35	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
36	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
37	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
38	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
39	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
40	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	
41	10.0.3.10	10.0.1.10	TCP	66	5001	58782	
42	10.0.1.10	10.0.3.10	TCP	1514	58782	5001	

Figure 4.6: Visualizing H2's Port Knocking Technique: Wireshark reveals the port-sequence manipulation for secure ticket authentication

While the switch was active, we reconfigured it with a new secret sequence, as depicted in figure 4.5. This reconfiguration is typically conducted by the controller in response to specific network incidents or anomalies. Thus, any potential node that seeks to initiate a connection must first present the new sequence to procure a valid ticket, as shown in figure 4.7. A previously cracked sequence is no longer applicable.

No.	Source	Destination	Protocol	Length	Source Port	Destination Port	Comment
1	10.0.1.10	10.0.3.10	TCP	74	58802	5001	
2	10.0.1.10	10.0.3.10	TCP	74	58802	5001	not authenticated >> drop
3	10.0.1.10	10.0.3.10	TCP	74	58802	5001	
5	10.0.1.10	10.0.3.10	UDP	42	53	5100	
7	10.0.1.10	10.0.3.10	UDP	42	53	6150	old sequence
9	10.0.1.10	10.0.3.10	UDP	42	53	7200	
10	10.0.1.10	10.0.3.10	TCP	74	58804	5001	
11	10.0.1.10	10.0.3.10	TCP	74	58804	5001	no ticket >> drop
12	10.0.1.10	10.0.3.10	TCP	74	58804	5001	
14	10.0.1.10	10.0.3.10	UDP	42	53	6100	
16	10.0.1.10	10.0.3.10	UDP	42	53	8150	new sequence
18	10.0.1.10	10.0.3.10	UDP	42	53	9500	
20	10.0.1.10	10.0.3.10	UDP	42	53	5100	
21	10.0.1.10	10.0.3.10	TCP	74	58806	5001	
22	10.0.3.10	10.0.1.10	TCP	74	5001	58806	
23	10.0.1.10	10.0.3.10	TCP	66	58806	5001	ticket gained >> forwarding
24	10.0.1.10	10.0.3.10	TCP	74	58806	5001	
25	10.0.3.10	10.0.1.10	TCP	74	5001	58806	
26	10.0.1.10	10.0.3.10	TCP	66	58806	5001	
27	10.0.1.10	10.0.3.10	TCP	90	58806	5001	
28	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
29	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
30	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
31	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
32	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
33	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
34	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
35	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
36	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
37	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
38	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
39	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
40	10.0.3.10	10.0.1.10	TCP	66	5001	58806	
41	10.0.1.10	10.0.3.10	TCP	1514	58806	5001	
42	10.0.3.10	10.0.1.10	TCP	78	5001	58806	

Figure 4.7: Visualizing H2's Port Knocking Technique: Wireshark reveals the new port-sequence manipulation for secure ticket authentication

4.5.2 Enhancing Network Security: Authentication-based Defense Against Port Scans

Attackers frequently conduct arbitrary port scans on IP addresses, seeking vulnerable servers to exploit. Port scanning methodologies [236], [237] function by dispatching requests to a broad range of port addresses on a host, and subsequently examining the responses to discern which ports on this node are open. Our proposed application is equipped to mitigate such attacks initiated from untrusted nodes. Nodes not suspected of malicious intent are permitted to establish a connection with the host/server without the need for authentication. Conversely, nodes suspected of unscrupulous connection initiation to a particular node/server are initially blocked. Any incoming packets from these suspected nodes are intercepted and discarded by the switch before they reach their intended destination. Essentially, as long as a suspected node has not authenticated itself, the node it aims to exploit remains invisible to it. To illustrate this mechanism, we utilized nmap from H2 to conduct a port scan on the safeguarded host, H4 (according to the initial configuration). Since H2 had not yet authenticated at that point, all packets sent from nmap were dropped by the switch, leading the scanner to return the message "Host seems down".

4.6 The experimental setup

The experimental setup depicted is utilized for implementing and evaluating our proposed method.

Table 4.2: System Configuration

Hardware	
graphic card	Nvidia RTX 2080 TI, 11GB GDDR6
processor	AMD Ryzen 7 1800x, 8 CPU Kerne, 3,6 GHz
RAM	G.Skill Trident Z DDR4 32 GB (4x8)

Software	Version
Mininet	2.3.0d6
Ryu Controller	4.34
P4	14

4.7 Evaluation

To evaluate the performance of our authentication technique, two methods will be employed. The first method involves comparing the performance of packet forwarding in two scenarios: one where the node is initially designated as benign, requiring no secret sequence of ports to be knocked, and another where the node is initially classified as untrustworthy, necessitating the knocking of the secret sequence of ports to obtain a ticket. Once a valid ticket is obtained, the switch only checks the arrival time of subsequent packets against the ticket's expiration time. This experiment aims to investigate whether the ticket validity check has a negative impact on the switch's performance. The second method focuses on comparing the performance of two switches. The first switch implements our proposed authentication technique along with packet forwarding, while the second switch solely performs packet forwarding without any additional functionality. By comparing the performance of these two switches, we can evaluate the effectiveness and efficiency of our proposed implementation.

4.7.1 Switch Performance Analysis: Evaluating Time Window Validity Check

To evaluate the switch performance in both scenarios, we conducted a series of measurements comparing the throughput under varying circumstances. This involved transmitting fifty TCP streams from the benign host H3 to H4, and a similar number of TCP streams from the untrustworthy host H2 to H4. Each stream lasted ten seconds and was initiated only after authentication via a ticketing system had been established (refer to figure 4.2 for the network topology). We computed the average throughput from these measurements. As illustrated in figure 4.10 a and b, the average throughput remained relatively consistent, implying that the switch's forwarding functionality performance remained unaffected once a suspicious node has been successfully authenticated. In a separate test, represented in figure 4.8, we studied the switch performance when both nodes, H2 and H3, sent TCP streams to H4, nearly concurrently. In this scenario, H3 initiated transmission while H2 was still in the process of authentication. We used a Python script from a secondary terminal to complete the authentication

sequence, securing a valid ticket for H2, and subsequently returned to the primary terminal to initiate the stream from H2 to H4. The manual process of switching between terminals introduced a slight delay, resulting in a delayed start of the stream from H2, as depicted in figure 4.8. We conducted another experiment, this time with a two-minute duration for each stream, with H2's authentication already established. The goal of this experiment was to study the effect of ticket expiry while the stream was in transit through the switch. Figure 4.9 reveals two instances of ticket expiration within the two-minute window (given the 50-second ticket validity period). This expiration led to a noticeable decline in overall throughput, an expected outcome under such conditions.

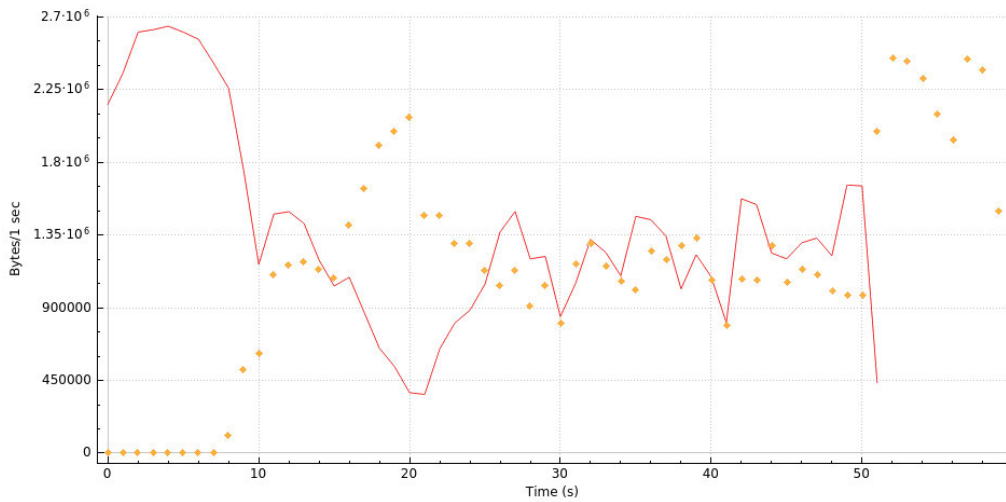


Figure 4.8: Comparing Throughput of H2 and H3 TCP Streams to H4 (H2: Dotted Curve, H3: Solid Curve) with H2 Authentication Delayed Initially

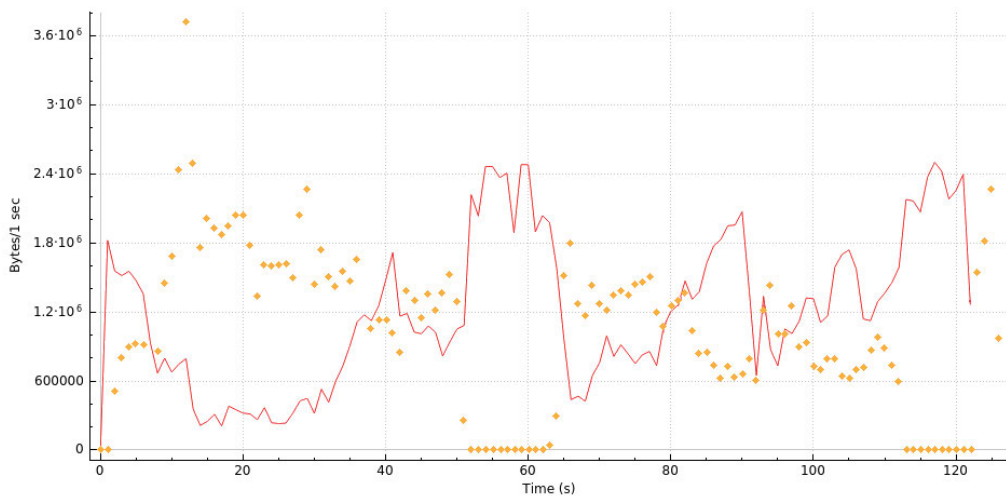


Figure 4.9: Doubled Instances of Ticket Expiration Within a Two-Minute Frame (Validity Time of 50 Seconds)

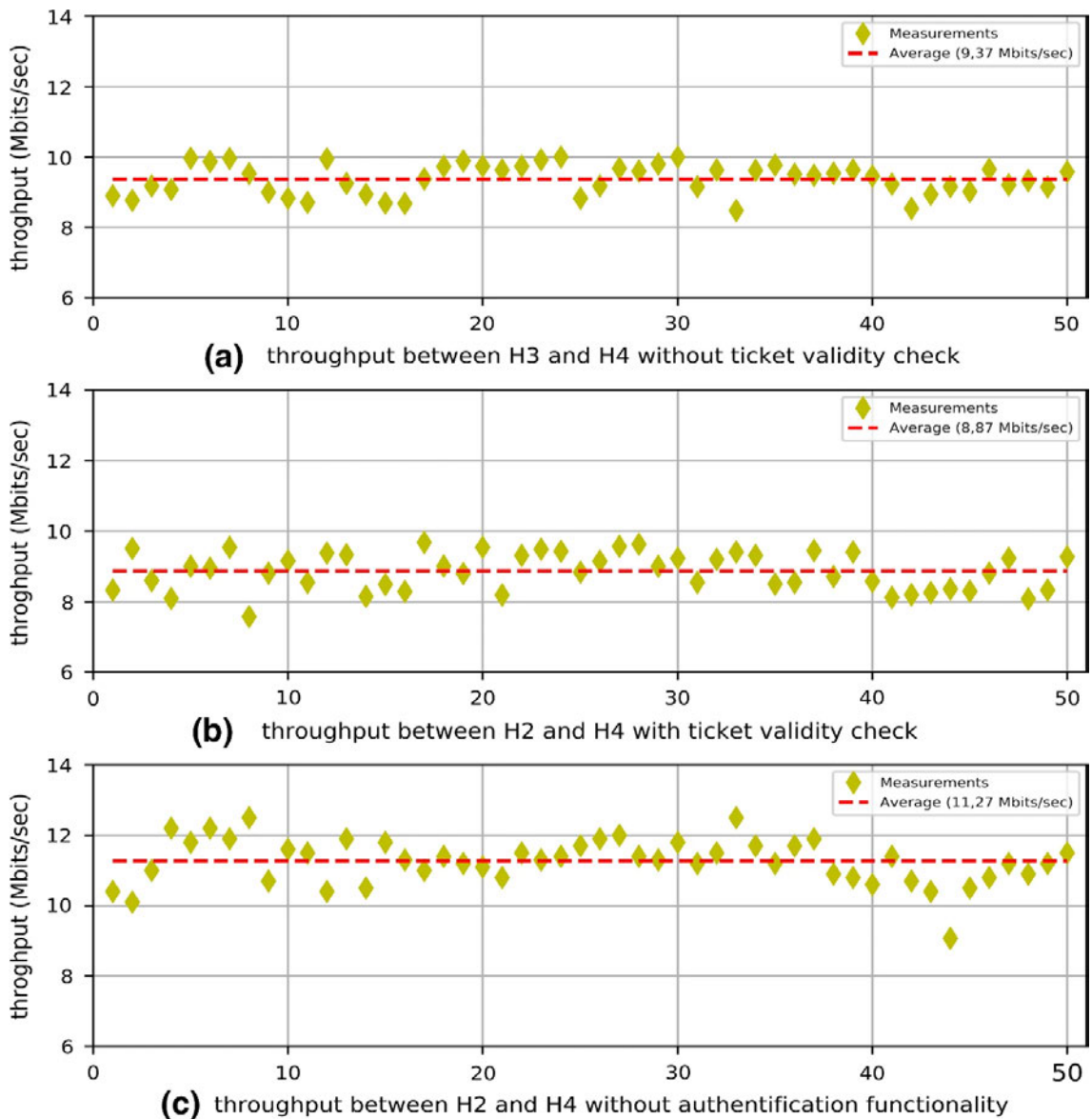


Figure 4.10: Average throughput for fifty measurements in different scenarios

4.7.2 Switch Performance: Evaluating the Impact of Authentication Functionality

The objective of this evaluation was to assess whether our application has a detrimental effect on the performance of the switch and, if so, to what extent. To conduct the evaluation, we replaced the switch implementing our application with an identical switch (bmv2) that lacked any additional functionality beyond basic forwarding capabilities. To measure the impact, we conducted a test by transmitting fifty TCP streams from the host H2 to H4, each lasting for ten seconds. Subsequently, we calculated the average throughput across all streams. As

depicted in figure 4.10 c, the average throughput improved by approximately 20% compared to the switch running our application. However, as discussed previously, despite the seemingly lower throughput, the implementation of the authentication functionality within the switch offers several advantages over implementing it in the logically centralized controller.

4.8 Summary

This chapter has introduced a novel technique that utilizes P4 for authentication and port scan mitigation within switches. The technique's runtime flexibility and dynamic nature have been demonstrated, highlighting its applicability to real-world scenarios, particularly in EC environments. This technique effectively addresses security and privacy concerns in EC, while concurrently enhancing the trustworthiness of edge devices. It enables lightweight user authorization for edge services, conserving computing power and reducing authentication time for a large and ever-changing population of IoT devices. In conclusion, this chapter presents an innovative technique leveraging P4 for authentication and port scan mitigation in switches, showcasing its adaptability in real-world EC settings.

5 Scalable Distributed Homomorphic Encryption for Complex Computational Models

5.1 Introduction

In chapter 1, we have highlighted the rapid growth of high-tech industries and the resulting groundbreaking innovations, which have led to an exponential increase in data volumes across various sectors such as healthcare, finance, and manufacturing. Managing these vast amounts of data effectively necessitates additional computational resources. Consequently, we have acknowledged the significance of CC and EC as potent solutions for handling the surge in big data. However, the utilization of CC and EC raises significant concerns regarding data privacy and security, particularly when sensitive user information is stored and processed on publicly accessible cloud/edge servers. Traditional encryption techniques are insufficient in addressing the conflict between processing large data sets in powerful yet potentially untrusted CC and EC environments. HE, which allows computations on encrypted data, emerges as a potential solution. Nonetheless, the high computational complexity and lengthy processing times of FHE hinder its widespread adoption, especially in scenarios with limited computational resources and strict time constraints. To overcome these limitations, this chapter introduces DHE as a promising avenue to address scalability issues. The DHE-based data processing strategy leverages distributed computing techniques to partition the computational workload across multiple instances, thereby reducing the overall computational stress on a single instance. This approach is an ideal application

scenario for our resource allocation framework. As discussed in Chapter 3, the AI agent dynamically gathers information about available resources, such as computing power and supported operations, before allocating the most suitable resources for the task at hand. In the context of DHE), the AI agent can allocate the most appropriate resources, which support HE operations, to the subtasks that result from dividing the main task into several smaller subtasks. Since these subtasks are smaller and require less computational power, the AI agent can distribute them across a broader range of edge devices that are available at the time of the request. Furthermore, allocating resources to the encrypted subtasks enhances privacy protection, even when the resources are located in a public domain. Furthermore, we evaluate the performance of our proposed approach by comparing three commonly used FHE schemes (CKKS, BFV, and BGV) using the Microsoft SEAL library and arithmetic operations on vectors up to size 2^{16} in both centralized and distributed approaches.

5.2 Centralized vs. Distributed Approaches in Homomorphic Encrypted Data Processing

This section presents two approaches: the centralized approach and the distributed approach. In the centralized approach, all homomorphic encrypted data is processed by a single resource, typically a central server. This approach offers the advantage of simplified computation since all data is located in one place. However, it also poses a single point of failure. If the central resource becomes unavailable, the entire system fails. Furthermore, the central resource must possess sufficient computing power to handle all computations, which can be challenging for large-scale data processing. On the other hand, the distributed approach involves parallel computations performed by multiple resources. Each resource processes a portion of the encrypted data, which is divided into smaller chunks and sent to different resources for processing. Once the computations are complete, the results are merged to obtain the final output. The distributed approach offers several advantages over the centralized approach. Firstly, it enables a more scalable solution as the processing load can be distributed across

multiple resources. This reduces the risk of a single resource becoming a bottleneck and improves overall processing times. Secondly, it provides greater fault tolerance as the system can continue to function even if one or more resources fail. In conclusion, both the centralized and distributed approaches have their strengths and weaknesses. The choice between the two depends on the specific requirements of each application.

5.2.1 Centralized Approach

In this thesis, we investigate a centralized approach (depicted in Figure 5.1) for ensuring data confidentiality in a secure computation scenario. We consider the generation of two messages, $M1$ and $M2$, where each message M is represented as a list of integers, $M = \{i_1, i_2, \dots, i_n\}$, with values ranging from 1 to 1000. The length of the message, denoted by n , is determined based on the specific test session and can vary from 2^7 to 2^{16} . To maintain data confidentiality, the client encrypts both messages using a homomorphic encryption procedure, resulting in the creation of encrypted messages, $E1$ and $E2$. Subsequently, the encrypted messages, along with the required relinearization key, are transmitted to the server for computation. On the server side, the addition of $E1$ and $E2$ is performed, yielding a new encrypted message, $E3$. Further computation involves multiplying each element in $E3$ by 0.5, resulting in the generation of the final encrypted message, $E4$. It is important to note that the client exclusively possesses the decryption key required to decrypt the message. After the calculations are completed, the encrypted message is sent back to the client.

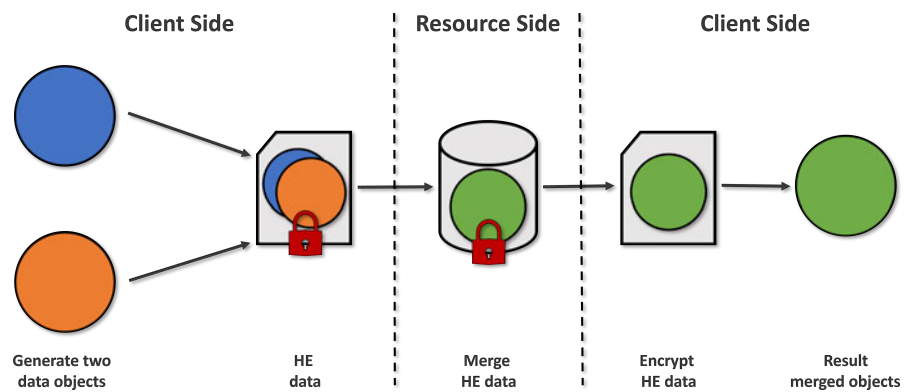


Figure 5.1: Centralized model

5.2.2 Distributed Approach

The primary focus of this thesis is the distributed approach, illustrated in Figure 5.2. In this approach, two messages, denoted as $M1$ and $M2$, are generated in the initial step, exhibiting similar content and structure as the centralized approach. However, the distinction lies in the availability of multiple workers for message processing. The set of workers is specified as $\{Worker_1, Worker_2, \dots, Worker_k\}$. Based on the chosen number of workers, messages $M1$ and $M2$ are divided into partial messages $M1_1, M1_2, \dots, M1_k$ and $M2_1, M2_2, \dots, M2_k$ correspondingly, ensuring that each worker receives a portion from both messages. These partial messages are encrypted using a homomorphic procedure, resulting in $E1_1, E1_2, \dots, E1_k$ and $E2_1, E2_2, \dots, E2_k$, which are then distributed to the respective workers. The workers carry out the same operations as in the centralized approach, i.e., addition and multiplication, but on the partial messages, yielding partial results, $E3_1, E3_2, \dots, E3_k$. These partial results are sent back to the client, where they are decrypted and combined to form an overall result $M3 = M3_1 \parallel M3_2 \parallel \dots \parallel M3_k$. The distributed approach offers several advantages over the centralized approach. Firstly, it enables parallel processing of messages, leading to reduced processing time. This feature is especially beneficial in large-scale computations, where substantial time savings can be achieved. Secondly, the distributed approach enhances security by encrypting the messages and distributing them among multiple workers, thus increasing the difficulty of unauthorized access. Additionally, this approach provides improved scalability since the number of workers can be easily adjusted to meet the processing requirements, allowing for efficient resource allocation.

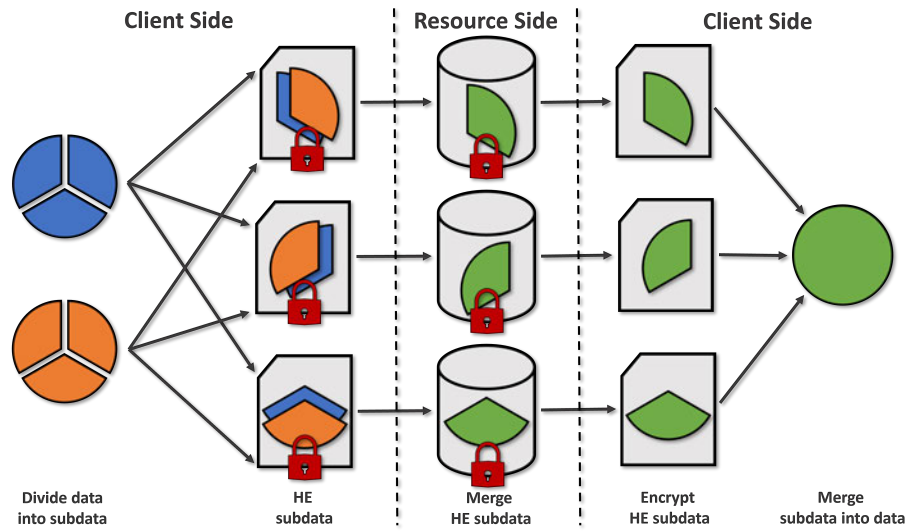


Figure 5.2: Distributed model

5.3 The experimental setup

The experimental setup depicted is utilized for implementing and evaluating our proposed method.

Table 5.1: System Configuration

Hardware			
processor	Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz		
RAM	G.Skill Trident Z DDR4 32 GB (4x8)		
Software	Version	Software	Version
mpi4py	3.1.4	kaleido	0.2.1
plotly	5.13.1	SEAL	4.0.0
pylatex	1.3.4	pySEAL	4.0.0
pandas	1.5.1	pybind	2.10.1
sympy	1.7.1	mpich	4.0.2

5.4 Evaluation

In this section, we present the findings of our scientific study. We provide a comprehensive overview of the libraries used, the evaluated schemes, and the time stamps applied. This detailed description aims to enhance understanding of the outcomes and facilitate replication of the experimental setup.

5.4.1 Distributed Homomorphic Computation with OpenMPI and SEAL

In this thesis, two libraries, namely openMPI and SEAL, were employed to facilitate distributed homomorphic computations:

OpenMPI: Is a communication framework that facilitates inter-process communication among multiple processes, including those distributed across different hosts or virtual machines [238]. It allows computations to be divided into multiple processes, which are then assigned priorities based on their ranks and processed by different machines. In this study, OpenMPI was utilized to implement a distributed approach for HE.

SEAL: Is an open-source library developed by the Cryptography Research Group at Microsoft Research, specifically designed to support homomorphic computations through various schemes [239]. As SEAL was originally designed for C++ platforms, a Python wrapper was employed in this study to enable Python support. In order to test a broader range of values, the predetermined safety constraints for SEAL, which enforce security standards proposed by [240], were deactivated during the test runs. These constraints impose restrictions on the allowable lengths of certain operations.

5.4.2 Homomorphic Encryption Schemes

In this study, we have implemented and evaluated three schemes based on a common operating principle. The process involves message encoding, followed by encryption. Subsequently, arithmetic operations are performed on the encrypted data. The data is then decrypted, decoded, and the final result is obtained. Figure 5.3 illustrates a timeline of the homomorphic operations.

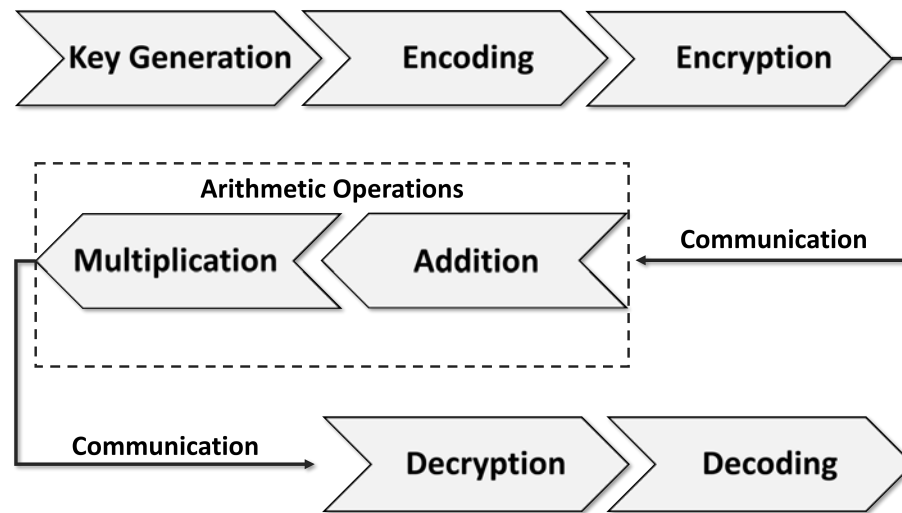


Figure 5.3: Composition of different used times

BGV/BFV: The Brakerski-Gentry-Vaikuntanathan (BGV) schema, proposed in 2011, and the Brakerski/Fan-Vercauteren (BFV) schema, introduced in 2012, belong to the second generation of encryption schemes capable of computing messages consisting of integers [241], [242]. Although these two schemes share structural similarities, they differ in their parameters. The BFV scheme is notable for maintaining a constant ciphertext modulus throughout the evaluation process, resulting in scaling independence. On the other hand, the BGV scheme utilizes multiple smaller moduli to effectively control the noise generated during computations through modulo switching. Despite belonging to the older generation of schemes, both BGV and BFV offer faster computational speeds compared to newer schemes. However, it is important to note that they are limited to processing integer values exclusively.

CKKS Scheme: The CKKS scheme, introduced in [243]–[245], demonstrates greater versatility compared to the previously proposed BGF and BFV schemes. Unlike the latter, CKKS supports multiplicative inverses, allowing for the implementation of divisions. Additionally, it facilitates computations involving complex numbers by encoding messages into real and imaginary values, which are then represented as polynomials. These polynomials serve as the basis for performing calculations, with the resulting outcomes approximated after decoding. However, it is important to note that this approach is more time-consuming when compared to the BGF and BFV schemes.

5.4.3 Time Analysis of Processing Steps

This section provides an in-depth analysis of the behavior of the two approaches by presenting the time records associated with each approach. To facilitate comprehension, the recorded times have been categorized into various processing steps, as illustrated in Figure 5.3. The initial steps involve the generation of keys, encoding of data, and encryption. In accordance with the scheme, the key generation process produces keys that are of the same length as the polynomial intended for encryption. Furthermore, scheme-specific functions, such as relinearization (required after multiplication), also necessitate key generation. As the values to be evaluated are provided as vectors, they need to be initially converted into polynomials through a process called encoding. The subsequent steps involve the mathematical operations of addition and multiplication. Finally, the data is decrypted and decoded in the remaining steps, employing the same method used during the encoding process.

5.5 Experimental Evaluation of Homomorphic Operations

In this thesis, the homomorphic addition and multiplication operations between vectors of varying sizes were investigated. The experiment focused on a set of vector sizes 2^n and employed suitable encryption parameters for each scheme. To assess the performance of the schemes, the processing time was measured using both centralized and distributed approaches. The evaluation was conducted by averaging the results of 50 test runs for each distinct message length (vector size). The results are presented in tables featuring a consistent format and acronyms. The tables include the time taken for key generation (Key Gen.), encryption (Encr.), which encompasses encoding time, communication (Com.), arithmetic operations (Arith.), decryption (Decr.), and the total time (Total). Furthermore, to facilitate comparison, an additional column (Impr.) was incorporated to indicate the percentage improvement achieved by the distributed approach in terms of total time when compared to the centralized approach.

5.5.1 Comparative Evaluation of Homomorphic Cryptographic Schemes

The objective of this study is not to compare the different schemes. However, for the purpose of demonstrating the variation in the time required for homomorphic computations across different schemes, a comparison was conducted. The results, presented in Table 5.2, depict the comparison between BFV, BGV, CKKS in the centralized approach. The experiment was conducted using 50 test runs with a vector size of 2^{16} (selected to be in the higher range for a better comparison of the time values). The results indicate that BGV was the fastest in terms of total time, attributed to its efficient multiplication time. The other times of BGV are comparable to those of BFV. On the other hand, CKKS required the longest time, with a higher overhead in key generation and encryption leading to the longest time loss. In conclusion, our experimental setup demonstrates that BFV is 6.78% slower and CKKS is 103.60% slower than BGV for equivalent arithmetic operations.

Scheme	Key Generation	Encryption	Communication	Addition	Multiplication	Decryption	Total
BFV	331.41	155.48	1251.63	1.67	186.78	17.23	1944.19
BGV	333.35	157.33	1218.63	1.62	84.95	16.47	1812.34
CKKS	630.86	437.47	2484.71	3.27	99.16	34.54	3690.00

Table 5.2: Processing time in (ms) of messages with vector size of 2^{16} using the different schemes on a single resource

5.5.2 Comparing Performance: Centralized vs. Distributed Approaches

The computation times for arithmetic operations, key generation, encoding, decoding, encryption, and the total time in the BFV scheme are detailed in table 5.3. The data is organized into three categories: a centralized approach, a distributed approach with four instances, and a distributed approach with eight instances. This categorization serves to underscore the effect of augmenting computational resources on the performance of the scheme. The findings reveal that computation times escalate with the increase in vector size across all methods. Nonetheless, the distributed approach demonstrates shorter computation times when compared to the centralized approach, leading to a decrease in overall time. The BFV scheme reports a maximal enhancement of 54% in total time depending on the vector size, and this enhancement climbs to 68% when the communication time among the computational resources is excluded. Table 5.4 showcases the computation times for all operations within the BGV scheme, paralleling the structure of the preceding table. The BGV scheme's performance is on par with that of the BFV scheme, considering the vector size and the count of computational resources employed. An advancement in total computation time of up to 51% for larger vector sizes was noted in the BGV scheme, and up to 65% excluding communication time. Lastly, table 5.5 illustrates the computation times for all operations in the CKKS scheme. This scheme's performance mirrors that of the other schemes. For the CKKS scheme, an enhancement in total time of up to 48% was observed for larger vector sizes, and up to 58% without considering communication time. All three tables share the same structure and display similar trends. The upper part of each table, representing the centralized approach, serves as the baseline metric for the required times of different operations related to vector size. To illustrate how to interpret the results, let's consider an example from table 5.3. It is clear that the required times for different operations, and consequently the total time, increase with vector size. For example, the total time required for a vector size of 2^{16} in the centralized approach is nearly 164 times that required for a vector size of 2^7 . The centralized approach, representing the traditional scenario, involves the client sending a request to a single central server for data

processing. It is also evident that the most time-consuming operations include key generation, arithmetic, encryption, and communication (the time it takes for data to travel from the client to the server and back). The improvement column indicates the efficiency of the distributed approaches compared to the centralized baseline for specific vector sizes. In the distributed approach with four computing nodes (workers) for a vector size of 2^{16} , the workload is divided into four roughly equal parts and distributed among the nodes. Consequently, all time-consuming functions now require significantly less time. For instance, key generation now takes only 88.82 ms to generate keys for four smaller vectors, each of size 2^{14} , instead of 31.41 ms required for a single 2^{16} vector. The total time required in the distributed approach with four workers is 997.65 ms, compared to 1944.19 ms in the centralized approach, marking an improvement of nearly 49% a fact highlighted in the improvement column. However, the distributed approach with four workers shows disadvantages for smaller vector sizes like 2^7 , where the improvement is -22%, indicating that this approach was 22% slower than the centralized approach for a vector size of 2^7 . This can be attributed to the additional overhead of encryption and distributing the four smaller datasets to the workers, which is not cost-effective for certain data volumes. In our experiments, it is advantageous to switch from the centralized to the distributed approach for vector sizes of up to 2^9 . The lower section of the tables, representing the distributed approach with eight workers, can be interpreted similarly to the section for four workers. The data indicates a consistent trend: as data size increases, it becomes more advantageous to use a distributed approach with more computing nodes. Conversely, for small data volumes, the distributed approach is less beneficial due to the added load from key generation, encryption, and decryption for each vector.

Vector Size	Key Generation	Encryption	Communication	Arithmetic	Decryption	Total	Improvement (%)
Centralized							
2^7	3.13	0.57	7.73	0.35	0.07	11.86	-
2^8	3.78	0.85	10.23	0.63	0.09	15.58	-
2^9	4.97	1.35	15.01	1.22	0.15	22.69	-
2^{10}	7.68	2.38	24.63	2.46	0.26	37.41	-
2^{11}	13.24	4.57	43.24	5.05	0.50	66.60	-
2^{12}	23.80	8.90	90.01	10.13	0.95	133.79	-
2^{13}	44.23	17.36	194.02	21.03	1.91	278.56	-
2^{14}	87.28	35.42	309.65	44.2	3.85	480.41	-
2^{15}	169.05	73.57	584.52	90.82	8.15	926.11	-
2^{16}	331.41	155.48	1251.63	188.45	17.23	1944.19	-
Distributed with 4 Workers							
2^7	3.00	1.44	9.71	0.13	0.15	14.44	-22
2^8	2.78	1.59	11.54	0.19	0.18	16.28	-4
2^9	3.14	2.00	14.35	0.36	0.22	20.07	12
2^{10}	3.88	3.17	20.17	0.65	0.32	28.20	25
2^{11}	4.99	5.01	29.82	1.29	0.52	41.63	37
2^{12}	7.82	9.11	55.91	2.68	0.94	76.46	43
2^{13}	12.97	17.48	95.71	5.13	1.84	133.13	52
2^{14}	23.68	34.18	193.48	10.17	3.59	265.10	45
2^{15}	45.99	68.36	377.39	21.35	7.30	520.38	44
2^{16}	88.82	136.85	712.99	43.93	15.05	997.65	49
Distributed with 8 Workers							
2^7	2.93	2.59	12.2	0.09	0.26	18.07	-52
2^8	2.96	2.72	14.4	0.13	0.27	20.48	-31
2^9	2.91	3.22	18.58	0.20	0.33	25.25	-11
2^{10}	3.13	3.98	22.79	0.36	0.42	30.68	18
2^{11}	3.93	6.33	33.3	0.65	0.65	44.85	33
2^{12}	5.30	10.64	52.73	1.33	1.07	71.05	47
2^{13}	7.76	18.01	96.99	2.67	1.91	127.34	54
2^{14}	13.31	35.83	169.49	5.28	3.69	227.6	53
2^{15}	23.37	69.65	373.29	10.66	7.56	484.53	48
2^{16}	46.11	139.43	742.58	22.43	15.58	966.12	50

Table 5.3: Comparison of processing times in (ms) between centralized and distributed approach with BFV scheme

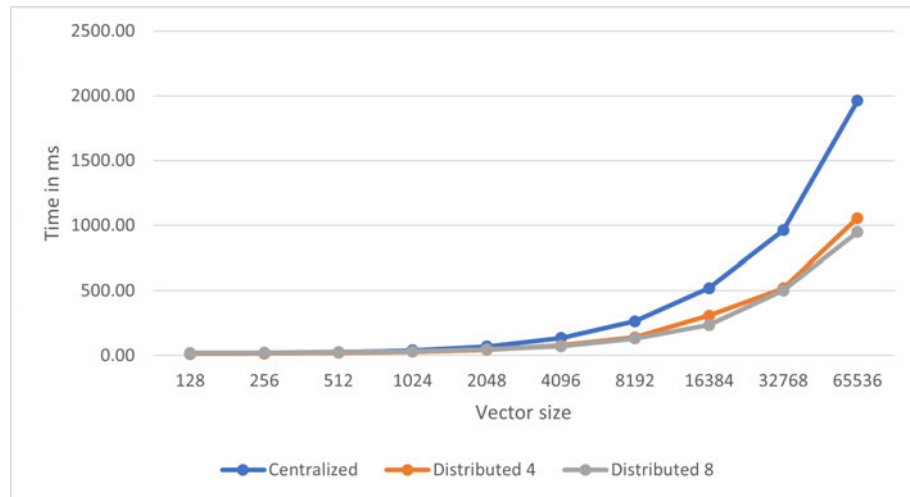
Vector Size	Key Generation	Encryption	Communication	Arithmetic	Decryption	Total	Improvement (%)
Centralized							
2^7	3.15	0.57	7.77	0.17	0.07	11.73	-
2^8	3.79	0.87	10.20	0.28	0.09	15.23	-
2^9	5.10	1.38	15.04	0.54	0.14	22.20	-
2^{10}	7.69	2.41	24.48	1.07	0.24	35.88	-
2^{11}	13.11	4.64	43.13	2.18	0.47	63.52	-
2^{12}	24.03	9.14	88.16	4.46	0.91	126.7	-
2^{13}	44.99	18.18	186.25	9.38	1.82	260.62	-
2^{14}	87.57	36.14	314.24	20.00	3.67	461.62	-
2^{15}	170.67	74.12	592.95	41.33	7.66	886.74	-
2^{16}	333.35	157.33	1218.63	86.56	16.47	1812.34	-
Distributed with 4 Workers							
2^7	2.98	1.45	9.18	0.07	0.15	13.83	-18
2^8	2.77	1.60	11.13	0.10	0.17	15.77	-4
2^9	3.10	1.99	13.73	0.17	0.21	19.20	14
2^{10}	3.81	3.11	18.89	0.32	0.30	26.43	26
2^{11}	5.03	5.08	29.34	0.57	0.50	40.52	36
2^{12}	7.78	9.21	52.73	1.07	0.90	71.70	43
2^{13}	12.82	17.37	92.72	2.22	1.73	126.86	51
2^{14}	23.77	34.50	192.10	4.48	3.42	258.27	44
2^{15}	45.24	68.69	380.40	9.48	6.93	510.73	42
2^{16}	89.34	139.72	742.75	19.95	14.34	1006.11	44
Distributed with 8 Workers							
2^7	2.91	2.62	12.32	0.05	0.25	18.14	-55
2^8	2.96	2.75	14.24	0.07	0.27	20.29	-33
2^9	2.81	3.14	17.79	0.10	0.31	24.15	-9
2^{10}	3.26	4.21	23.49	0.17	0.42	31.55	12
2^{11}	3.90	6.33	33.90	0.32	0.62	45.07	29
2^{12}	5.08	10.14	51.66	0.60	0.98	68.46	46
2^{13}	7.75	18.25	98.22	1.19	1.77	127.19	51
2^{14}	13.23	35.68	170.74	2.37	3.50	225.53	51
2^{15}	23.30	69.48	392.43	4.72	6.83	496.76	44
2^{16}	45.54	136.52	729.15	9.95	13.75	934.92	48

Table 5.4: Comparison of processing times in (ms) between centralized and distributed approach with BGV scheme

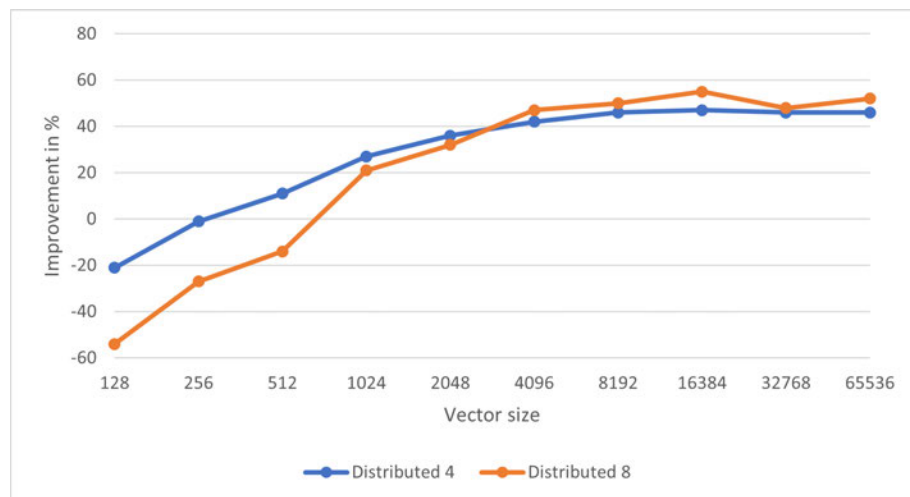
Vector Size	Key Generation	Encryption	Communication	Arithmetic	Decryption	Total	Improvement (%)
Centralized							
2^7	3.61	0.97	10.10	0.18	0.09	14.95	-
2^8	4.66	1.63	14.43	0.33	0.15	21.19	-
2^9	7.14	2.91	23.82	0.63	0.27	34.77	-
2^{10}	12.04	5.73	41.78	1.27	0.53	61.36	-
2^{11}	22.01	11.53	82.31	2.56	0.98	119.39	-
2^{12}	42.51	23.37	167.44	5.28	1.96	240.56	-
2^{13}	80.37	46.85	306.22	11.26	3.96	448.65	-
2^{14}	155.99	98.53	569.52	23.21	8.08	855.32	-
2^{15}	305.64	207.46	1211.48	48.34	16.76	1789.67	-
2^{16}	630.86	437.47	2484.71	102.43	34.54	3690.00	-
Distributed with 4 Workers							
2^7	2.71	1.75	10.29	0.07	0.17	14.99	0
2^8	3.00	2.22	13.70	0.11	0.22	19.27	9
2^9	3.62	3.58	19.02	0.20	0.33	26.74	23
2^{10}	4.69	6.14	28.40	0.34	0.54	40.12	35
2^{11}	7.31	11.42	51.83	0.66	1.00	72.23	40
2^{12}	12.32	22.34	91.22	1.27	1.87	129.02	46
2^{13}	21.64	43.29	196.51	2.55	3.69	267.68	40
2^{14}	41.79	88.30	370.48	5.35	7.43	513.34	40
2^{15}	81.23	181.18	750.58	11.17	15.18	1039.34	42
2^{16}	157.05	376.06	1419.04	23.26	31.37	2006.78	46
Distributed with 8 Workers							
2^7	2.94	3.00	14.35	0.05	0.29	20.63	-38
2^8	2.72	3.43	18.44	0.07	0.33	24.99	-18
2^9	3.09	4.65	23.03	0.11	0.43	31.30	10
2^{10}	3.76	7.44	33.21	0.22	0.66	45.30	26
2^{11}	4.84	12.51	51.40	0.39	1.10	70.24	41
2^{12}	7.30	22.58	98.15	0.70	1.99	130.71	46
2^{13}	12.43	45.66	170.76	1.38	3.95	234.19	48
2^{14}	21.92	88.40	392.42	2.80	7.70	513.25	40
2^{15}	42.11	184.91	737.99	5.58	15.65	986.23	45
2^{16}	83.99	373.96	1423.49	11.37	33.27	1926.07	48

Table 5.5: Comparison of processing times in (ms) between centralized and distributed approach with CKKS scheme

The figures below illustrate the variations among the tested approaches. The top figure displays the processing time in milliseconds (y-axis) for the three approaches: centralized (represented in blue), distributed with four workers (represented in orange), and distributed with eight workers (represented in gray), as a function of the vector size of the messages (x-axis). The bottom figure demonstrates the performance improvement of the distributed approaches (y-axis) compared to the centralized approach in terms of percentage.



(a) Comparison of the processing times

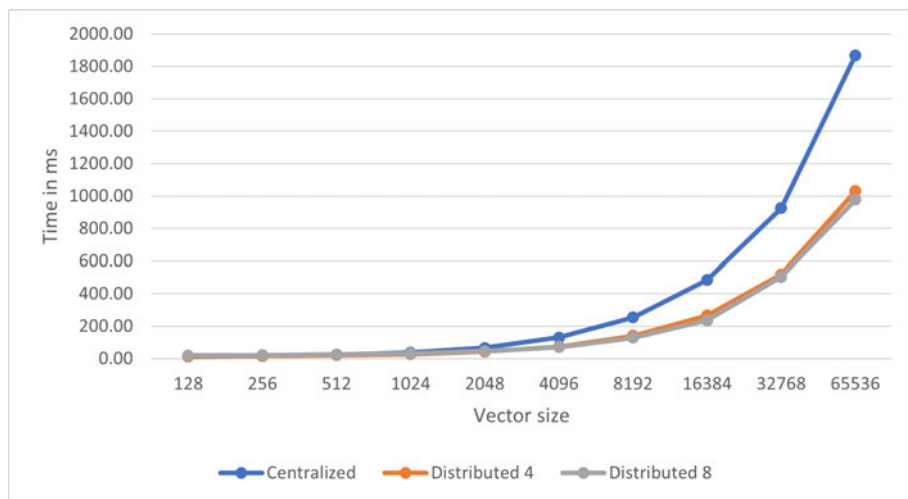


(b) Improvement of the distributed approach over the centralized approach

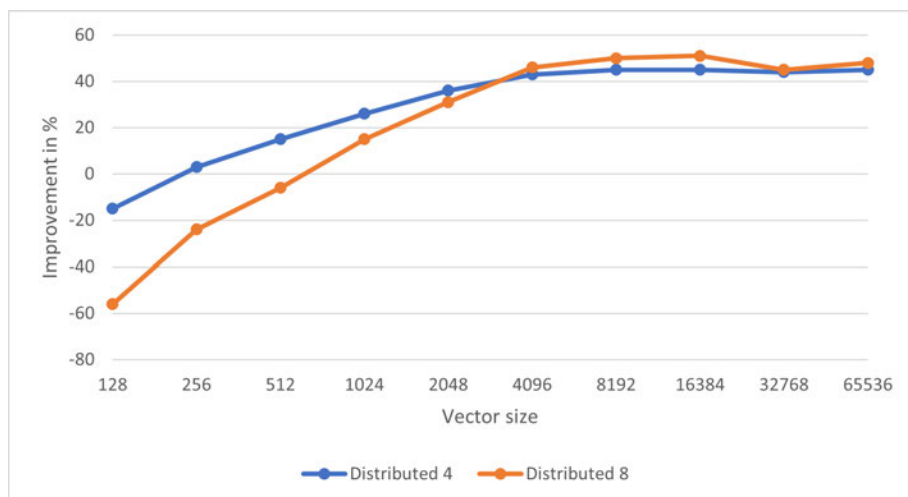
Figure 5.4: Comparison of centralized and distributed approaches at different vector sizes with BVF scheme.

The results of the BVF scheme, shown in Figure 5.4, reveal that adopting a distributed approach with four workers becomes advantageous in terms of processing time once the message length reaches 2^9 . Specifically, there's a significant reduction in processing time from 22.69ms to 20.07ms, corresponding to a 12% improvement. Furthermore, for message lengths of 2^{10} and above,

a distributed approach with eight workers outperforms the centralized method, achieving a processing time of 37.41ms compared to 30.68ms (an 18% improvement). For message lengths of 4096 and beyond, employing eight workers rather than four is advisable, as the benefits of larger message sizes outweigh the increased overhead associated with eight workers. Overall, the results suggest that up to a message size of 2^{11} , the different approaches perform similarly. However, as the message size increases, the differences between the approaches become more marked.



(a) Comparison of the processing times

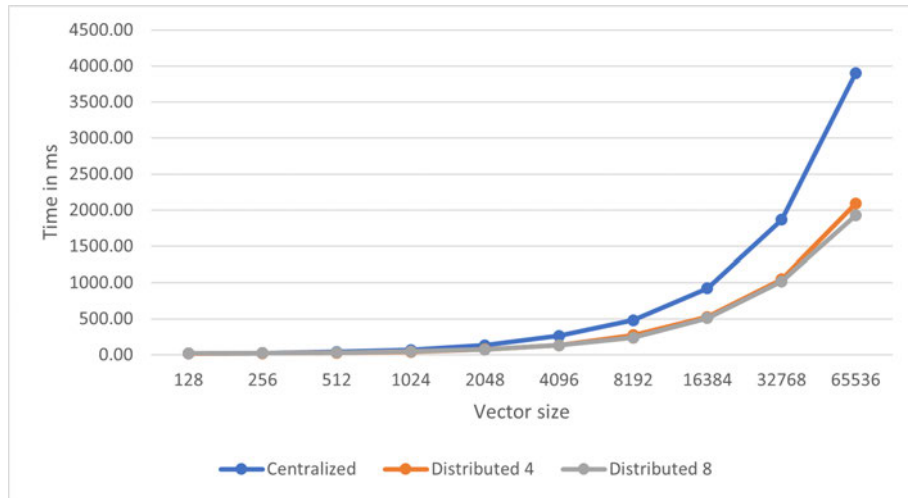


(b) Improvement of the distributed approach over the centralized approach

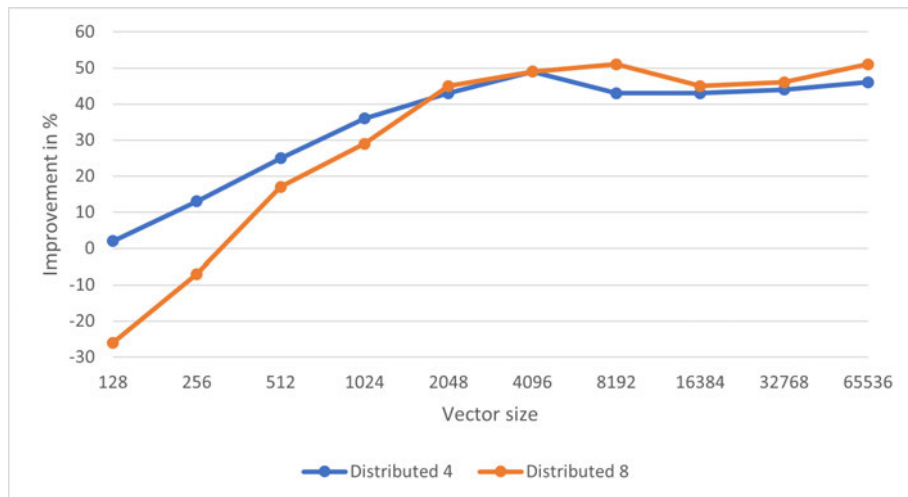
Figure 5.5: Comparison of centralized and distributed approaches at different vector sizes with BGV scheme.

The findings from the BGV scheme, depicted in Figure 5.5, show results similar to those of the BVF scheme. Regarding processing times, the distributed approach using four workers outperforms the centralized approach for a message size of 2^9 , with a processing time of 19.20ms compared to 22.20ms in the BVF scheme (a

14% enhancement). Moreover, the distributed approach with eight workers in BGV generally yields similar results to the eight-worker distributed approach in BVF, leading to a closely resembling results curve between BGV and BVF.



(a) Comparison of the processing times



(b) Improvement of the distributed approach over the centralized approach

Figure 5.6: Comparison of centralized and distributed approaches at different vector sizes with CKKS scheme.

The results presented in Figure 5.6 indicate that the CKKS scheme experiences longer message processing times compared to BGV and BVF. Nevertheless, it highlights the benefits of a distributed approach for smaller message sizes. Specifically, the distributed approach with four workers improves processing time by 9% (19.27ms vs. 21.19ms) for a message size of 2^8 . Additionally, using eight workers in the distributed approach surpasses the centralized approach's performance starting from a message size of 2^9 , achieving a 23% improvement (31.30ms vs. 34.77ms). Notably, even at a message size of 2^{11} , the distributed approach with

eight workers shows a 3% improvement over the performance achieved with four workers (70.24ms vs. 72.23ms). This trend of improvement follows the pattern observed in previous schemes but manifests at an earlier stage.

5.6 Discussion and Analysis

The analysis of processing times across different approaches—centralized, distributed with four workers, and distributed with eight workers—reveals significant insights into optimizing computational performance based on the size of the message vector. A critical observation from the data is the threshold effect in the efficacy of distributed computing. Specifically, the transition points at which distributed approaches with varying numbers of workers start to outperform the centralized method provide practical guidelines for selecting an optimal computing strategy. For example, for the BFV scheme at a message length of 2^9 , transitioning from a centralized to a distributed model with four workers yields a 12% improvement in processing times. This enhancement becomes more pronounced with message lengths of 2^{10} and above, where employing eight workers offers an 18% improvement over the centralized approach. The BGV scheme shows a similar trend; at a message length of 2^9 , shifting from a centralized to a distributed model with four workers yields a 14% improvement, and for lengths of 2^{10} and above, the distributed model with eight workers yields a 12% improvement. Notably, the CKKS scheme starts to outperform the centralized method at a message length of 2^8 , yielding a 9% improvement over the centralized model, and 23% for message lengths of 2^9 . With the CKKS scheme, the distributed model with eight workers begins to outperform the centralized method at a message length of 2^9 , yielding a 10% improvement over the centralized model, and 26% for the message length of 2^{10} . The CKKS scheme adds another layer of complexity, indicating that for smaller message sizes, the benefits of distributed computing are already evident. Such data suggest that the distributed model's superiority becomes more apparent as task complexity increases, highlighting the scalability advantage of distributed computing. For large messages, the superiority of the distributed approach is evident; for instance, for a message size of 2^{16} , the BFV scheme outperforms the centralized method with a 49% improvement utilizing the distributed model with four workers, and 50% with eight workers. The BGV

scheme outperforms the centralized method with a 44% improvement utilizing the distributed model with four workers, and 48% with eight workers. Finally, the CKKS scheme outperforms the centralized method with 46% and 48% improvement utilizing the distributed model with four and eight workers, respectively. These findings collectively underscore the importance of context in selecting an optimal computing approach. Factors such as the specific encryption scheme, message vector size, and available computing resources play crucial roles in determining whether and to what extent a distributed approach should be adopted. Additionally, the data indicates a general trend toward greater efficiency with increased distribution for larger tasks, suggesting that as computational demands grow, so too does the benefit of leveraging more distributed computing resources.

5.7 Summary

This chapter presents a novel distributed FHE approach designed to process sensitive and confidential data while ensuring information security and privacy. To evaluate the performance of the proposed approach, a test environment was set up using multiple virtual instances hosted on an external server. The CKKS, BGV, and BFV, which are widely adopted FHE schemes, were employed to perform arithmetic operations on datasets of varying sizes up to 2^{16} . Fifty test runs were conducted for each combination of the three FHE schemes and different data sizes, comparing the centralized and distributed approaches. The experimental results demonstrated significant time savings of up to 54% in the distributed approach as compared to the centralized approach. These findings indicate that the proposed distributed FHE approach holds great promise for efficiently handling large volumes of confidential data in untrustworthy public environments while maintaining security and privacy. Additionally, scalability analysis was conducted by increasing the number of virtual instances and dataset sizes. The results revealed that the distributed FHE approach exhibits high scalability and is capable of effectively managing substantial amounts of data. In conclusion, the research presented in this chapter introduces a novel distributed FHE approach for processing sensitive data with a focus on information security and privacy.

6 Shadow-Analyzer: Efficient Detection of Ghost Objects in Autonomous Driving using Neural Networks

6.1 Introduction

In recent years, EC has emerged as a pivotal technology that enhances the safety and intelligence of modern transportation systems. By enabling vehicles to communicate with each other, EC ensures the maintenance of road traffic safety and balance [43]. Moreover, EC facilitates more efficient handling of special road scenarios, such as accidents or emergency situations, through Vehicle-to-Thing communication. This capability has proven crucial in enabling effective response strategies and optimizing resource allocation in critical situations. Furthermore, EC plays a vital role in enhancing the capabilities of road cameras by enabling vehicle detection and tracking [45]. This integration empowers transportation infrastructure with intelligent monitoring and surveillance systems, augmenting overall traffic management and enabling proactive measures to ensure safety and efficiency. While EC has demonstrated its potential in advancing transportation systems, certain security concerns remain, particularly in the context of autonomous driving. LiDAR spoofing attacks, for instance, pose significant risks that must be addressed to ensure the integrity and reliability of autonomous vehicles. In this regard, offloading security-related tasks to edge nodes presents a promising approach. The primary focus of this chapter is to explore strategies

and protocols that can enhance Vehicle-to-Thing communication using EC, ultimately aiming to improve traffic safety and efficiency while effectively handling special road scenarios, including the threat of LiDAR spoofing attacks. To achieve these objectives, several key areas will be addressed. First, it is crucial to reduce the amount of data required for reliable and accurate detection. Efficient data processing at the edge ensures optimal utilization of available computing power nearby, thereby enhancing real-time response capabilities. However, it is essential to minimize the data that needs to be processed without compromising the reliability and accuracy of object detection. In this chapter, we will delve into the complexities of V2X communication and explore the role of EC in improving traffic safety and efficiency. We will examine existing strategies, identify their strengths and limitations, and propose a novel approach to address the challenges posed by LiDAR spoofing attacks. This approach involves using neural networks and object shadow verification, requiring minimal data and focusing on 2D transformed LiDAR images for predictions. As discussed in the following chapter, this method serves as a case study to demonstrate the efficiency of leveraging edge computing power to counter security threats like LiDAR spoofing attacks, particularly when combined with efficient resource allocation.

6.2 Fundamentals and Vulnerabilities of LiDAR

Sensing Technology

The LiDAR sensor operates based on the fundamental principle of measuring the travel time of optical light. This is achieved by emitting a focused laser beam of specified intensity, which reflects back to the sensor upon contact with an object. The distance d to the object can be calculated considering the speed of light $c = 299792458 \frac{m}{s}$, the refractive index of a non-vacuum medium $n = 1.000292$, and the time Δt required for the signal to return to the sensor, as illustrated in Equation 6.1.

$$d = \frac{c * \Delta t}{2n} \quad (6.1)$$

The operational methodology of LiDAR bears similarities to radar systems, with the primary difference lying in the nature of the emitted signal. The crucial determinant is the type of laser beam used. Hence, a majority of contemporary LiDAR systems predominantly operate in the near-infrared spectrum, within a range of 850 to 950 nanometers. Various methods can be employed to generate a LiDAR signal, with one of the oldest and most commonly used being the rotating LiDAR system (see Figure 6.1).

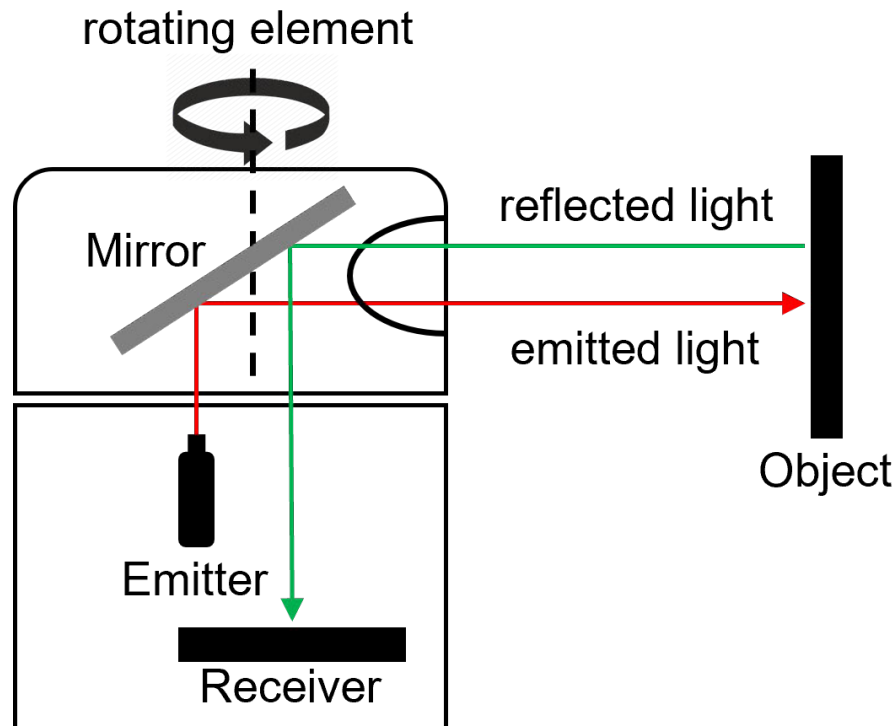


Figure 6.1: A Rotating LiDAR System for 360-Degree Environmental Detection

This technique involves directing a laser pulse towards a rotating mirror by a transmitter, which subsequently reflects the signal into the environment being scanned. The signal, upon hitting an object, is reflected back to the LiDAR sensor and directed to the receiver via the mirror, facilitating a 360-degree environmental scan around the vehicle [246]. However, a notable drawback of this setup is the formation of numerous blind spots in the immediate vicinity of the vehicle due to the sensor's roof-mounted position. Being an optical technique for environmental perception that relies on light, LiDAR sensors are susceptible to deception or manipulation with relatively minimal effort. A typical spoofing attack could be carried out by transmitting a signal at the same frequency as the LiDAR sensor. Since there is no physical interaction with the sensor, such an attack is undetectable by the system [247].

6.2.1 Feasibility of Injecting False Points in LiDAR Systems

This is corroborated by various studies that have successfully conducted attacks on LiDAR sensors under diverse conditions, yielding differing results. Petit's work [193] serves as an early significant contribution in this field. The study explored multiple attack forms on LiDAR, including jamming, relay, and spoofing attacks. The experimental setup comprised a victim LiDAR sensor, an attack transceiver, and a pair of pulse generators functioning as control logic. Upon the transmission of a signal by the LiDAR, it was intercepted by the transceiver and relayed to the control logic. The latter generated a series of delayed signals that were then returned to the transceiver. These signals were then transmitted back to the victim LiDAR with the intent of tricking it. The experiment managed to recreate a detected wall multiple times and positioned it at distances of 40, 50, and 70 m from the LiDAR sensor. Subsequently, the object detector recognized and classified these as objects. However, it is essential to highlight some limitations observed in this experimental design. For instance, the study was unable to introduce an object closer than 20 m, regardless of the signal delay. Moreover, the distance between the spoofing equipment and the victim vehicle significantly impacted the attack's quality; a larger distance resulted in many attack pulses falling outside the LiDAR's recording time window and thus, not being registered. It is also important to note that the experiments were conducted in a controlled indoor setting on a stationary object. Shin [192] addresses the aforementioned limitations, using a structure almost identical to that of Petit [193], but with the added capability of inserting points closer than the spoofer tools themselves. Under outdoor conditions, it was possible to insert 10 false points through an attack. While this may seem insignificant, these points are adequate to represent an object at a distance of 55 meters. It's crucial to note that 55 meters corresponds to the braking distance at a speed of around 100 km/h, implying a significant risk under certain conditions. Moreover, the number of insertable points could be expanded by the attacker increasing their spoofer tools. The reduced number of points inserted in comparison to Petit's method could be attributed to differences in LiDAR sensors used. Petit employed an IBEO LUX3, while Shin used a VLP-16, making a direct comparison challenging. Cao's study [191] builds upon Shin's

work [192], adopting the same attack setup, while introducing enhancements in the individual tools. The laser driver module was improved, enabling a markedly superior pulse rate of 2.304μ in comparison to the 100μ from Shin's work. This high pulse rate is comparable to that of the VLP-16 sensor. Further enhancements to the optical lens expanded the attack range beyond 5 meters. Consequently, this allowed for the insertion of up to 100 fake points starting from a 10-meter distance, although these were executed in a controlled environment. However, it was observed that at 100 points, the points occasionally fail to land at their intended locations, leading to instability. Optimal results were achieved within the range of 60 points, where points could be inserted precisely. While Cao's research [191] successfully increased the number of points an attacker could insert, certain limitations persist. These experiments, like their predecessors, were conducted in secure, proprietary environments, lacking the realistic attack performance and practical application in real-world traffic. This presents new challenges, such as the dynamic targeting of the victim's LiDAR sensor by an attacker. Sun's research [248] demonstrated the possibility of inserting as many as 200 fake points into a LiDAR system, the VLP-16 sensor. This was achieved by adapting the experimental setup of Cao [191] and enhancing it with a finely-tuned comparator circuit that bridges the photodiode and delay components, which offered a more controllable time delay. Also, by adding a COTS lens in front of the attack laser, the laser beam's refraction was improved, thereby extending the azimuth range. This research predominantly relies on black-box attacks, in contrast to previous studies that focused on white-box attacks, where the attacker has extensive knowledge about the victim system's internal processes. However, it is worth noting that this study lacks testing in actual road conditions, marking it as a potential limitation. To progress from the proof-of-concept stage, further testing is imperative. Cao's recent work [249] addresses the limitations identified in his previous research [191], extending the attack scenarios to include moving targets rather than just static or simulated scenarios. By leveraging the SSD Inception v2 COCO model, the attacker successfully located and spoofed the VLP-16 LiDAR sensor on a robot. Experiments under various conditions, including speed, distance to the spoofer, and lighting conditions, were carried out. During the

attack procedures, up to 100 fabricated data points were inserted while the robot moved at speeds reaching 0.11 m/s towards the attacker. These outcomes offer preliminary evidence regarding the viability of spoofing attacks under realistic road conditions.

6.2.2 Attack Distance Feasibility

The study conducted by Hau, Zhongyuan, et al. [201] demonstrated a defined distance beyond which an attacker could not effectively spoof an object while simultaneously imitating a genuine shadow. In order to characterize the LiDAR resolution, the researchers identified the point density within certain regions. A thorough analysis was carried out on the average point-cloud density throughout the entire KITTI dataset, as presented in Table 6.1. The findings reveal that, in close proximity (5 meters) to the test vehicle, the number of points is remarkably high, with 4858 points for a car, 1718 for a bicyclist, and 1187 for a pedestrian. As the distance from the vehicle increases, there is a marked decline in the average number of LiDAR points recorded per object. For example, at a range of 20 to 25 meters from the test vehicle, the point count for a vehicle drops to 208. Bicyclists and pedestrians, being relatively smaller objects, have fewer than 100 points, considering the attacker's capacity is restricted to 200 spoofed LiDAR points [248]. This suggests that automobiles located at a distance of 20-25 meters and pedestrians and cyclists located at a distance of 10-15 meters are likely to have an average of 200 points within their bounding box. Hence, at a distance of 10 meters or more, an attacker could potentially spoof objects that are indistinguishable from real objects in the LiDAR data. Consequently, for the purposes of this study, the simulation of spoofing attacks was focused on distances of 10 meters and beyond. This area in front of the targeted vehicle is particularly relevant in terms of potential consequential damage, as discussed in the next Section 6.3.

Distance/Avg. Points	Car	Cyclist	Pedestrian
0-5	4858	1718	1187
5-10	2040	651	455
10-15	865	263	207
15-20	405	125	99
20-25	208	78	62
25-30	117	53	42
35-40	73	37	27
40-45	51	17	29
45-50	36	12	16

Table 6.1: Relationship between Object Distance and Point Density within Bounding Boxes as Detected by LiDAR.

6.3 Attack Goal and Threat Model

In our proposed approach, we investigate a scenario wherein an attacker seeks to deceive LiDAR perception systems by generating the illusion of non-existent objects within a short distance (10–15 meters) in front of the vehicle. This range holds particular significance, especially in urban environments, due to the potentially severe consequences it poses. Should a false object suddenly appear within this proximity to an autonomously driven vehicle, it would trigger an immediate emergency braking or evasive response. Such a reaction is critical, as the calculated braking distance in an urban scenario, traveling at a speed of 50 km/h, amounts to 12.5 meters, as denoted by Equation (6.2). Consequently, an emergency braking situation induced by such a spoofing attack could create hazardous conditions for the vehicle’s occupants or other road users, as depicted in Figure 6.2. It is essential to emphasize that the reaction time is negligible in the case of an autonomously driven vehicle.

$$braking\ distance = \left(\frac{speed^2}{100}\right)/2 \quad (6.2)$$

Threat model: To achieve the attack objectives outlined previously, our approach utilizes the threat model proposed by [191], [201]. We consider an adversary with expertise in deceiving an AV's 3D object detector by spoofing LiDAR return signals. There are various scenarios in which an adversary can execute such an attack. In one scenario, the attacker can position a malicious device on the roadside, sending deceptive laser pulses to passing AVs. Alternatively, the adversary may operate an attack vehicle equipped with the malicious device, driving in front of the targeted AV or in an adjacent lane [191].

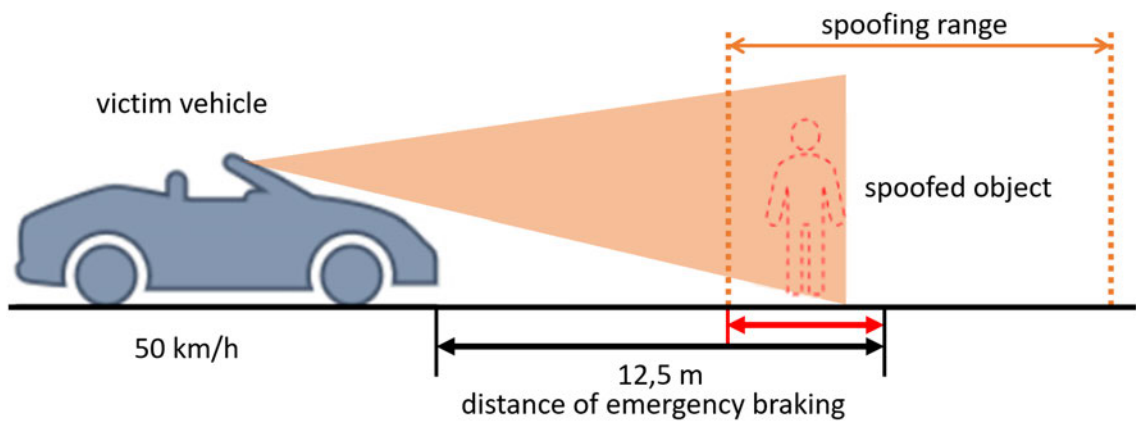


Figure 6.2: Critical Safety Zone: Importance of 10-15m Ahead of Autonomous Vehicles in Spoofing Attacks.

Our approach closely aligns with the spoofing attack method described by [248]. This attack involves three key components: a photodiode, a delay unit, and an infrared laser (see Figure 6.3). This method can be categorized as a white box attack, given the attacker's comprehensive understanding of the victim's system in this scenario. The execution of the spoofing attack proceeds as follows: first, the photodiode detects the light pulse emitted by the targeted LiDAR system. Upon detection, the diode activates a delay unit, which triggers the infrared laser after a precisely defined time. The timing is carefully chosen to coincide with the next LiDAR pulse. Given the attacker's comprehensive knowledge, the precise interval between LiDAR pulse transmissions can be ascertained. As a result of this well-timed activation, the laser beam emitted by the attacker is misinterpreted by the LiDAR system as a legitimate reflection. This leads to the successful execution of the spoofing attack, generating a deceptive point in the point cloud. The location of these inserted points within the LiDAR's field of view can be controlled

by adjusting the duration of the laser pulse delay. By employing this spoofing attack method, the adversary can deceive the AV's 3D object detector, potentially causing critical misperceptions that may compromise the vehicle's safety and performance.

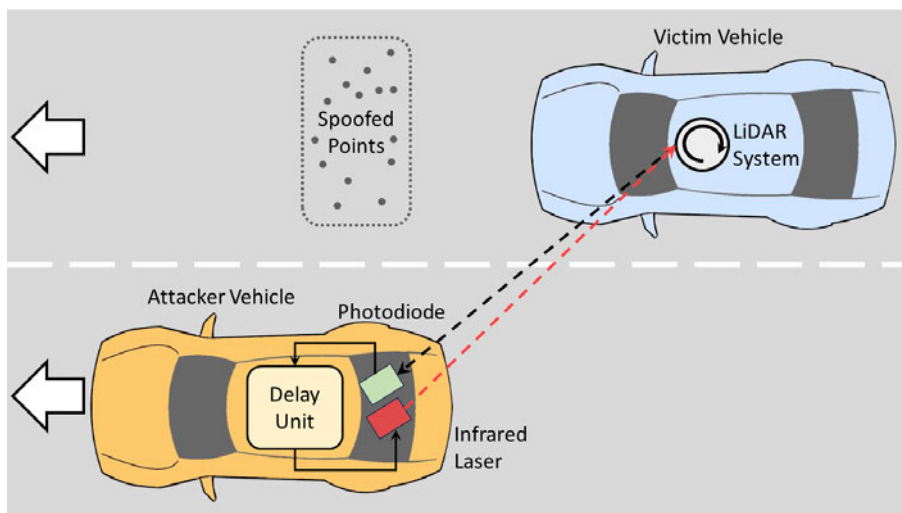


Figure 6.3: Graphical Representation of LiDAR Spoofing Technique.

6.4 2D Shadow-Analyzer approach

The LiDAR shadow verification approach proposed in this study focuses exclusively on 2D bird's-eye view images (refer to Figure 6.4). The process begins with object detection using a LiDAR sensor, which generates a 3D point cloud of the surrounding environment. A 3D object detector is then employed to identify and classify all objects within this point cloud. Each identified object is enclosed within a "bounding box", providing important information about its dimensions, such as height, width, and length. Next, a 3D point cloud representing the environment and its objects, along with their local positions and dimensions, is prepared. This information is transmitted to the Shadow-Analyzer to generate 2D images immediately. An object region proposal algorithm is utilized to select objects within a critical distance from the vehicle, creating a bird's-eye view image for each object. The key step in 2D image generation involves masking out all points that are above the respective object's height. Once the 2D images of the objects are prepared, a shadow check is performed using an artificial neural network based on Convolutional Neural Network (CNN). This pre-trained network determines whether the object has a genuine shadow (i.e., an authentic object) or if it is a

ghost attack, which refers to an object without a shadow. CNNs are inherently designed to recognize patterns directly from pixel images with minimal preprocessing, making them highly suitable for tasks that involve image or spatial data, such as the detection of shadows or the absence thereof to identify ghost objects. The choice is further justified by the system's requirements to make real-time decisions to ensure safe driving and minimize risk. By using a reduced dataset, the Shadow-Analyzer optimizes computational resources, enabling efficient processing at the edge, which is critical for real-time detection in autonomous vehicles. This efficient data processing capability of Convolutional Neural Network (CNN)s, coupled with their proven success in image recognition and classification tasks, makes them an ideal choice for the Shadow-Analyzer system, where speed and accuracy are paramount.

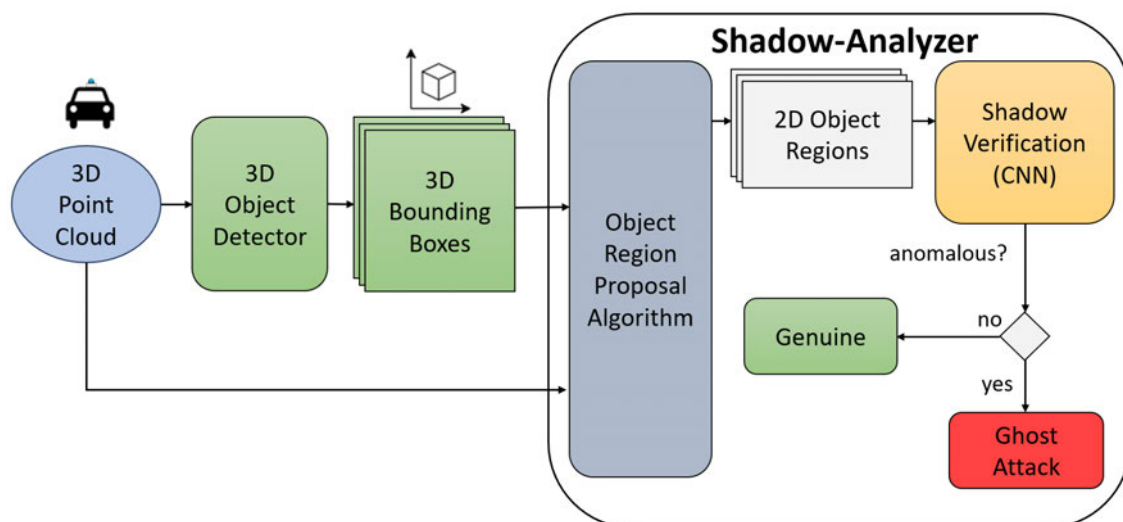


Figure 6.4: LiDAR Shadow Verification: 2D Image-Based Approach for Detecting Ghost Attacks.

6.4.1 3D Object Detector

In the domain of 3D object detection from LiDAR point clouds, the prevailing approach involves the utilization of a Graph Neural Network (GNN) known as Point-GNN. In this research, we have chosen to adopt the object detection method proposed in [250]. The selected approach effectively identifies objects within the point cloud and subsequently generates precise 3D bounding boxes for each detected object. These bounding boxes encapsulate crucial data, including the object's height, length, and width, offering comprehensive information about the objects present in the scene.

6.4.2 2D Bird's-eye View Generator

The generator for 2D images plays a crucial role as the final preprocessing step before feeding data into the artificial neural network. It takes in the 3D point cloud of the current scene, along with bounding box information, including height, width, and length dimensions of objects. Subsequently, the scene is examined to identify objects within the defined examination area, known as the Critical Low Range Distance (CLRD), which is determined using the Pythagorean theorem as shown in Equation (6.3). Subsequently, only objects within the CLRD are evaluated, while the rest are disregarded. Notably, the assessment is limited to objects located in front of the vehicle.

$$r^2 = x^2 + y^2 \quad (6.3)$$

After identifying objects within the CLRD, a bird's-eye view image is generated for each of them (see Figure 6.5).

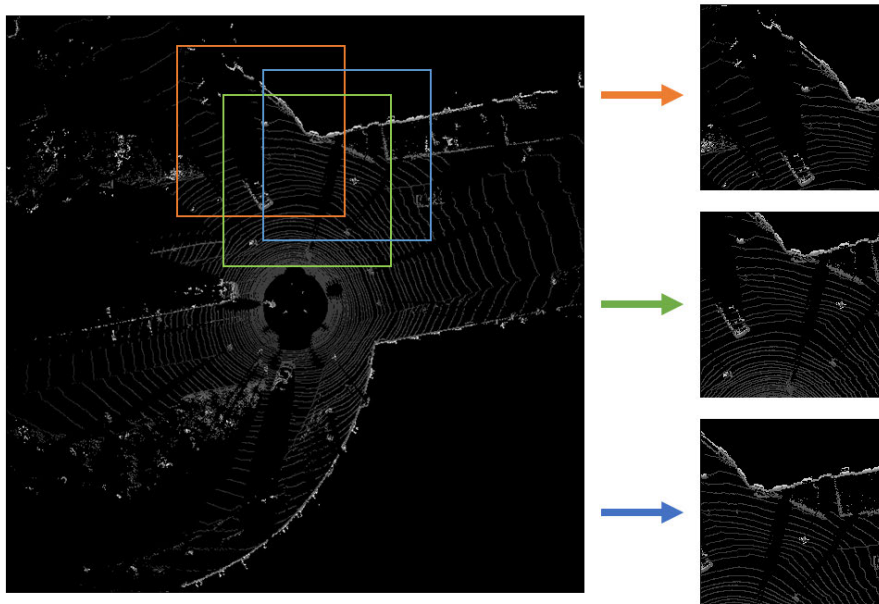


Figure 6.5: Visualization of 2D Image Generation for Scene Objects.

To prevent distortion of the LiDAR shadow, any points above the object height are concealed (see Figure 6.6). By doing so, the shadow of the object is clearly exposed and can be accurately identified in the generated 2D image.

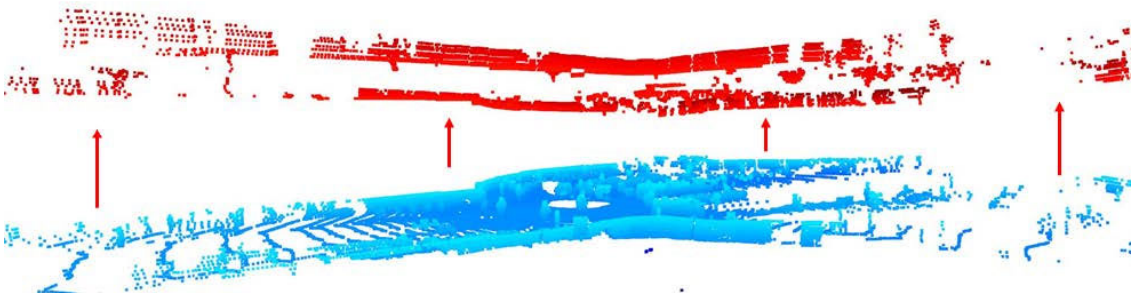
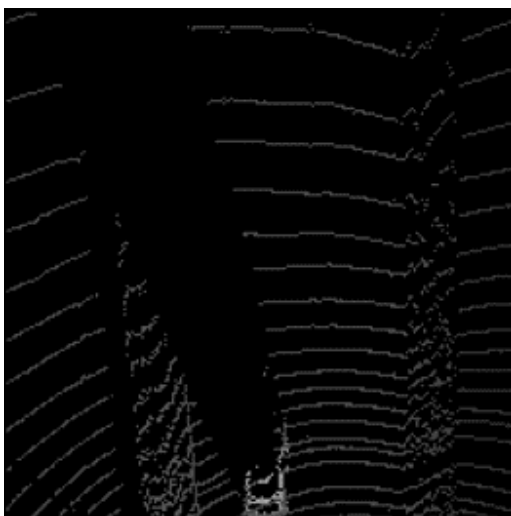


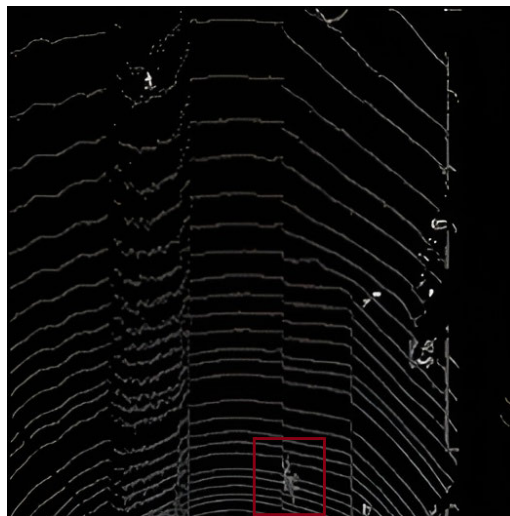
Figure 6.6: LiDAR point separation to mitigate shadow adulteration: Removing points above object height (shown in red).

6.4.3 Shadow Verification

After the image generator produces 2D images of the scene, each with dimensions of 150 x 150 pixels, these images are subjected to a classification model for verification. The purpose of this model is to perform binary classification on each image derived from the current scene, with the goal of distinguishing the presence or absence of shadows. This enables the model to identify whether each image represents a legitimate object or a deceptive attack. This approach is now feasible as all points above each object have been eliminated, thereby preserving the purity of the shadow region. As a result, the existence or absence of a shadow region corresponding to each object is evaluated (see Figure 6.7). To achieve this, a verification procedure is implemented by utilizing a CNN. The CNN is specifically designed to verify LiDAR shadow regions. The model is trained to recognize and evaluate the existence or absence of a shadow region corresponding to each object in the scene.



(a) Real object with clear shadow



(b) Spoofed Object without shadow

Figure 6.7: Comparison of 2D image generation results with genuine (a) and spoofed (b) objects.

6.5 Data Reduction

The chapter begins by emphasizing the importance of minimizing the data required for reliable and accurate detection. Efficient data processing at the edge enables optimal use of nearby computing power, leading to enhanced real-time response capabilities. However, it was highlighted that the reduction in data processing must not compromise the precision and reliability of object detection. The Point Cloud Data (PCD) format, which stores the XYZ coordinates of a 3D point cloud in ASCII format, proves highly effective for comparing volumes of point cloud data. The average data volume of a 3D point cloud from the KITTI benchmark dataset is approximately 1449.34KB (calculated from 100 scenes) in PCD format. Thanks to the multiple reductions of the point cloud implemented in our approach, the average data size for examining a single object is reduced to 215.62KB. This represents a reduction of more than 85% in the size of the original point cloud for each object. Although multiple objects can exist within a scene, the entire 3D point cloud's data volume does not need to be processed, thanks to the reduction in point cloud size (refer to Figure 6.8).

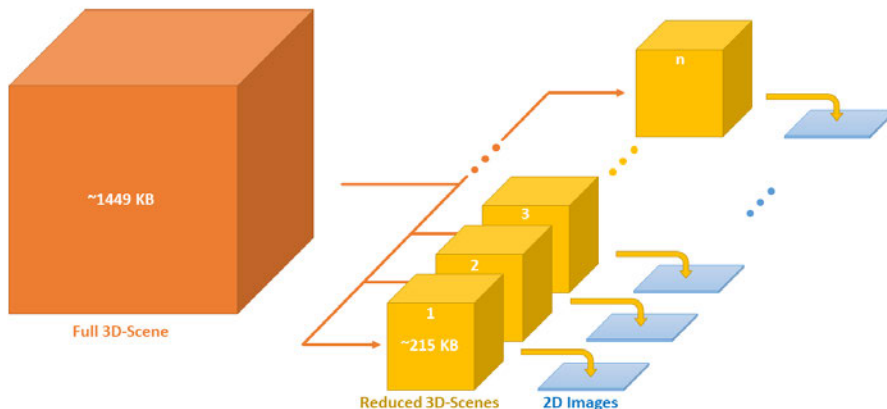


Figure 6.8: Effective Reduction of Point Cloud Data: Examining Single Objects with 85% Less Data Volume.

6.6 Shadow-Analyzer Models

In this section of the PhD thesis, we present the development of two CNN models. While both models share a common architecture in their initial layers, they demonstrate notable distinctions in their final layer configurations. The detailed

structure and unique characteristics of each model will be elaborated in their respective sections that follow. The choice of layers and convolution sizes was determined through empirical testing and optimization to achieve high accuracy and low processing time for the application in ghost object detection for autonomous vehicles.

6.6.1 CNN-Sigmoid

The model is composed of five blocks, as illustrated in Table 6.2. The first four blocks consist of alternating convolution and max-pooling operations.

Block	Layer	Activation
1	Convolution (32, (3 x 3))	ReLU
	MaxPooling (2,2)	
2	Convolution (64, (3 x 3))	ReLU
	MaxPooling (2,2)	
3	Convolution (126, (3 x 3))	ReLU
	MaxPooling (2,2)	
4	Convolution (128, (3 x 3))	ReLU
	MaxPooling (2,2)	
5	Flattening	ReLU
	Dense (512)	
	Dense (1)	Sigmoid

Table 6.2: Overview of the CNN-Sigmoid model architecture.

The fifth block includes a Flatten Layer, a Fully-Connected Dense Layer, and the Output Layer. The Flatten Layer takes the results of the feature extraction performed by the preceding layers and converts them into a one-dimensional array. Following this transformation, the data is forwarded to the Fully-Connected Dense Layer for feature analysis. The final layer, designed for the binary classification task, contains only one neuron and performs the ultimate classification to determine the presence or absence of a shadow.

Activation function: In the CNN sigmoid model, the activation function employed is the sigmoid function, as the name implies. The sigmoid function is frequently used for solving non-linearly separable problems in classification tasks. It is characterized by its domain, which includes all real numbers, and its range, which

lies within the interval $(0, 1)$ [251]. Due to this property, the sigmoid function is often referred to as a squashing function because its output is always constrained between 0 and 1, regardless of whether the input is a large positive or negative value. This behavior holds true for any input value between $-\infty$ and $+\infty$. The sigmoid function is typically applied in the final layer of the network to map the prediction probability. The specific form of the sigmoid function used in this study is provided in Equation (6.4).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (6.4)$$

Loss function: The selected loss function for this model is cross-entropy, also known as log loss. It is used to evaluate the performance of a classification model that produces output values ranging from 0 to 1. When calculating the loss, the approach considers not only the accuracy of the predicted outcomes but also the model's confidence in differentiating between a real and a non-real object. The cross-entropy function, specifically designed for binary classification, is demonstrated in Equation (6.5).

$$Loss_{binary} = -(y * \log(p) + (1 - y) * \log(1 - p)) \quad (6.5)$$

The variable y takes the value of 1 when the prediction aligns with the actual label, while in all other cases, its value is set to 0. The variable p represents the output probability of the model's prediction, which is compared to the variable y [252].

Hyperparameter: Hyperparameters are user-defined settings that cannot be determined by the model itself during a training session. These parameters are crucial as they establish the general conditions for training.

The hyperparameters employed in this study are as follows:

- The batch size, which determines the number of training samples processed before updating the internal parameters of the model.
- Steps per epoch refers to the number of batches utilized from a dataset to complete a single epoch.

- Epoch refers to the total number of complete passes through the entire training dataset.
- The learning rate, which determines the extent of adjustments in response to the error function during each update of the model weights.

The values for the CNN-Sigmoid model’s hyperparameters, as shown in Table 6.3, were determined through several fine adjustments.

Hyperparameter	Value
Epoch:	25
Steps per Epoch:	10
Batch Size:	28
Learning Rate:	0.0001

Table 6.3: Hyperparameters for the CNN-Sigmoid Model.

6.6.2 CNN-Linear

The structure of the second model is identical to that of the first model, with the only difference being the activation function used in the last block, as illustrated in Table 6.4.

Block	Layer	Activation
1	Convolution (32, (3 x 3))	ReLU
	MaxPooling (2,2)	
2	Convolution (64, (3 x 3))	ReLU
	MaxPooling (2,2)	
3	Convolution (126, (3 x 3))	ReLU
	MaxPooling (2,2)	
4	Convolution (128, (3 x 3))	ReLU
	MaxPooling (2,2)	
5	Flattening	ReLU
	Dense (512)	Linear
	Dense (1)	

Table 6.4: Overview of the CNN-Linear model architecture.

Linear activation function: The output from the dense layer is passed through, resulting in the application of a simple linear activation function, as shown in Equation (6.6):

$$f(x) = x \quad (6.6)$$

Hinge loss function: The hinge loss function is specifically designed for training classifiers in Support Vector Machines (SVMs). It penalizes predictions that fall too close to or directly on the specified boundary, assigning a value up to 1. Correctly classified predictions, located at an appropriate distance from the boundary, receive a reward with a value of 0. Conversely, incorrectly classified predictions that lie on the wrong side of the boundary are penalized with a value greater than 1, which depends on the distance to the boundary. Mathematically, the hinge loss function can be expressed as shown in Equation (6.7).

$$Loss = \max(0, 1 - y_i(x_i - b)) \quad (6.7)$$

In the equation, the value y_i corresponds to the label, representing the expected result, while x_i corresponds to the prediction made by the model in a single iteration. If a bias b is present, it is subtracted from the prediction [252].

Hyperparameter: The parameters for this model are identical to those of the CNN-Sigmoid model in terms of batch size and steps per epoch. However, there are key differences characterized by a significantly higher number of epochs and a slightly higher learning rate. The specific values are presented in Table 6.5.

Hyperparameter	Value
Epoch:	100
Steps per Epoch:	10
Batch Size:	28
Learning Rate:	0.0002

Table 6.5: Hyperparameters for the CNN-Linear model.

6.7 The experimental setup

The experimental setup depicted is utilized for both training and evaluating our proposed method.

Table 6.6: System Configuration

Host Hardware	
Processor	AMD Ryzen 9 5900X
graphic card	AMD Radeon RX 6900 XT
RAM	32 GB DDR4 (2x16)
VirtualBox	Version 7.0.10

Main Virtual Machine Used to train and test the ShadowAnalyzer	
Prozessor	12 Logical CPU Cores
RAM	16GB
OS	Ubuntu 20.04.6 LTS

Software	Version	Software	Version
Mininet	2.3.0.d6	Open vSwitch	2.13.8
Ryu	4.34	Python	3.8.16
Tensorflow	2.10.0	Numpy	1.23.3
Pandas	1.0.5	Flask	1.1.2
Flask-Restful	0.3.8	sklearn	1.1.2
Dash	2.9.3		

4xRyu Network Testing Machines Used to test remote Machines with Ryu and Mininet	
Prozessor	1 Logical CPU Cores
RAM	2GB
OS	Debian 10.13
Software	Mininet 2.3.0.d6

6.8 Training

To ensure successful training, it is imperative to use a dataset that closely mirrors real-world conditions. In this study, we utilized the KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) dataset, which consists of six hours of traffic scenarios recorded during a drive through Karlsruhe, Germany. The data were gathered using two high-resolution color and grayscale video cameras, a Velodyne laser scanner, and a GPS-equipped vehicle for localization. The KITTI dataset is renowned for its applicability to a variety of research tasks, including stereo vision, optical flow, visual odometry, 3D object detection, and 3D tracking, making it a valuable resource for advancing algorithms in environment perception for autonomous driving [253]. Our focus was on the Velodyne Lidar Data and the 3D Tracklets provided by the dataset. The data is stored as floating-point binaries. The first preprocessing step involves converting the tracklet.xml data of a scene into a CSV file. To generate realistic Shadow Images, we use this CSV BoundingBox file along with the Lidar data of the entire scene to create a 2D Bird's Eye View of each object in front of the vehicle, cropped to a 150x150 pixel file, while excluding all points above the bounding box's height. To assemble a large and well-balanced dataset, we selected multiple scenes from different environments and conditions within the KITTI dataset. To simulate realistic attacks, we extracted point clouds of actual objects, such as cars, cyclists, and pedestrians, from additional scenes within the same dataset. These are then saved as X, Y, Z coordinates in a CSV file. If necessary, the number of points saved can be adjusted to different values, for example, 60, 100, and 200. This pattern can then be injected into a different scene in the shadow to simulate a spoofed object, resulting in compromised Point Cloud Data. The result of such a modification is demonstrated in Figure 6.9. In the image, the vehicle with the LiDAR sensor is located in the central empty area. Due to its operational characteristics, the LiDAR sensor is unable to detect points directly in front of the vehicle, resulting in the characteristic empty white circle. In front of the vehicle, a cyclist can be seen as a spoofed object, inserted into the scene and therefore without a corresponding shadow. Conversely, real objects, such as the parked vehicles on the side of the road, are visible, with empty white areas behind them, which we define as shadows.

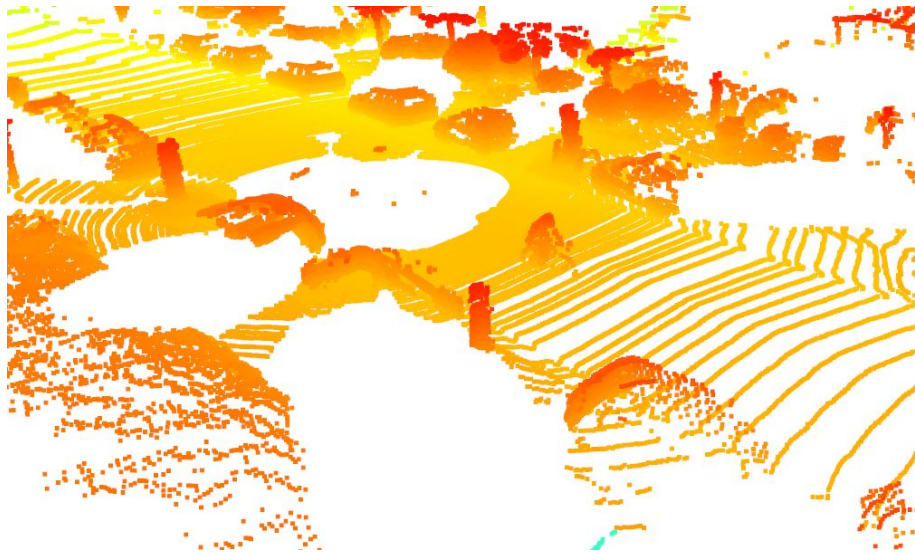


Figure 6.9: Visualizing Cyclist Class Object Insertion in KITTI Benchmark Point Cloud Dataset.

As mentioned in previous sections and supported by prior studies, the current state of the art allows for the insertion of up to 200 fake points. Therefore, we extracted target objects from datasets containing at least this number of points. Figure 6.10 provides a visualization that demonstrates the appearance of different objects with this number of points. It is evident that the number of points is more than sufficient to accurately reproduce the shapes of pedestrians (a) and cyclists (b), making them easily identifiable. While the shape of the vehicle (c) can also be discerned, the larger object requires the points to be dispersed over a greater area, resulting in an unnaturally low point density. Nonetheless, the number of points ultimately depends on the distance to the LiDAR sensor, implying that the vehicle (c) can be considered realistic at an appropriate distance.

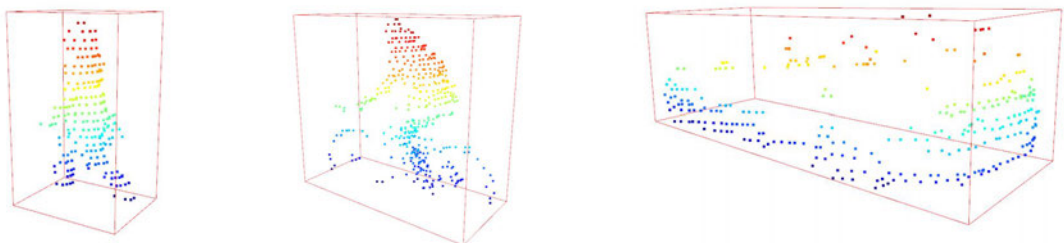


Figure 6.10: Visualization of Object Selection for LiDAR-Based Spoofing Attack using 200 Points.

Using these methods, we generated two datasets, each consisting of 400 images. The datasets were equally divided, with one half containing 200 images of real objects accompanied by shadows, and the other half featuring 200 images with artificially inserted objects without shadows. For effective evaluation, each dataset was further split into a training and a test dataset in a 70:30 ratio, resulting in 280 images for the training dataset and 120 images for the test dataset. In addition to its role in training, the test dataset was utilized for the final evaluation of the models, as elaborated in the subsequent evaluation section.

6.9 Metrics for performance measurement

In this section, we will discuss the performance metrics employed to evaluate the results presented in the subsequent section. Performance metrics are utilized to assess the effectiveness of a neural model and are conceptually similar to loss functions. Within this study, we focus on two key metrics: **Accuracy**: Is one of the most straightforward metrics used to evaluate the performance of a model. It can be calculated using the following equation (see 6.8):

$$Accuracy = \frac{\text{Number of correct Predictions}}{\text{Number of all Predictions}} \quad (6.8)$$

Essentially, accuracy measures the percentage of correct predictions made by a model. For instance, if a model yields 99 accurate predictions from a total of 100, the model's accuracy is 99%. However, when a neural network is trained on a dataset that exhibits class imbalance, this straightforward metric may not provide a reliable evaluation of the model's performance. More specifically, if data class 1 is overrepresented, the model may effectively identify it. However, if data class 2 is underrepresented, the model could struggle to accurately classify instances from this less-represented class, thereby leading to poor performance [254].

ROC-Curve: This refers to the entire area under the Receiver Operating Characteristic (ROC) curve. The construction of the ROC curve requires two values: the True Positive Rate (TPR) and the False Positive Rate (FPR):

$$TPR = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (6.9)$$

$$FPR = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}} \quad (6.10)$$

The AUC-ROC curve serves as a performance metric for classification problems under various threshold settings. The ROC itself is a probability curve, while the Area Under the Curve (AUC) represents the degree of separability between classes [255]. A higher AUC value indicates better predictive performance. An excellent model typically exhibits an AUC close to 1, implying a high degree of separability. Conversely, a poor model tends to have an AUC near 0, suggesting the worst measure of separability and, in fact, a reversal of expected results. The ROC curve is plotted with TPR against FPR, where TPR is on the y-axis and FPR is on the x-axis (see Figure 6.11).

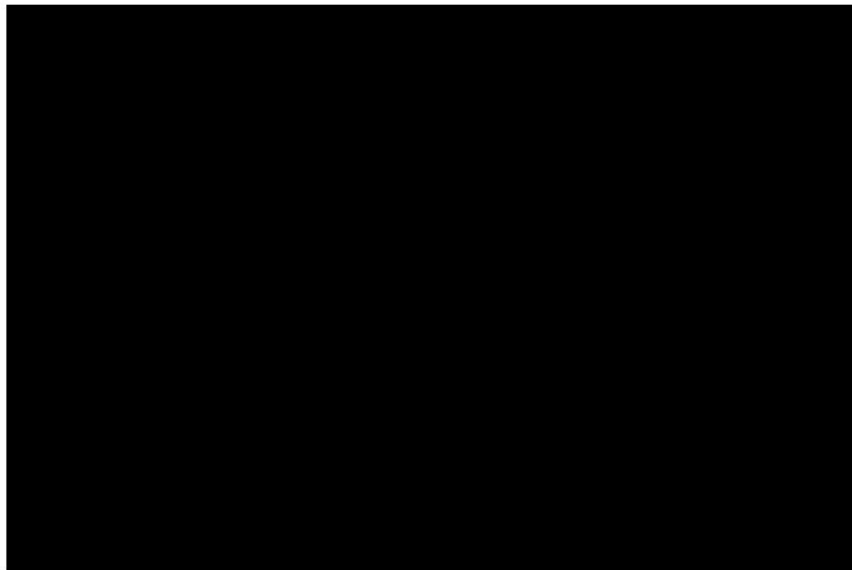


Figure 6.11: Presentation of a ROC curve (green) with an AUC score of 0.9. [256]

6.10 Evaluation

In this section, we present a comparison and evaluation of our research findings with those put forth in the pioneering work [201]. Hau, Zhongyuan, et al. proposed a novel method for verifying LiDAR objects based on their shadow regions. They

explored two approaches: one based on 2D bird's-eye views and the other focused on 3D shadow regions. However, the 2D approach was discontinued early in the research due to potential shadow contamination from larger objects situated behind the detected objects. Such objects caused points within the shadow area that could obscure or contaminate the shadow in the bird's-eye view. As a result, for the purpose of evaluation in this study, we refer to the results obtained using the 3D approach presented in [201]. Hau et al. reported accuracy and the AUC value as the performance metrics for their final model, Shadow Catcher. The achieved accuracy and AUC value were found to be 96.5% and 98.1%, respectively (refer to Table 6.7). Furthermore, their research suggests that anomaly classification can be effectively executed within a timeframe ranging from 3ms to 21ms.

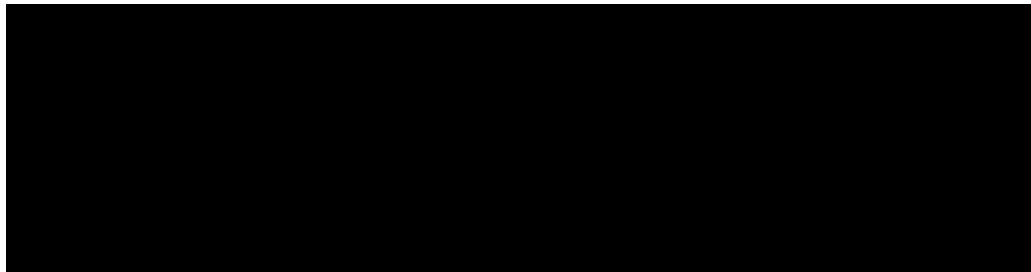


Table 6.7: Performance Evaluation of the 3D Shadow Catcher Approach: Training Results [201].

Despite the advancements made in previous work, our study addresses several unresolved challenges related to the identification of shadow regions. These challenges include objects occluded by other objects' shadows, objects positioned in front of larger objects (e.g., walls), objects situated far from the LiDAR unit resulting in lower resolution, and overlapping shadow regions formed by multiple aligned objects. To tackle these issues, we have developed and extended the 2D approach. Our novel solution for preventing contamination of an object's shadow region involves neglecting all points above the object's height that could potentially lead to contamination. By confining our method to the 2D dimension, we also strive to optimize computational resources. Furthermore, the proposed method is intended for integration within an SDN-controlled network, managed by our intelligent resource allocator. The low computing power required by our application empowers the resource allocator to deploy the application across various resources, as we discuss in the subsequent chapter. To validate the

efficacy of our approach, we developed and trained two CNN models, utilizing multiple custom datasets. To ensure unbiased evaluation, we employed test data extracted from these datasets, which were not previously encountered by the models. Subsequently, the models were tested on two additional datasets, and their results were compared with those reported by [201]. Our investigation further sought to determine whether the newly developed 2D approach could effectively address the aforementioned challenges related to shadow region identification.

6.10.1 CNN-Sigmoid

In the first model, we employed a CNN with a sigmoid activation function in the final layer. This model was evaluated using two distinct datasets: one on which the model was trained and another new dataset. The evaluation considered various metrics, including accuracy, AUC value, and the models' losses. The model produced exceptional results on both datasets, with a particularly notable performance in accuracy and AUC value. For the first dataset, the average accuracy was 98.3% and the AUC value was 99.4% (see Table 6.8). The model's performance on the second dataset was comparable, yielding an average accuracy of 96.7% and an AUC value of 99.4% (see Table 6.9).

Eval.-Nr.	Loss	Accuracy	AUC-ROC
1	0.478	0.983	0.999
2	0.554	0.983	0.998
3	0.544	0.983	0.998
4	0.544	0.983	0.998
5	0.544	0.983	0.998
6	0.544	0.983	0.998
7	0.611	0.983	0.958
8	0.490	0.983	0.999
9	0.544	0.983	0.998
10	0.544	0.983	0.998
Total Average	0.597	0.983	0.994

Table 6.8: Training results of CNN-Sigmoid model evaluated on dataset 1.

Eval.-Nr.	Loss	Accuracy	AUC-ROC
1	0.505	0.967	0.996
2	0.460	0.967	0.997
3	0.460	0.967	0.996
4	0.563	0.967	0.997
5	0.518	0.967	0.967
6	0.518	0.967	0.996
7	0.494	0.967	0.996
8	0.460	0.967	0.997
9	0.460	0.967	0.997
10	0.518	0.967	0.996
Total Average	0.496	0.994	0.994

Table 6.9: Training results of CNN-Sigmoid model evaluated on dataset 2.

However, the promising performance of the trained model is overshadowed by the high losses. The Loss for the first and second datasets was 0.597 and 0.496, corresponding to 59.7% and 49.6% respectively. These high losses indicate that the model's accurate detection may be more serendipitous than systematic. Generally, a lower loss indicates a better model, except in cases of overfitting. As the evaluations were conducted on test data rather than training data, we can exclude overfitting as a factor. The observed high losses might be attributed to several factors. For example, the simplicity of the network architecture might cause it to recalibrate completely with each run, hindering observable learning progress. Considering the high accuracy, it is plausible that the network effectively recognizes a portion of the data but fails with others. Thus, the issue could be related to binary classification. Another factor to consider is the duration of classification, which affects the model's suitability for real-time detection. The CNN model classifies data within an average time of 5.322 ms (see Table 6.10). However, despite this swift classification, the considerable losses compromise the model's reliability.

Eval.-Nr.	Prediction time (ms)	
	Dataset 1	Dataset 2
1	4.994	4.995
2	3.996	6.000
3	5.606	3.998
4	5.536	5.006
5	5.997	5.994
6	4.996	5.007
7	5.008	5.005
8	5.984	5.985
9	5.996	6.010
10	5.350	4.982
Total Average	5.346	5.298

Table 6.10: Performance analysis of CNN-Sigmoid model for classification time duration.

6.10.2 CNN-Linear

We trained and evaluated a second model, referred to as the CNN-Linear model, using the same datasets as the previous model. We conducted a detailed examination of accuracy, AUC values, and losses. The progression of the training can be precisely tracked through the provided graphs (see Figure 6.12). The losses initially decrease dramatically and later plateau, signaling the cessation of further learning progress. Accuracy displays a similar pattern, increasing rapidly at the outset before stabilizing. The AUC value, in this case, exhibits optimal behavior, closely mirroring accuracy.

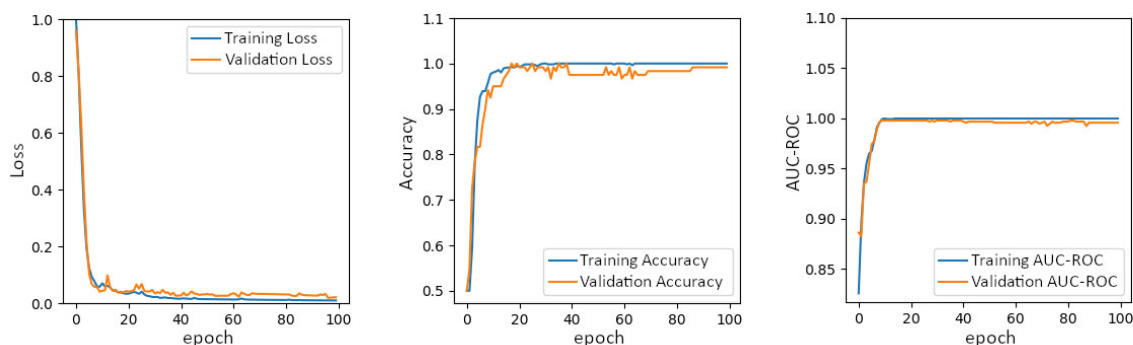


Figure 6.12: Training progress of the CNN-Linear model evaluated by the metrics.

An evaluation of the first dataset yielded an average accuracy of 97.7% and an AUC value of 99.6% (see Table 6.11). The model's low losses indicate successful learning progress during training. Notably, the average loss for this model is 8.7%, a marked improvement over the CNN model that utilizes the sigmoid function.

Eval.-Nr.	Loss	Accuracy	AUC-ROC
1	0.106	0.958	0.993
2	0.053	0.983	0.999
3	0.036	0.992	0.998
4	0.139	0.986	0.994
5	0.068	0.967	0.999
6	0.096	0.987	0.988
7	0.110	0.975	0.999
8	0.099	0.988	0.995
9	0.103	0.975	0.998
10	0.059	0.958	0.996
Total Average	0.087	0.977	0.993

Table 6.11: Training results of CNN-Linear model evaluated on dataset 1.

In the evaluation of the second dataset, the results were comparable, with an average accuracy of 97.9% and an AUC of 99.7% (see Table 6.12). The average losses recorded were 9.1%.

Eval.-Nr.	Loss	Accuracy	AUC-ROC
1	0.121	0.979	0.994
2	0.060	0.983	0.999
3	0.062	0.983	0.998
4	0.082	0.992	0.999
5	0.088	0.983	0.998
6	0.098	0.975	0.998
7	0.105	0.983	0.996
8	0.061	0.967	0.999
9	0.126	0.958	0.994
10	0.105	0.983	0.996
Total Average	0.091	0.979	0.997

Table 6.12: Training results of CNN-Linear model evaluated on dataset 2.

The outcomes achieved with this model surpass those reported in [201]. The presented results yield an accuracy of 96.5% and an AUC of 98.1%. Our technique outperforms these values with average scores of 97.8% (Accuracy) and 99.6% (AUC). Furthermore, this model employs a 2D approach, which demands significantly less computational power than the 3D approach used by Shadow Catcher [201]. Nevertheless, it is crucial to establish whether the model is fit for real-time operation. To this end, we scrutinize the classification duration. Shadow Catcher [201] required a duration ranging from 3ms to 21ms for the classification process. In contrast, the CNN-Linear model achieves an average classification duration of 5.856ms when evaluated on both datasets (see Table 6.13). This duration aligns with the optimal case reported in the previous study. As a result, from a temporal perspective, this model is well-suited for real-time applications.

Eval.-Nr.	Prediction time (ms)	
	Dataset 1	Dataset 2
1	5.993	5.008
2	6.082	5.983
3	6.994	5.994
4	4.983	5.994
5	5.550	4.987
6	6.996	4.992
7	5.719	5.997
8	4.954	6.288
9	6.995	5.002
10	6.624	5.989
Total Average	6.089	5.623

Table 6.13: Performance analysis of CNN-Linear model for classification time duration.

6.11 Invalidation Attack

The 3D approach adopted by ShadowCatcher allows for LiDAR object verification by inspecting their shadow regions. Similar to the 2D approach, it determines the presence or absence of a shadow region. However, the potential for contamination of legitimate shadow regions due to natural phenomena has led to the implementation of a distance-based point weighting system within the 3D approach. Points situated close to the object within the shadow region are allocated a higher weighting, which subsequently raises the probability of a spoofing attack. Conversely, points located at greater distances from the object within the shadow region are given lower weights, thus only marginally increasing the attack likelihood (see Fig. 6.13). This technique of shadow region verification inadvertently generates a new attack surface that can be exploited by malicious actors. In a white-box attack scenario, where the attacker has comprehensive

knowledge of the victim’s system and defense tactics, the attacker could opt to introduce false points into the shadow regions of legitimate objects rather than injecting false objects into the point cloud. This action could effectively classify these genuine objects as ghost entities within the victim’s system. Given that the newly created attack surface in ShadowCatcher permits such an invalidation attack, the 2D approach proposed in this study mitigates their feasibility by reducing the attack surface. Our method initially eliminates all points above the object’s height, thereby increasing the complexity of the attack as the spoofed points must now be situated within a more confined area.

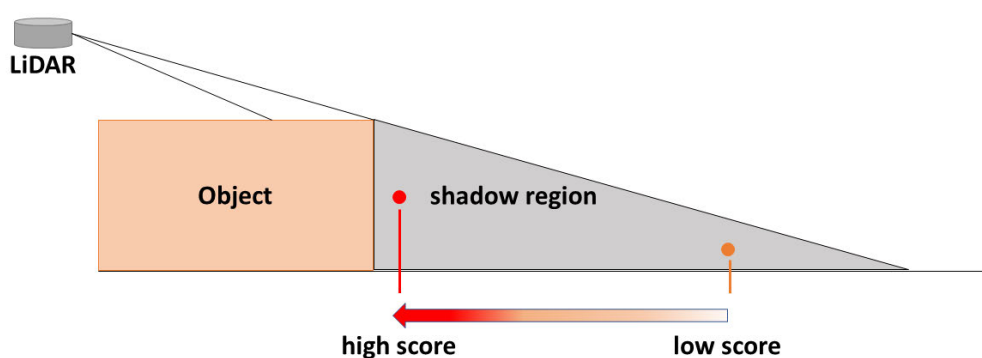


Figure 6.13: ShadowCatcher - Scoring Problem

6.12 Summary

In this chapter, we addressed the critical issue of enhancing V2T communication through the use of EC to enhance traffic safety and efficiency, with a particular focus on countering the threat of LiDAR spoofing attacks. The main objectives were to reduce data requirements while ensuring reliable and accurate detection and enabling real-time response capabilities. To achieve these goals, we first emphasized the importance of efficient data processing at the edge, which allows optimal utilization of computing power in the vicinity of vehicles. This edge-centric approach improves real-time response capabilities and is crucial for handling time-sensitive scenarios on the road. A central aspect of our approach was to minimize data processing without compromising the precision and reliability of object detection. To this end, we proposed a novel lightweight LiDAR spoofing-attack detection technique based on 2D bird’s-eye view images. Our technique,

the Shadow-Analyzer, demonstrated outstanding results, distinguishing ghost objects with an impressive average accuracy of 97.8%. Moreover, it achieved this detection within an average duration of just 6ms. These results provide strong evidence that our solution meets real-time detection requirements while significantly enhancing the performance of autonomous vehicles. The conclusion of this chapter underscores the significance of EC in V2T communication, enabling efficient data processing, and real-time detection capabilities.

7 Empirical Validation of Usability: Extensive Evaluation of Our Approach in an Edge Computing Hardware Testbed

7.1 Introduction

In the previous chapters, we explored the significant growth of high-tech industries and the groundbreaking innovations that have transformed our world. At the heart of the ongoing technological revolution is the IoT, which has led to an era of exponential data growth across various sectors. Managing these large volumes of data requires additional computational resources. As a result, we have identified the crucial roles of CC and EC as effective solutions to address the big data surge. Efficient resource management is key to unlocking the full potential of the IoT paradigm, offering a wide range of services and applications that improve our lives with unmatched flexibility and convenience. To illustrate the effectiveness and applicability of our proposed solution in chapter 3, we developed a hardware testbed consisting of devices with varying computing capabilities, reflecting the spectrum of IoT computing resources at the edge (see figure 7.1). This testbed acts as a powerful proof of concept, demonstrating how our dynamic resource allocation strategy, detailed in chapter 3, can optimally utilize available resources for task execution. Specifically, we tailored our resource allocation to support the shadow analyzer functionality discussed in Chapter 6. Through various empirical tests, we evaluated our solution's ability to efficiently use resources for

these tasks. The interaction between our resource allocation framework and the shadow analyzer application in this testbed serves as a case study, illustrating the feasibility and benefits of processing tasks at the edge with dynamic resource allocation. In the following sections of this chapter, we will elaborate on the structure and setup of our testbed and experimental procedures. We will then present and extensively analyze the results of our empirical tests, highlighting the efficiency and potential of our dynamic resource allocation method.

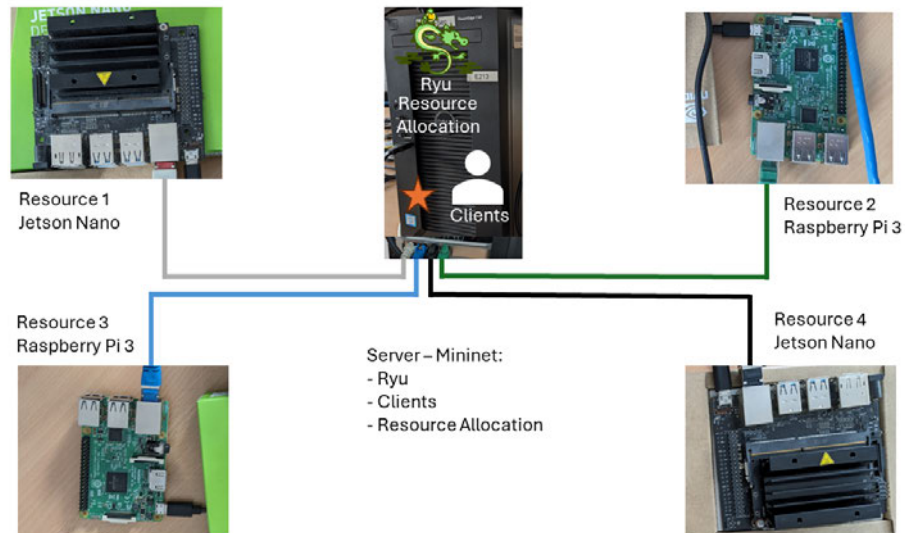


Figure 7.1: Testbed components

7.2 Testbed Setup for Resource Allocation Validation

In order to validate our resource allocation framework, we established a testbed characterized by a hybrid configuration. This testbed integrates a mininet environment, which emulates client behaviors, a virtualized SDN Controller built upon the Ryu platform, and various hardware resources interconnected with the network.

7.2.1 Hardware Resources

Our primary objective is to execute a variety of tasks on computational resources, spanning from basic arithmetic computations to complex processes such as homomorphic encryption and machine learning tasks. For a realistic representation,

we utilized two distinct computational platforms, each possessing unique capabilities. Firstly, we employed the Raspberry Pi 3 B+, a Single Board Computer (SBC) equipped with a Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4Ghz, and 1GB of RAM. Its compact size and energy efficiency are notable. This platform serves as a representation of less powerful computational resources. In contrast, the Nvidia Jetson Nano was used to signify a more robust resource. The Nano is equipped with an ARM[®] Cortex[®]-A57, complemented by a 128-core Maxwell GPU that is CUDA-compatible, and 4 GB of RAM (see figure 7.2). Remarkably, it maintains a footprint and power consumption akin to the Raspberry Pi 3 B+. NVIDIA's CUDA is an API that enables the delegation of computational tasks to the GPU. Depending on the nature of the task, a GPU, utilizing CUDA, can often accomplish tasks at multiple times the speed of traditional CPU computations. Comparative performance data illustrates this distinction. The non-optimized Shadow Analyzer, when run solely on the Raspberry Pi's CPU, requires approximately 390ms to complete a given task, as depicted in Figure 7.3. In contrast, the Jetson Nano, relying exclusively on its CUDA cores, completes the same task in less than half the time, at 190ms.

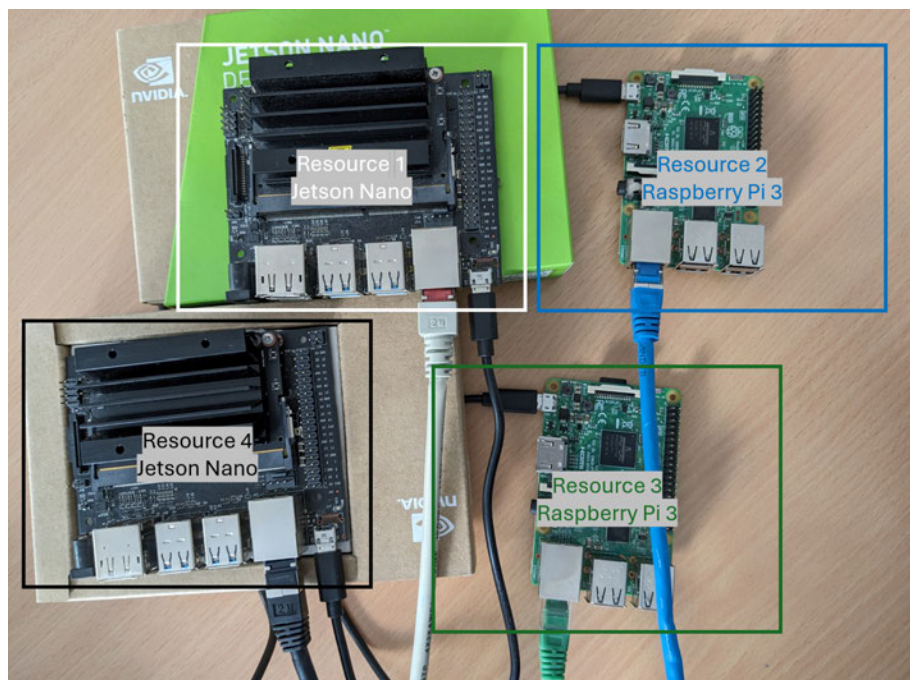


Figure 7.2: Testbed components

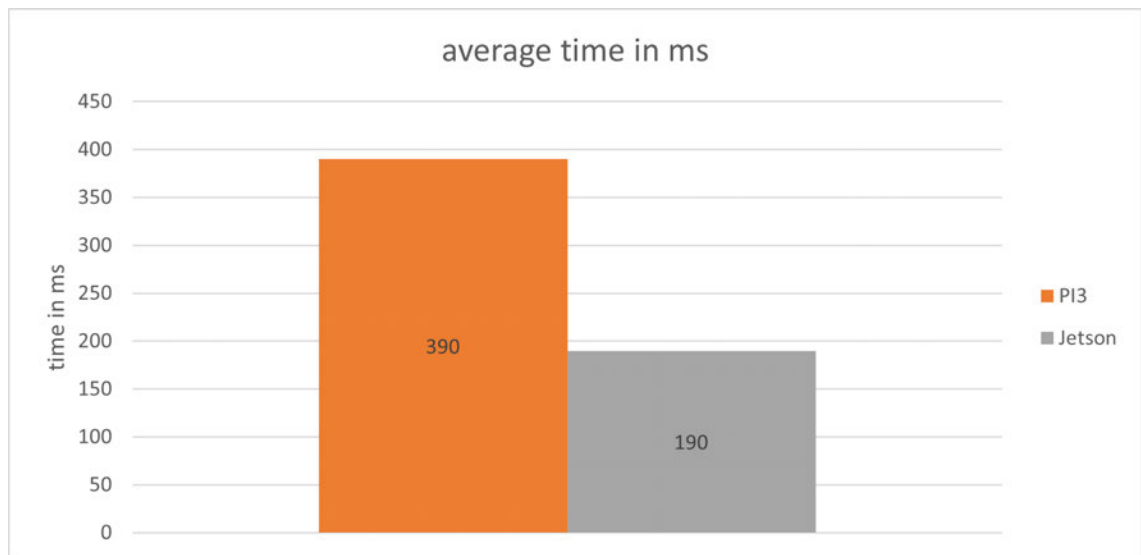


Figure 7.3: GPU vs CPU Performance.

7.2.2 Virtualized Environments

In the given virtualized environment, a comprehensive mapping of the entire network is achieved using the mininet network. This encompasses network nodes/switches, distinct network pathways, clients that exclusively make requests to hardware resources, and the interface bridging real and virtualized hardware. To this end, dedicated physical network ports are directly linked to particular virtualized switches, thereby establishing a bridge between the simulated network and actual hardware. The entire network is orchestrated by an SDN controller that is based on Ryu. Our proposed resource allocation has been incorporated into this controller. Figure 7.4 (visualized through the Mininet Topology Visualizer) illustrates a representative setup comprising two clients and four resources. The network follows a straightforward tree topology: the first layer features a centralized main switch (msw0), while the second layer is bifurcated into two switches. One of these switches connects to resources (rsw0), and the other to the virtualized clients (csw0). While there's potential for connections to virtualized resources, this hybrid configuration is specifically oriented towards hardware resources rather than their virtualized counterparts. The depicted resources are tangible, each being connected through a four-port network card, with every port being mapped to a distinct port on rsw0. In this particular scenario, Jetson Nanos serve as rs1 and rs3, while Raspberry Pi 3s are employed for rs2 and rs4.

Not illustrated in the figure is the SDN-Controller, based on Ryu. This OpenFlow Controller manages the entire network, inclusive of switches, network routes, and related components.

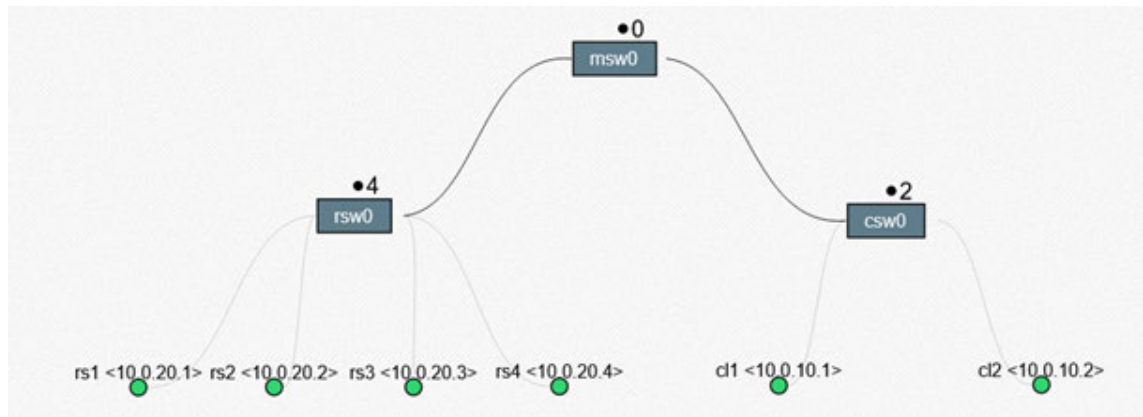


Figure 7.4: Visual Representation of a Hybrid Virtual-Physical Network with SDN Control and Direct Hardware Resource Mapping.

7.2.3 SDN-Based Resource Allocation and Client Request Flow

Figure 7.5 illustrates the flow of a client request and its subsequent response. Initially, the client issues a request, which is then directed to the resource allocation module operating on the SDN controller. Within the controller, a predictive analysis takes place to ascertain the most suitable resource to address the request. Subsequently, the request is relayed to the identified resource where it undergoes processing. The processed response is then channeled back to the resource allocation module, which subsequently forwards it to the initiating client. A notable advantage of this indirect communication method, as compared to direct communication, is the client's unawareness; it remains uninformed about specific resource parameters such as suitability, utilization, and IP address. All requisite information resides centrally within the resource allocation. Hence, it suffices for the client to direct its inquiry to the SDN controller.

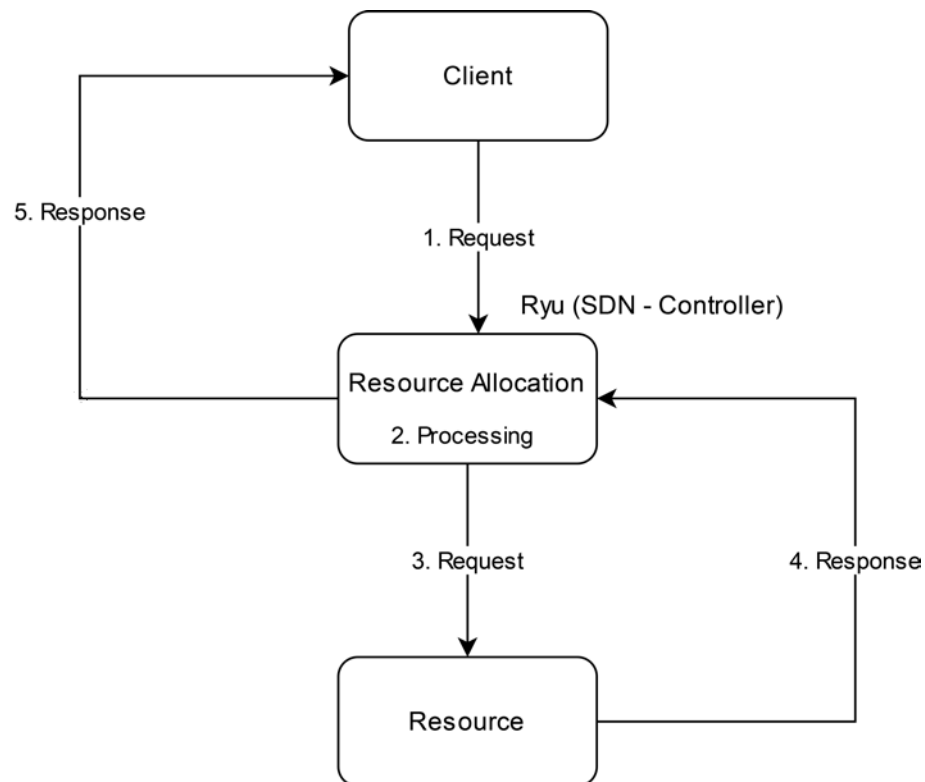


Figure 7.5: Client-Controller-Resource Workflow.

7.3 Performance Benchmarking Using a Single Local Device

In order to assess the efficacy of our resource allocation framework, we examined various configurations. The primary configuration involved the use of a single local device. This scenario is intended to represent cases where the prediction of LiDAR spoofing attacks is executed directly on the dedicated hardware of the vehicle. This configuration serves as a benchmark against which other measurements can be compared. To emulate this scenario, a Jetson was employed locally as the dedicated hardware for the task. Table 7.1 details the average prediction time for a single image. We conducted ten test runs, each encompassing 1,000 images. The average prediction time for each run was computed, with the overall mean across all runs presented in the table as the reference value.

Round	Average Time per Image in ms
1	191,792
2	190,510
3	190,554
4	191,215
5	190,108
6	190,277
7	190,050
8	189,924
9	189,786
10	190,611
Average	190,483

Table 7.1: Average Prediction Time for LiDAR Spoofing Attack Detection on Jetson Device as a Benchmark.

7.4 Remote Device Evaluation at the Edge and its Communication Delays

In the subsequent phase of our evaluation, we utilize a methodology akin to the one delineated in the preceding section. However, the distinction lies in the device's location. While the prior test considered the device to be local and on the vehicle, in this instance, the device is a remote entity situated at the edge. When dealing with a singular remote resource without the mediation of the resource allocation framework, a client dispatches requests straight to the resource. This is done without interfacing with the resource allocation, implying a potential increase in communication delays. The results presented in Table 7.2 indicate that the average time in this scenario is marginally higher than in the former case. It is crucial to underscore that these experiments were conducted under controlled laboratory conditions. Consequently, the communication time observed here might deviate from actual communication latency encountered in real-world scenarios.

Round	Average Time per Image in ms
1	190,277
2	190,050
3	189,924
4	189,786
5	191,190
6	190,611
7	190,678
8	191,297
9	191,067
10	192,141
Average	190,702

Table 7.2: Average Prediction Time for LiDAR Spoofing Attack Detection on remote Jetson Device.

7.5 Resource Allocation Between Jetson and Raspberry Pi

In this section, we examine the influence of mediation within the resource allocation framework using two distinct edge-located resources: a Jetson and a Raspberry Pi. These devices were selected to represent the diverse computational capacities available on the edge. Table 7.3 delineates the outcomes of ten iterations, each encompassing 1,000 images, consistent with the testing method in the preceding sections. Columns two and three of Table 7.3 highlight the distribution of images across the two devices in each iteration. The data indicates that the Jetson, being the more robust device, is consistently chosen by the framework over the Raspberry Pi. On average, approximately 70% of all images are directed to the Jetson. This image distribution is illustrated in Figure 7.6. This allocation strategy results in an efficient overall processing time. As shown in the fourth

column of Table 7.3, the average processing time per image improved by 11.79% compared to the reference value obtained in the initial test. The final column details the average duration taken by the resource allocation framework in each iteration to predict the optimal resource for the current task. The minimal time recorded suggests its impact on the overall processing duration is negligible.

Round	Jetson 1	Raspberry Pi	Average Time per Image in ms	Average Prediction Time in ms
1	709	291	160,823	8,949
2	688	312	183,993	8,825
3	697	303	174,706	8,413
4	676	324	161,024	8,475
5	689	311	160,879	9,223
6	709	291	160,831	8,943
7	723	277	160,804	8,874
8	804	196	161,123	9,398
9	701	299	167,923	8,845
10	793	207	188,211	9,101
Average	718,9	281,1	168,032	8,905

Table 7.3: Distribution of Images and Processing Time across 2 Edge Devices.

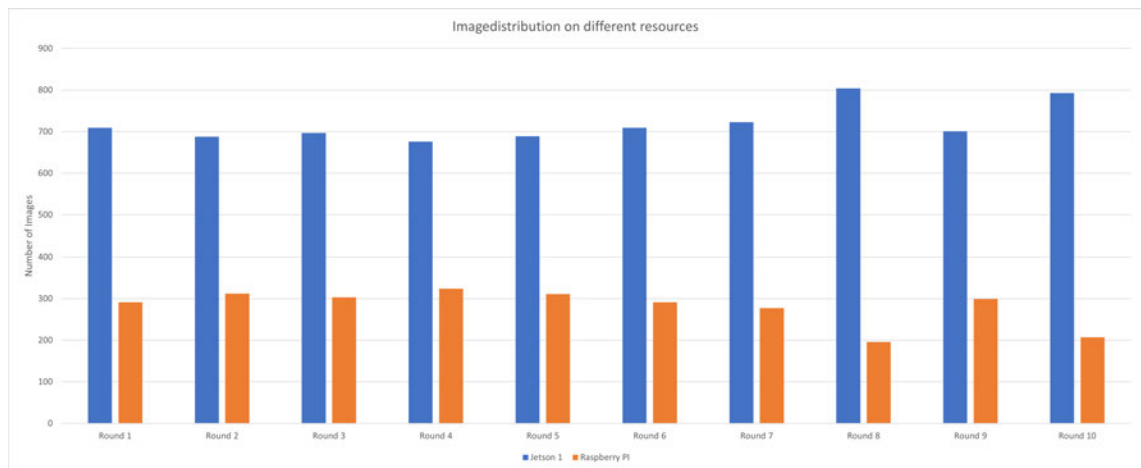


Figure 7.6: Distribution of Images Between Jetson and Raspberry Pi.

7.6 Image Distribution and Processing Time with Four Edge-located Resources

In this section, we expand upon the previous experiment by incorporating four specific edge-located resources: two Jetsons and two Raspberry Pis. The objective is to assess the impact of increasing the number of available resources on both the image distribution among these resources and the average image processing time, which consequently affects the total processing time. Table 7.4 presents the results of ten iterations, with each iteration processing 1,000 images. This procedure aligns with the testing methodologies employed in the preceding sections. Columns two through five in Table 7.4 depict the image distribution across the four devices for each iteration. Notably, the data reveals a consistent preference by the framework for the Jetsons over the Raspberry Pis, with an average of approximately 62% of all images being allocated to the Jetsons. This distribution pattern is further visualized in Figure 7.7. As indicated in the sixth column of Table 7.4, there was a 49.94% improvement in average processing time per image when compared to the benchmark set in the initial test. Furthermore, it was 43.25% more efficient than the previous experiment using only two resources. The final column of the table offers insights into the average time taken by the resource allocation framework in each iteration to determine the optimal resource for the given task.

Round	Jet. 1	RPI 1	Jet. 2	RPI 2	Avg. Time per image in ms	Avg. Prediction Time in ms
1	290	158	364	188	100,450	14,638
2	272	175	341	212	93,119	14,710
3	263	165	369	203	91,599	14,617
4	279	143	367	211	93,820	14,822
5	239	187	366	208	93,706	16,071
6	269	168	356	207	93,131	14,809
7	266	181	323	230	100,129	14,403
8	284	186	341	189	98,939	14,695
9	267	163	357	213	94,179	14,893
10	288	151	350	211	94,531	14,734
Average	271,7	167,7	353,4	207,2	95,360	14,839

Table 7.4: Distribution of Images and Processing Time across 4 Edge Devices.

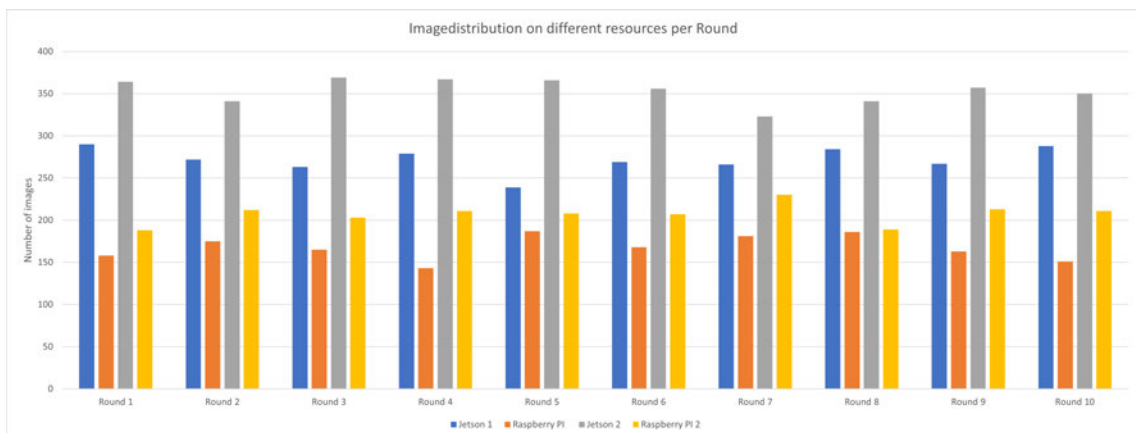


Figure 7.7: Distribution of Images Between Jetsons and Raspberry Pis.

7.7 Summary

In this chapter, we have explored the practicality of our proposed solution through a specially designed testbed. This testbed embodies devices that vary in their computing capacities, representative of the diverse IoT computing resources present at the edge. We integrated our resource allocation framework into the SDN controller, which is responsible for managing these distinct resources. Notably, this framework has been tailored to facilitate the shadow analyzer task (chapter 6). Through a series of tests involving resources of differing computing powers, the results obtained are encouraging. They suggest that our resource allocation framework is adept at efficiently allocating the available computing resources, thereby optimizing the completion of the task at hand. In conclusion, our research in this chapter has demonstrated the practicality and effectiveness of our proposed resource allocation framework in an IoT environment at the edge. By integrating this framework into the SDN controller, we have shown its ability to efficiently allocate computing resources across diverse devices. The results of our tests are promising, indicating that our framework optimizes task completion in this context.

8 Conclusions and future work

This chapter provides an overview of the core problems addressed in this thesis, along with a concise discussion of the contributions made and their associated limitations. Additionally, potential avenues for future research are highlighted to pave the way for further exploration in this field.

8.1 Thesis Summary and Objectives Review

The primary objective of this thesis is to make a significant contribution to the field of resource management, which is essential in enhancing the capabilities of the IoT. We aim to tackle the fundamental challenges of resource management within the context of EC and IoT, with the ultimate goal of enabling the effortless deployment of applications and services across these networks. To achieve this overarching aim, the following specific objectives have been outlined:

The first objective was to model dynamic resource management in EC, taking into account the heterogeneity of edge nodes. This involved analyzing current resource management strategies in EC, identifying their limitations, and recognizing the necessity for dynamic and adaptable resource management models. Additionally, the impact of task division and parallel execution on improving the performance of EC environments was examined, as detailed in Chapters 1, 2, 3, and 5. Chapters 1 and 2 provide a comprehensive background on resource management in both Cloud Computing (CC) and EC domains, offering an overview of resource management techniques, their characteristics, and limitations, with a particular emphasis on the dynamic nature of edge resources. In Chapter 3, a novel model based on NN for dynamic resource allocation is proposed and implemented, considering the diverse and fluctuating capabilities of edge nodes. The effectiveness of this model is thoroughly evaluated through experimentation and analysis. Two TensorFlow models, a classification model, and a regression

model, have been developed. Experimental results in a dynamic environment show significant improvements; the regression model achieves an 87% task completion rate within the specified timeframe, while the classification model achieves a 56% rate. Finally, Chapter 5 investigates and validates the significant role of task division and parallel execution in enhancing overall performance within EC environments. The empirical results indicate that the proposed approach leads to a significant reduction in time, up to 54%, compared to the traditional centralized approach. The second objective of the thesis was to enhance Intelligent Transportation Management Systems through the improvement of EC strategies. This involved exploring the application of EC to boost V2T communications, developing techniques to counter security threats such as LiDAR spoofing attacks using edge nodes, and validating the proposed strategies and protocols through simulation. This objective is achieved and detailed in Chapter 6, where a novel technique to counter LiDAR spoofing attacks is introduced. This method utilizes NN and object shadow verification, requiring minimal data and focusing primarily on the 2D transformed LiDAR images for its predictions. The results demonstrate that our technique can distinguish ghost objects with an average accuracy of 97.8% within an average duration of 6ms, indicating that our solution is suitable for real-time detection requirements. Chapter 7 further validates the applicability of this technique for edge applications, demonstrated through a hardware testbed that reflects the heterogeneity of the edge environment. The third objective of this thesis was to ensure security and privacy in EC systems. This involved conducting a comprehensive literature review to understand the existing security and privacy challenges in EC systems. Additionally, the thesis aimed to develop mechanisms that ensure user data privacy and secure interactions among edge nodes, particularly in sensitive applications. This goal is achieved and documented in Chapters 4 and 5. Chapter 4 introduces an innovative technique for authenticating trustworthy nodes and subsequently granting them access to edge resources through the integration of SDN and the P4 programming language. Chapter 5 presents a Distributed Homomorphic Encryption methodology, designed to enhance data privacy and security, while also ensuring efficient applicability in EC. The empirical results show that the proposed approach leads to a significant reduction in processing time, up to 54%, compared to the traditional centralized approach.

8.2 Future Directions

This thesis primarily targets three objectives: devising an efficient resource management framework, advancing intelligent transportation management systems through optimized EC strategies, and ensuring security and privacy in EC systems. Although significant strides have been made towards these objectives, myriad challenges and research directions remain to be explored. These are outlined in the subsequent subsections.

8.2.1 Secure Authentication with Switch-based One-Time

Passwords

In Chapter 4, we have explored a lightweight authentication technique implemented directly on the switch using the P4 language. This technique provides an efficient method for ensuring security during data access in EC, allowing only authorized devices to access the servers and services at the edge. In this section, we introduce a second authentication technique known as the One-Time Password (OTP). The OTP is an authentication/login system where a password is valid for only one login or transaction. However, the challenge with this approach lies in tracking the next correct password for both the switch and the client. Two potential solutions exist to circumvent this issue. The first involves using a password generator, which generates and distributes the password for the next login upon request. The second solution involves maintaining a password list on both the server and the client, an approach commonly employed in online banking through TAN-Lists. We propose employing this methodology for authenticating between the switch and the client intending to access the network. The system utilizes cryptographic hashing functions such as SHA-2, SHA-3, or BLAKE2 and is based on the Leslie Lamport algorithm [257]. The core idea is to employ a sequence of passwords x_1, x_2, \dots, x_{100} , where x_i represents the password for the client's i th identification (100 is arbitrarily chosen). Our solution involves setting the i th password x_i as $F^{100-i}(x)$, where F is a cryptographic hashing function and F^n denotes n successive applications of F . Hence, the sequence of 100 passwords is $[F^{99}(x), \dots, F(F(F(x))), F(F(x)), F(x), x]$, which is referred to as hash chains. With

this foundation, we design an algorithm to achieve OTP authentication in the data plane. The algorithm comprises several steps, as depicted in figure 8.1:

1. Generate a random initial value (seed) s .
2. Apply a cryptographic hash function f iteratively for n times to generate the hash chain, for example, $[f^{100}(s), \dots, f(f(f(s))), f(f(s)), f(s), s]$. The first $n-1$ values from the chain are stored on the client side in a table.
3. Store the value of $f^{100}(s)$ on the switch as the target value T .
4. Before a client can access the network, it must authenticate itself with the switch. For the first authentication, the client presents the first password $f^{99}(s)$.
5. On the switch side, compare the password sent from the client with the target value. In the first authentication round, this involves comparing the following values: $f(f^{99}(s)) = f^{100}(s)$. If the authentication is successful, the switch replaces $f^{100}(s)$ with $f^{99}(s)$ as its new target.
6. For subsequent successful authentications, the client needs to present $f^{98}(s)$, then $f^{97}(s)$, and so on. Authentication can be established successfully for $n-1$ times, where n is the length of the hash chain, until the chain is exhausted and a new chain must be generated.

This approach, using a cryptographic hash function, possesses the inherent property of being non-invertible. Therefore, an eavesdropper who successfully intercepts one password has no means to calculate the entire chain. Additionally, the system is resistant to Replay-Attacks, as each password is valid for one use only. However, this process is vulnerable to man-in-the-middle attacks when an attacker intercepts communication between the client and the switch, posing as the legitimate switch. Another challenge is that the client must be aware of the password currently in use, which can become problematic in the event of packet loss. This issue can be addressed by sending acknowledgments for each received password. By incorporating this approach, it can subsequently be compared with the existing technique to examine any improvements in performance efficiency.

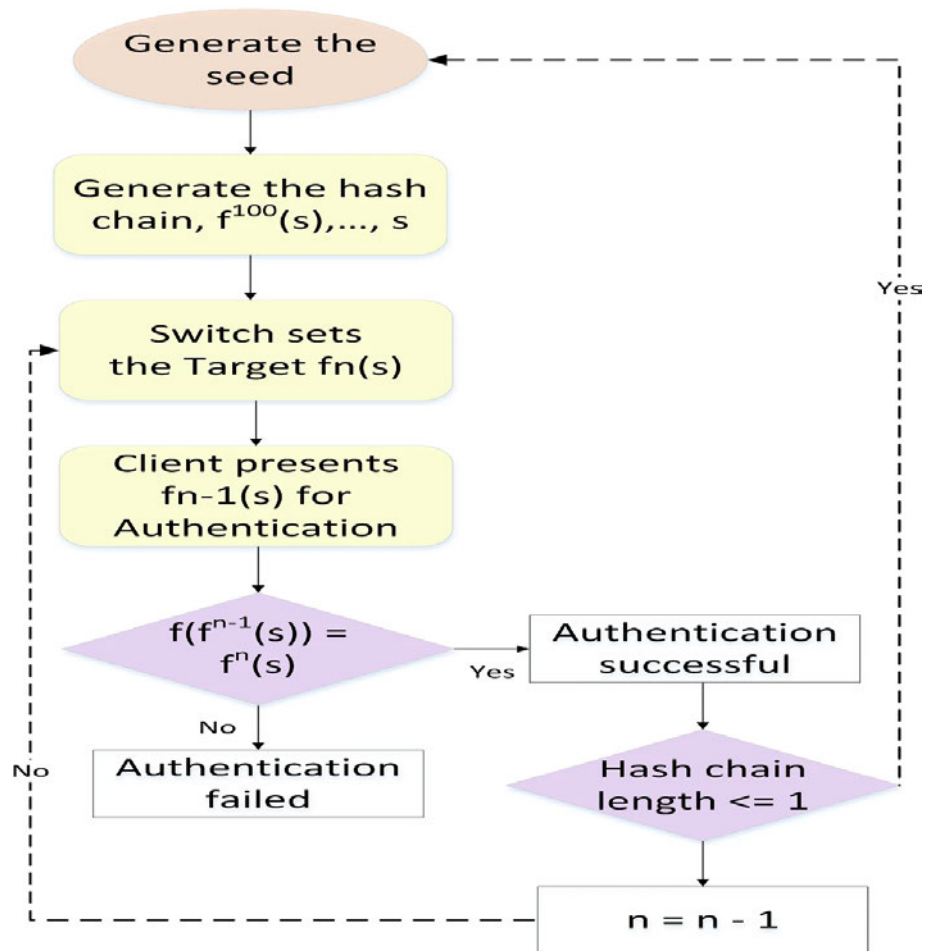


Figure 8.1: One-Time-Password Authentication Flow

8.2.2 Queue Management

In its present configuration, the AI agent exhibits limited flexibility. Its functionality is primarily confined to receiving a task and determining the appropriate resource for outsourcing. This mechanism operates on a First In – First Out (FIFO) principle. To enhance the system's effectiveness, it is suggested to introduce a queue management mechanism, enabling the AI agent to process tasks based on prioritization rather than chronological order [258], [259]. By doing so, the agent gains the capability to choose which task to process first, thereby potentially increasing efficiency. The revised system can subsequently be compared with the existing FIFO model to examine any improvements in performance efficiency.

8.2.3 Deep-Q-Network Approach

In the existing model, the AI agent perceives a given state and proceeds to make a decision, thereby receiving a reward. This sequence continues until all tasks have been accomplished. Within this framework, the subsequent state is not contingent on its predecessor, thereby implying that the AI agent is rewarded solely for short-term actions. Consequently, the long-term implications of these actions are overlooked. This model could potentially benefit from the introduction of the Deep-Q-Network (DQN) approach, an extension of the reinforcement learning methodology. By incorporating this approach, a provision for future rewards could be made, enabling a consideration of the long-term ramifications of each action [121], [260]–[263].

8.2.4 Federated Learning Applicability Evaluation

In an effort to optimize resource utilization within edge environments, it could be of significant value for AI agents to learn from each other by sharing their respective experiences. This paradigm posits that each AI agent can contribute to collaborative learning. However, traditional collaborative learning methods may lack feasibility due to privacy concerns and challenges related to network issues such as latency. In conventional scenarios, each AI agent is required to share its entire dataset with its counterpart for collaboration, thereby increasing the risk of data leakage. Federated learning can serve as an alternative, wherein each AI agent shares its knowledge with others, but instead of transmitting the entire dataset, only the updated models are transferred, thereby enhancing both efficiency and security [264]–[266].

8.3 Concluding Remarks

Enabling EC with efficient resource management techniques represents a significant stride towards realizing this paradigm. If such networks are to be widely deployed, they must effectively meet user requirements, offering reliability, scalability, security, and energy-efficiency even under challenging conditions. The findings of this research lay the groundwork for such advancement by developing an efficient resource management framework that strives to fulfill the aforementioned requirements. However, it is essential to acknowledge that the accomplishments of this research are just one step towards achieving the vision of a fully realized EC. Further research is necessary to bring this vision to fruition.

literature

- [1] L. S. Vailshery, *lot connected devices worldwide 2019-2030*, Nov. 2022. [Online]. Available: <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/> (visited on 09/07/2023).
- [2] O. Burkacky, J. Deichmann, L. Rott, and A. v. Falkenhausen, *Automotive semiconductors for the autonomous age*, Feb. 2022. [Online]. Available: <https://www.mckinsey.com/industries/industrials-and-electronics/our-insights/automotive-semiconductors-for-the-autonomous-age> (visited on 09/07/2023).
- [3] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019. doi: 10.1109/COMST.2018.2869360.
- [4] W. Rudschies and T. Kroher, *Autonomes fahren: So fahren wir in zukunft*, Mar. 2021. [Online]. Available: <https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/technik-vernetzung/aktuelle-technik/> (visited on 09/07/2023).
- [5] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *Journal of parallel and distributed computing*, vol. 79, pp. 3–15, 2015.
- [6] P. A. Abdalla and A. Varol, "Advantages to disadvantages of cloud computing for small-sized business," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, IEEE, 2019, pp. 1–6.
- [7] S. Mustafa, B. Nazir, A. Hayat, S. A. Madani, et al., "Resource management in cloud computing: Taxonomy, prospects, and challenges," *Computers & Electrical Engineering*, vol. 47, pp. 186–203, 2015.

-
- [8] S. M. Parikh, N. M. Patel, and H. B. Prajapati, *Resource management in cloud computing: Classification and taxonomy*, 2017. arXiv: 1703.00374 [cs.DC].
- [9] Gartner, *What edge computing means for infrastructure and operations leaders*. [Online]. Available: <https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders> (visited on 09/07/2023).
- [10] G. Vizzard, *5g network transforms the industrial, transportation and commercial landscape with real-time ai and accelerated response times*, Feb. 2022. [Online]. Available: <https://www.ibm.com/blog/iot-5g-transforms/> (visited on 09/07/2023).
- [11] *Edge computing market size, share & growth report, 2030*. [Online]. Available: <https://www.grandviewresearch.com/industry-analysis/edge-computing-market> (visited on 09/07/2023).
- [12] A. T. Atieh, "The next generation cloud technologies: A review on distributed cloud, fog and edge computing and their opportunities and challenges," *ResearchBerg Review of Science and Technology*, vol. 1, no. 1, pp. 1–15, 2021.
- [13] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.
- [14] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
- [15] H. D. Chantre and N. L. Saldanha da Fonseca, "The location problem for the provisioning of protected slices in nfv-based mec infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1505–1514, 2020. doi: 10.1109/JSAC.2020.2986869.
- [16] A. S. AlAhmad, H. Kahtan, Y. I. Alzoubi, O. Ali, and A. Jaradat, "Mobile cloud computing models security issues: A systematic review," *Journal of Network and Computer Applications*, vol. 190, p. 103152, 2021.

- [17] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, 2019. doi: 10.1109/ACCESS.2019.2938660.
- [18] Z. Liao, J. Peng, B. Xiong, and J. Huang, "Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–16, 2021.
- [19] X. Wang, J. Ye, and J. C. Lui, "Joint d2d collaboration and task offloading for edge computing: A mean field graph approach," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–10. doi: 10.1109/IWQOS52092.2021.9521271.
- [20] I. Kovacevic, E. Harjula, S. Glisic, B. Lorenzo, and M. Ylianttila, "Cloud and edge computation offloading for latency limited services," *IEEE Access*, vol. 9, pp. 55 764–55 776, 2021. doi: 10.1109/ACCESS.2021.3071848.
- [21] J. Long, Y. Luo, X. Zhu, E. Luo, and M. Huang, "Computation offloading through mobile vehicles in iot-edge-cloud network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–21, 2020.
- [22] M. D. Hossain, T. Sultana, M. A. Hossain, et al., "Fuzzy decision-based efficient task offloading management scheme in multi-tier mec-enabled networks," *Sensors*, vol. 21, no. 4, p. 1484, 2021.
- [23] T. Truong, Q. Fu, and C. Lorier, "Flowmap: Improving network management with sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2016, pp. 821–824.
- [24] N. McKeown, T. Anderson, H. Balakrishnan, et al., "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.
- [25] J. L. Herrera, J. Galán-Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing the response time in sdn-fog environments for time-strict iot applications," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 17 172–17 185, 2021. doi: 10.1109/JIOT.2021.3077992.

- [26] J. Liu, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Performance evaluation of integrated multi-access edge computing and fiber-wireless access networks," *IEEE Access*, vol. 6, pp. 30 269–30 279, 2018. doi: 10.1109/ACCESS.2018.2833619.
- [27] S. D. A. Shah, M. A. Gregory, S. Li, and R. D. R. Fontes, "Sdn enhanced multi-access edge computing (mec) for e2e mobility and qos management," *IEEE Access*, vol. 8, pp. 77 459–77 469, 2020. doi: 10.1109/ACCESS.2020.2990292.
- [28] P. Zhao, W. Yu, X. Yang, *et al.*, "Context-aware multi-criteria handover at the software defined network edge for service differentiation in next generation wireless networks," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2032–2046, 2022. doi: 10.1109/TSC.2020.3031181.
- [29] V. Chamola, C.-K. Tham, S. Gurunaryanan, N. Ansari, *et al.*, "An optimal delay aware task assignment scheme for wireless sdn networked edge cloudlets," *Future Generation Computer Systems*, vol. 102, pp. 862–875, 2020.
- [30] P. Thorat and N. Kumar Dubey, "Sdn-based machine learning powered alarm manager for mitigating the traffic spikes at the iot gateways," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2020, pp. 1–6. doi: 10.1109/CONECCT50063.2020.9198356.
- [31] R. Farahani, F. Tashtarian, H. Amirpour, C. Timmerer, M. Ghanbari, and H. Hellwagner, "Csdn: Cdn-aware qoe optimization in sdn-assisted http adaptive video streaming," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 525–532. doi: 10.1109/LCN52139.2021.9524970.
- [32] R. Shinkuma, Y. Yamada, T. Sato, and E. Oki, "Flow control in sdn-edge-cloud cooperation system with machine learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 1304–1309. doi: 10.1109/ICDCS47774.2020.00169.

- [33] M. Nasimi, M. A. Habibi, B. Han, and H. D. Schotten, "Edge-assisted congestion control mechanism for 5g network using software-defined networking," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5. doi: 10.1109/ISWCS.2018.8491233.
- [34] D. Zhang, F. R. Yu, R. Yang, and L. Zhu, "Software-defined vehicular networks with trust management: A deep reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1400–1414, 2022. doi: 10.1109/TITS.2020.3025684.
- [35] S. Forti, G.-L. Ferrari, and A. Brogi, "Secure cloud-edge deployments, with trust," *Future Generation Computer Systems*, vol. 102, pp. 775–788, 2020.
- [36] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE access*, vol. 8, pp. 85 714–85 728, 2020.
- [37] N. Ghodsian, *9 smart building examples; 2024 reviews*, Jan. 2024. [Online]. Available: <https://neuroject.com/smart-building-examples/> (visited on 03/26/2024).
- [38] A. Willner and V. Gowtham, "Toward a reference architecture model for industrial edge computing," *IEEE Communications Standards Magazine*, vol. 4, no. 4, pp. 42–48, 2020. doi: 10.1109/MCOMSTD.001.2000007.
- [39] F. Liu, C. Liang, and Q. He, "Remote malfunction smart meter detection in edge computing environment," *IEEE Access*, vol. 8, pp. 67 436–67 443, 2020. doi: 10.1109/ACCESS.2020.2985725.
- [40] T. Hafeez, L. Xu, and G. Mcardle, "Edge intelligence for data handling and predictive maintenance in iiot," *IEEE Access*, vol. 9, pp. 49 355–49 371, 2021. doi: 10.1109/ACCESS.2021.3069137.
- [41] *4 notable examples of smart factories exhibiting the industry 4.0 journey in southeast asia*, Sep. 2023. [Online]. Available: <https://www.abiresearch.com/blogs/2023/09/05/smart-factory-examples-southeast-asia/> (visited on 03/26/2024).

- [42] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6206–6221, 2022. doi: 10.1109/TITS.2021.3084396.
- [43] Q. Yuan, J. Li, H. Zhou, *et al.*, "Cross-domain resource orchestration for the edge-computing-enabled smart road," *IEEE Network*, vol. 34, no. 5, pp. 60–67, 2020. doi: 10.1109/MNET.011.2000007.
- [44] X. Xu, K. Liu, K. Xiao, L. Feng, Z. Wu, and S. Guo, "Vehicular fog computing enabled real-time collision warning via trajectory calibration," *Mobile Networks and Applications*, vol. 25, pp. 2482–2494, 2020.
- [45] X. Huang, P. He, A. Rangarajan, and S. Ranka, "Intelligent intersection: Two-stream convolutional networks for real-time near-accident detection in traffic video," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 6, no. 2, pp. 1–28, 2020.
- [46] A. Almaini, J. Folz, R. Boeder, *et al.*, "Shadow-analyzer an efficient neural networks-based ghost objects detection for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems [in revision]*, 2023.
- [47] *Smart roads explained*, May 2022. [Online]. Available: <https://www.nanowerk.com/smart/smart-roads-explained.php> (visited on 03/26/2024).
- [48] A. F. Subahi, "Edge-based iot medical record system: Requirements, recommendations and conceptual design," *IEEE Access*, vol. 7, pp. 94 150–94 159, 2019. doi: 10.1109/ACCESS.2019.2927958.
- [49] I. García-Magariño, J. Varela-Aldas, G. Palacios-Navarro, and J. Lloret, "Fog computing for assisting and tracking elder patients with neurodegenerative diseases," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 1225–1235, 2019.
- [50] S. Sedaghat and A. H. Jahangir, "Rt-telsurg: Real time telesurgery using sdn, fog, and cloud as infrastructures," *IEEE Access*, vol. 9, pp. 52 238–52 251, 2021. doi: 10.1109/ACCESS.2021.3069744.

- [51] Jun. 2023. [Online]. Available: <https://www.synappz.nl/?lang=en> (visited on 03/26/2024).
- [52] [Online]. Available: <https://medicus.ai/> (visited on 03/26/2024).
- [53] R. Das and M. M. Inuwa, "A review on fog computing: Issues, characteristics, challenges, and potential applications," *Telematics and Informatics Reports*, p. 100 049, 2023.
- [54] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *Journal of Communications and Networks*, vol. 24, no. 1, pp. 83–98, 2022. doi: 10.23919/JCN.2021.000041.
- [55] N. Fernando, S. W. Loke, I. Avazpour, F.-F. Chen, A. B. Abkenar, and A. Ibrahim, "Opportunistic fog for iot: Challenges and opportunities," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8897–8910, 2019. doi: 10.1109/JIOT.2019.2924182.
- [56] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017, pp. 6–9. doi: 10.1109/PERCOMW.2017.7917508.
- [57] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with vanets: A consumer perspective," *IEEE Communications Standards Magazine*, vol. 3, no. 1, pp. 19–25, 2019. doi: 10.1109/MCOMSTD.2019.1800050.
- [58] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE communications surveys & tutorials*, vol. 20, no. 1, pp. 416–464, 2017.
- [59] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *Proceedings of the first international workshop on Mobile cloud computing & networking*, 2013, pp. 19–26.

- [60] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: A multi-user case," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, IEEE, 2015, pp. 1–5.
- [61] W. Masri, I. Al Ridhawi, N. Mostafa, and P. Pourghomi, "Minimizing delay in iot systems through collaborative fog-to-fog (f2f) communication," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, 2017, pp. 1005–1010.
- [62] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–39, 2019.
- [63] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, 2020.
- [64] E. K. Markakis, K. Karras, N. Zotos, et al., "Exegesis: Extreme edge resource harvesting for a virtualized fog environment," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 173–179, 2017. DOI: 10.1109/MCOM.2017.1600730.
- [65] S. Sharma and D. Parihar, "A review on resource allocation in cloud computing," *Int. J. Adv. Res. Ideas Innovat. Technol*, vol. 1, pp. 1–7, 2014.
- [66] A. Saraswathi, Y. R. Kalaashri, and S. Padmavathi, "Dynamic resource allocation scheme in cloud computing," *Procedia Computer Science*, vol. 47, pp. 30–36, 2015.
- [67] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.
- [68] A. Belgacem, "Dynamic resource allocation in cloud computing: Analysis and taxonomies," *Computing*, vol. 104, no. 3, pp. 681–710, 2022.

- [69] C. Ghribi, M. Hadji, and D. Zeglache, "Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, IEEE, 2013, pp. 671–678.
- [70] R. Sandhu and S. K. Sood, "Scheduling of big data applications on distributed cloud based on qos parameters," *Cluster Computing*, vol. 18, pp. 817–828, 2015.
- [71] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE transactions on services Computing*, vol. 5, no. 2, pp. 164–177, 2011.
- [72] H. Zhao, J. Wang, Q. Wang, and F. Liu, "Queue-based and learning-based dynamic resources allocation for virtual streaming media server cluster of multi-version vod system," *Multimedia Tools and Applications*, vol. 78, pp. 21 827–21 852, 2019.
- [73] J. Zhang, N. Xie, X. Zhang, K. Yue, W. Li, and D. Kumar, "Machine learning based resource allocation of cloud computing in auction," *Comput. Mater. Continua*, vol. 56, no. 1, pp. 123–135, 2018.
- [74] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, "Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 10, pp. 1127–1139, 2020.
- [75] V. Ramasamy and S. Thalavai Pillai, "An effective hpso-mga optimization algorithm for dynamic resource allocation in cloud environment," *Cluster Computing*, vol. 23, pp. 1711–1724, 2020.
- [76] Z. Chen, L. Yang, Y. Huang, X. Chen, X. Zheng, and C. Rong, "Pso-ga-based resource allocation strategy for cloud-based software services with workload-time windows," *IEEE Access*, vol. 8, pp. 151 500–151 510, 2020.
- [77] E. N. Alkhanak, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost optimization approaches for scientific workflow scheduling in cloud and grid comput-

- ing: A review, classifications, and open issues," *Journal of Systems and Software*, vol. 113, pp. 1–26, 2016.
- [78] S. Challita, F. Paraiso, and P. Merle, "A study of virtual machine placement optimization in data centers," Apr. 2017. doi: 10.5220/0006236503430350.
- [79] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian informatics journal*, vol. 16, no. 3, pp. 275–295, 2015.
- [80] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *Journal of Network and Computer Applications*, vol. 66, pp. 64–82, 2016.
- [81] P. Salot, "A survey of various scheduling algorithm in cloud computing environment," *International Journal of Research in Engineering and Technology*, vol. 2, no. 2, pp. 131–135, 2013.
- [82] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021. doi: 10.1109/COMST.2021.3106401.
- [83] A. Kiani, *From geographically dispersed data centers towards hierarchical edge computing*. New Jersey Institute of Technology, 2018.
- [84] H. Badri, *Stochastic optimization methods for resource management in edge computing systems*. Wayne State University, 2019.
- [85] J. Lim and D. Lee, "A load balancing algorithm for mobile devices in edge cloud computing environments," *Electronics*, vol. 9, no. 4, p. 686, 2020.
- [86] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018. doi: 10.1109/JIOT.2018.2826006.
- [87] A. Madej, N. Wang, N. Athanasopoulos, R. Ranjan, and B. Varghese, "Priority-based fair scheduling in edge computing," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, 2020, pp. 39–48. doi: 10.1109/ICFEC50348.2020.00012.

- [88] H. Sun, H. Yu, G. Fan, and L. Chen, "Qos-aware task placement with fault-tolerance in the edge-cloud," *IEEE Access*, vol. 8, pp. 77 987–78 003, 2020. doi: 10.1109/ACCESS.2020.2977089.
- [89] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017. doi: 10.1109/TCC.2015.2449834.
- [90] T. Lähderanta, T. Leppänen, L. Ruha, et al., "Edge computing server placement with capacitated location allocation," *Journal of Parallel and Distributed Computing*, vol. 153, pp. 130–149, 2021. doi: <https://doi.org/10.1016/j.jpdc.2021.03.007>.
- [91] Q. Vo and D. A. Tran, "Probabilistic partitioning for edge server assignment with time-varying workload," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–8. doi: 10.1109/ICCCN.2019.8846932.
- [92] A. Santoyo González and C. Cervelló Pastor, "Edge computing node placement in 5g networks: A latency and reliability constrained framework," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2019, pp. 183–189. doi: 10.1109/CSCloud/EdgeCom.2019.00024.
- [93] B. Li, Q. He, G. Cui, et al., "Read: Robustness-oriented edge application deployment in edge computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1746–1759, 2022. doi: 10.1109/TSC.2020.3015316.
- [94] H. M. Makrani, H. Sayadi, N. Nazari, et al., "Adaptive performance modeling of data-intensive workloads for resource provisioning in virtualized environment," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 5, no. 4, pp. 1–24, 2021. doi: 10.1145/3442696. [Online]. Available: <https://doi.org/10.1145/3442696>.

- [95] Z. Zhou, S. Yu, W. Chen, and X. Chen, "Ce-iot: Cost-effective cloud-edge resource provisioning for heterogeneous iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8600–8614, 2020. doi: 10.1109/JIOT.2020.2994308.
- [96] X. Xu, Z. Fang, L. Qi, X. Zhang, Q. He, and X. Zhou, "Tripres: Traffic flow prediction driven resource reservation for multimedia iov with edge computing," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 2, pp. 1–21, 2021.
- [97] J. Liu, K. Luo, Z. Zhou, and X. Chen, "Erp: Edge resource pooling for data stream mobile computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4355–4368, 2019. doi: 10.1109/JIOT.2018.2882588.
- [98] A. Qasem, P. Shirani, M. Debbabi, L. Wang, B. Lebel, and B. L. Agba, "Automatic vulnerability detection in embedded devices and firmware: Survey and layered taxonomies," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–42, 2021.
- [99] R. Smith, D. Palin, P. P. Ioulianos, V. G. Vassilakis, and S. F. Shahandashti, "Battery draining attacks against edge computing nodes in iot networks," *Cyber-Physical Systems*, vol. 6, no. 2, pp. 96–116, 2020.
- [100] K. Matsui and H. Nishi, "Error correction method considering fog and edge computing environment," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 2019, pp. 517–521. doi: 10.1109/ICPHYS.2019.8780317.
- [101] W. Tong, B. Jiang, F. Xu, Q. Li, and S. Zhong, "Privacy-preserving data integrity verification in mobile edge computing," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1007–1018. doi: 10.1109/ICDCS.2019.00104.
- [102] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Inspecting edge data integrity with aggregate signature in distributed edge computing environment," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2691–2703, 2022. doi: 10.1109/TCC.2021.3059448.

- [103] H. Cui, X. Yi, and S. Nepal, "Achieving scalable access control over encrypted data for edge computing networks," *IEEE Access*, vol. 6, pp. 30 049–30 059, 2018. doi: 10.1109/ACCESS.2018.2844373.
- [104] A. Almaini, A. Al-Dubai, I. Romdhani, and M. Schramm, "Delegation of authentication to the data plane in software-defined networks," in *2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*, IEEE, 2019, pp. 58–65.
- [105] A. Almaini, A. Al-Dubai, I. Romdhani, M. Schramm, and A. Alsarhan, "Light-weight edge authentication for software defined networks," *Computing*, vol. 103, no. 2, pp. 291–311, 2021.
- [106] L. Ma, Q. Pei, L. Zhou, H. Zhu, L. Wang, and Y. Ji, "Federated data cleaning: Collaborative and privacy-preserving data cleaning for edge intelligence," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6757–6770, 2021. doi: 10.1109/JIOT.2020.3027980.
- [107] A. Almaini, J. Folz, D. Woelfl, A. Al-Dubai, M. Schramm, and M. Heigl, "A new scalable distributed homomorphic encryption scheme for high computational complexity models," in *2023 International Wireless Communications and Mobile Computing (IWCMC)*, 2023, pp. 890–897. doi: 10.1109/IWCMC58020.2023.10183131.
- [108] Y. Wang and T. Nakachi, "Secure face recognition in edge and cloud networks: From the ensemble learning perspective," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2393–2397. doi: 10.1109/ICASSP40776.2020.9052992.
- [109] H. Zhong, Y. Zhou, Q. Zhang, Y. Xu, and J. Cui, "An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare," *Future Generation Computer Systems*, vol. 115, pp. 486–496, 2021.

- [110] L. Jiang, R. Tan, X. Lou, and G. Lin, "On lightweight privacy-preserving collaborative learning for internet-of-things objects," in *Proceedings of the international conference on internet of things design and implementation*, 2019, pp. 70–81.
- [111] Y. S. Can and C. Ersoy, "Privacy-preserving federated deep learning for wearable iot-based biomedical monitoring," *ACM Trans. Internet Technol.*, vol. 21, no. 1, Jan. 2021, ISSN: 1533-5399. doi: 10.1145/3428152. [Online]. Available: <https://doi.org/10.1145/3428152>.
- [112] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6. doi: 10.1109/ICC.2018.8422743.
- [113] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, 2019. doi: <https://doi.org/10.1016/j.dcan.2018.10.003>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352864818301469>.
- [114] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in iot networks via reinforcement learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761385.
- [115] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in iot networks with edge computing," *China Communications*, vol. 17, no. 9, pp. 220–236, 2020. doi: 10.23919/JCC.2020.09.017.
- [116] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in iot networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020. doi: 10.1109/JIOT.2020.2970110.

- [117] N. Kiran, C. Pan, and Y. Changchuan, "Reinforcement learning for task offloading in mobile edge computing for sdn based wireless networks," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, 2020, pp. 268–273. doi: 10.1109/SDS49854.2020.9143888.
- [118] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1529–1541, 2021. doi: 10.1109/TETC.2019.2902661.
- [119] S. Goudarzi, M. H. Anisi, H. Ahmadi, and L. Musavian, "Dynamic resource allocation model for distribution operations using sdn," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 976–988, 2021. doi: 10.1109/JIOT.2020.3010700.
- [120] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 1, pp. 31–42, 2018. doi: 10.1109/TPDS.2017.2748578.
- [121] O. Urmonov, H. Aliev, and H. Kim, "Multi-agent deep reinforcement learning for enhancement of distributed resource allocation in vehicular network," *IEEE Systems Journal*, vol. 17, no. 1, pp. 491–502, 2023. doi: 10.1109/JSYST.2022.3197880.
- [122] A. S. Kumar, L. Zhao, and X. Fernando, "Multi-agent deep reinforcement learning-empowered channel allocation in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1726–1736, 2022. doi: 10.1109/TVT.2021.3134272.
- [123] K. Lu, Z. Du, J. Li, and G. Min, "Resource-efficient distributed deep neural networks empowered by intelligent software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4069–4081, 2022. doi: 10.1109/TNSM.2022.3218173.
- [124] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Communications Surveys & Tutorials*, 2022.

- [125] C. Mechalikh, H. Taktak, and F. Moussa, "A fuzzy decision tree based tasks orchestration algorithm for edge computing environments," in *Advanced Information Networking and Applications: Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)*, Springer, 2020, pp. 193–203.
- [126] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Network*, vol. 34, no. 2, pp. 128–134, 2019.
- [127] S. Imtiaz, H. Ghauch, G. P. Koudouridis, and J. Gross, "Random forests resource allocation for 5g systems: Performance and robustness study," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2018, pp. 326–331.
- [128] M. E. Mavroforakis and S. Theodoridis, "A geometric approach to support vector machine (svm) classification," *IEEE transactions on neural networks*, vol. 17, no. 3, pp. 671–682, 2006.
- [129] S. Wang, M. Chen, C. Yin, *et al.*, "Federated learning for task and resource allocation in wireless high-altitude balloon networks," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 460–17 475, 2021.
- [130] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 361–371, 2020.
- [131] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "Ql-heft: A novel machine learning scheduling scheme base on cloud computing environment," *Neural Computing and Applications*, vol. 32, pp. 5553–5570, 2020.
- [132] Z. Tong, Z. Xiao, K. Li, and K. Li, "Proactive scheduling in distributed computing—a reinforcement learning approach," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2662–2672, 2014.
- [133] T.-P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using a two-stage machine learning approach," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 256–268, 2017.

- [134] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.
- [135] P. Yu, F. Zhou, X. Zhang, X. Qiu, M. Kadoch, and M. Cheriet, "Deep learning-based resource allocation for 5g broadband tv service," *IEEE Transactions on Broadcasting*, vol. 66, no. 4, pp. 800–813, 2020.
- [136] G. Rjoub, J. Bentahar, O. A. Wahab, and A. Bataineh, "Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud," in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, IEEE, 2019, pp. 189–196.
- [137] P. Goswami, A. Mukherjee, M. Maiti, S. K. S. Tyagi, and L. Yang, "A neural-network-based optimal resource allocation method for secure iiot network," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2538–2544, 2021.
- [138] J. Shi, Q. Zhang, Y.-C. Liang, and X. Yuan, "Distributed deep learning power allocation for d2d network based on outdated information," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2020, pp. 1–6.
- [139] Z. Hu, J. Tu, and B. Li, "Spear: Optimized dependency-aware task scheduling with deep reinforcement learning," in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, IEEE, 2019, pp. 2037–2046.
- [140] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2019.
- [141] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 8, pp. 5404–5419, 2020.

- [142] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, 2019.
- [143] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2212–2230, 2020.
- [144] S. Xu, Q. Liu, B. Gong, et al., "Rjcc: Reinforcement-learning-based joint communicational-and-computational resource allocation mechanism for smart city iot," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8059–8076, 2020.
- [145] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for sdn based wireless networks," *Journal of Communications and Networks*, vol. 22, no. 1, pp. 1–11, 2019.
- [146] L. Lei, H. Xu, X. Xiong, K. Zheng, and W. Xiang, "Joint computation offloading and multiuser scheduling using approximate dynamic programming in nb-iot edge computing system," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5345–5362, 2019.
- [147] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid mec networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6252–6265, 2019.
- [148] H. Ye and G. Y. Li, "Deep reinforcement learning based distributed resource allocation for v2v broadcasting," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2018, pp. 440–445.
- [149] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

- [150] F. Xu, F. Yang, S. Bao, and C. Zhao, "Dqn inspired joint computing and caching resource allocation approach for software defined information-centric internet of things network," *IEEE Access*, vol. 7, pp. 61 987–61 996, 2019.
- [151] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint dnn partition deployment and resource allocation for delay-sensitive deep learning inference in iot," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9241–9254, 2020.
- [152] Y. Fan, Z. Zhang, and H. Li, "Message passing based distributed learning for joint resource allocation in millimeter wave heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 5, pp. 2872–2885, 2019.
- [153] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, "Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale mec networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9278–9290, 2020.
- [154] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, pp. 1–24, 2019.
- [155] J. Zhou, "Real-time task scheduling and network device security for complex embedded systems based on deep learning networks," *Microprocessors and Microsystems*, vol. 79, p. 103 282, 2020.
- [156] A. M. Kintsakis, F. E. Psomopoulos, and P. A. Mitkas, "Reinforcement learning based scheduling in a workflow management system," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 94–106, 2019.
- [157] S. Shadroo, A. M. Rahmani, and A. Rezaee, "The two-phase scheduling based on deep learning in the internet of things," *Computer Networks*, vol. 185, p. 107 684, 2021.
- [158] T. Dong, F. Xue, C. Xiao, and J. Li, "Task scheduling based on deep reinforcement learning in a cloud manufacturing environment," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, e5654, 2020.

- [159] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.
- [160] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1357–1371, 2019.
- [161] Z. Wei, F. Liu, Y. Zhang, J. Xu, J. Ji, and Z. Lyu, "A q-learning algorithm for task scheduling based on improved svm in wireless sensor networks," *Computer Networks*, vol. 161, pp. 138–149, 2019.
- [162] M. H. Moghadam and S. M. Babamir, "Makespan reduction for dynamic workloads in cluster-based data grids using reinforcement-learning based scheduling," *Journal of computational science*, vol. 24, pp. 402–412, 2018.
- [163] A. Chowdhury, S. A. Raut, and H. S. Narman, "Da-drls: Drift adaptive deep reinforcement learning based scheduling for iot resource management," *Journal of Network and Computer Applications*, vol. 138, pp. 51–65, 2019.
- [164] Y. Wang, H. Liu, W. Zheng, *et al.*, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE access*, vol. 7, pp. 39 974–39 982, 2019.
- [165] T. T. Sung, J. Ha, J. Kim, A. Yahja, C.-B. Sohn, and B. Ryu, "Deepsocks: A neural scheduler for heterogeneous system-on-chip (soc) resource scheduling," *Electronics*, vol. 9, no. 6, p. 936, 2020.
- [166] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2020.
- [167] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.

- [168] W. Zhan, C. Luo, J. Wang, et al., "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.
- [169] D. Rahbari and M. Nickray, "Task offloading in mobile fog computing by classification and regression tree," *Peer-to-Peer Networking and Applications*, vol. 13, pp. 104–122, 2020.
- [170] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, 2020.
- [171] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," *Journal of Optical Communications and Networking*, vol. 11, no. 1, pp. A84–A95, 2019.
- [172] F. P.-C. Lin and Z. Tsai, "Hierarchical edge-cloud sdn controller system with optimal adaptive resource allocation for load-balancing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 265–276, 2019.
- [173] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, 2017, pp. 164–176.
- [174] P. Vörös and A. Kiss, "Security middleware programming using p4," in *Human Aspects of Information Security, Privacy, and Trust: 4th International Conference, HAS 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings 4*, Springer, 2016, pp. 277–287.
- [175] Y. Afek, A. Bremler-Barr, and L. Shafir, "Network anti-spoofing with sdn data plane," in *IEEE INFOCOM 2017-IEEE conference on computer communications*, IEEE, 2017, pp. 1–9.
- [176] Y. Li, R. Miao, C. Kim, and M. Yu, "Lossradar: Fast detection of lost packets in data center networks," in *Proceedings of the 12th International Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 481–495.

- [177] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "Openstate: Programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44–51, 2014.
- [178] G. K. Ndonga and R. Sadre, "A two-level intrusion detection system for industrial control system networks using p4," in *5th International Symposium for ICS & SCADA Cyber Security Research 2018 5*, 2018, pp. 31–40.
- [179] F. Kuliesius and V. Dangovas, "Sdn enhanced campus network authentication and access control system," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, 2016, pp. 894–899.
- [180] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Accelerating fully homomorphic encryption using gpu," Sep. 2012, pp. 1–5, ISBN: 978-1-4673-1577-7. DOI: 10.1109/HPEC.2012.6408660.
- [181] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, ser. EUROCRYPT'11, Tallinn, Estonia: Springer-Verlag, 2011, pp. 129–148, ISBN: 9783642204647.
- [182] M. Matsumoto and M. Oguchi, "Speeding up encryption on iot devices using homomorphic encryption," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2021, pp. 270–275.
- [183] S. M. Fawaz, N. Belal, A. ElRefaey, and M. W. Fakhr, "A comparative study of homomorphic encryption schemes using microsoft seal," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2128, 2021, p. 012 021.
- [184] N. Samardzic, A. Feldmann, A. Krastev, et al., "F1: A fast and programmable accelerator for fully homomorphic encryption," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21, Virtual Event, Greece: Association for Computing Machinery, 2021, pp. 238–252, ISBN: 9781450385572. DOI: 10.1145/3466752.3480070. [Online]. Available: <https://doi.org/10.1145/3466752.3480070>.

- [185] A. C. Mert, E. Öztürk, and E. Savaş, "Design and implementation of encryption/decryption architectures for bfv homomorphic encryption scheme," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 353–362, 2019.
- [186] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-lwe cryptoprocessor," in *International workshop on cryptographic hardware and embedded systems*, Springer, 2014, pp. 371–391.
- [187] R. De Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Efficient software implementation of ring-lwe encryption," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015, pp. 339–344.
- [188] W. Dai and B. Sunar, "Cuhe: A homomorphic encryption accelerator library," in *International Conference on Cryptography and Information Security in the Balkans*, Springer, 2015, pp. 169–186.
- [189] I. Syafalni, G. Jonatan, N. Sutisna, R. Mulyawan, and T. Adiono, "Efficient homomorphic encryption accelerator with integrated prng using low-cost fpga," *IEEE Access*, vol. 10, pp. 7753–7771, 2022.
- [190] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 114–148, 2021.
- [191] Y. Cao, C. Xiao, B. Cyr, et al., "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.
- [192] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2017, pp. 445–467.

- [193] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, no. 2015, p. 995, 2015.
- [194] M. Abdelfattah, K. Yuan, Z. J. Wang, and R. Ward, "Adversarial attacks on camera-lidar models for 3d car detection," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2189–2194. doi: 10.1109/IROS51168.2021.9636638.
- [195] J. Zhang, Y. Zhang, K. Lu, et al., "Detecting and identifying optical signal attacks on autonomous driving systems," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1140–1153, 2021. doi: 10.1109/JIOT.2020.3011690.
- [196] R. Ivanov, M. Pajic, and I. Lee, "Attack-resilient sensor fusion," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2014, pp. 1–6.
- [197] R. S. Hallyburton, Y. Liu, and M. Pajic, "Security analysis of camera-lidar semantic-level fusion against black-box attacks on autonomous vehicles," *CoRR*, vol. abs/2106.07098, 2021. arXiv: 2106.07098. [Online]. Available: <https://arxiv.org/abs/2106.07098>.
- [198] Y. Cao, N. Wang, C. Xiao, et al., "Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," *CoRR*, vol. abs/2106.09249, 2021. arXiv: 2106.09249. [Online]. Available: <https://arxiv.org/abs/2106.09249>.
- [199] K. Lim and K. M. Tuladhar, "Lidar: Lidar information based dynamic v2v authentication for roadside infrastructure-less vehicular networks," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–6. doi: 10.1109/CCNC.2019.8651684.
- [200] J. Tu, M. Ren, S. Manivasagam, et al., "Physically realizable adversarial examples for lidar object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 716–13 725.

- [201] Z. Hau, S. Demetriou, L. Muñoz-González, and E. C. Lupu, “Shadow-catcher: Looking into shadows to detect ghost objects in autonomous vehicle 3d sensing,” in *European Symposium on Research in Computer Security*, Springer, 2021, pp. 691–711.
- [202] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, “Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2021. doi: 10.1109/TWC.2020.3024538.
- [203] L. P. Qian, A. Feng, Y. Huang, Y. Wu, B. Ji, and Z. Shi, “Optimal sic ordering and computation resource allocation in mec-aware noma nb-iot networks,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2806–2816, 2019. doi: 10.1109/JIOT.2018.2875046.
- [204] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, D. Vigo, et al., “Bin packing approximation algorithms: Survey and classification.,” in *Handbook of combinatorial optimization*, Springer, 2013, pp. 455–531.
- [205] P. Festa, “A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems,” in *2014 16th International Conference on Transparent Optical Networks (ICTON)*, IEEE, 2014, pp. 1–20.
- [206] M. A. Rodriguez and R. Buyya, “A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, e4041, 2017.
- [207] F. Wu, Q. Wu, and Y. Tan, “Workflow scheduling in cloud: A survey,” *The Journal of Supercomputing*, vol. 71, pp. 3373–3418, 2015.
- [208] W. Lu, W. Wu, J. Xu, P. Zhao, D. Yang, and L. Xu, “Auction design for cross-edge task offloading in heterogeneous mobile edge clouds,” *Computer Communications*, vol. 181, pp. 90–101, 2022.
- [209] K. Choudhary, B. DeCost, C. Chen, et al., “Recent advances and applications of deep learning methods in materials science,” *npj Computational Materials*, vol. 8, no. 1, p. 59, 2022.

- [210] G. Rodola, *Psutil documentation*. [Online]. Available: <https://psutil.readthedocs.io/en/latest/> (visited on 09/07/2023).
- [211] S. Abirami and P. Chitra, "Energy-efficient edge based real-time healthcare support system," in *Advances in computers*, 1, vol. 117, Elsevier, 2020, pp. 339–368.
- [212] L. Wen, X. Li, and L. Gao, "A new reinforcement learning based learning rate scheduler for convolutional neural network in fault classification," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 12, pp. 12 890–12 900, 2020.
- [213] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, and L. Yang, "Model-based reinforcement learning with value-targeted regression," in *International Conference on Machine Learning*, PMLR, 2020, pp. 463–474.
- [214] R. Medar, V. S. Rajpurohit, and B. Rashmi, "Impact of training and testing data splits on accuracy of time series forecasting in machine learning," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017, pp. 1–6. doi: 10.1109/ICCUBEA.2017.8463779.
- [215] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105 524, 2020, ISSN: 1568-4946. doi: <https://doi.org/10.1016/j.asoc.2019.105524>.
- [216] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Mathematical problems in engineering*, vol. 2013, 2013. doi: <https://doi.org/10.1155/2013/425740>.
- [217] J.-Y. Li, T. Chow, and Y.-L. Yu, "The estimation theory and optimization algorithm for the number of hidden units in the higher-order feedforward neural network," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 3, 1995, 1229–1233 vol.3. doi: 10.1109/ICNN.1995.487330.

- [218] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layers versus three," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 251–255, 1997. doi: 10.1109/72.557662.
- [219] S. Xu and L. Chen, "A novel approach for determining the optimal number of hidden layer neurons for fnn's and its application in data mining," *ICITA*, 2008.
- [220] K. Shibata and Y. Ikeda, "Effect of number of hidden neurons on learning in large-scale layered neural networks," in *2009 ICCAS-SICE*, IEEE, 2009, pp. 5008–5013.
- [221] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—a comparative study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228–240, 2012. doi: 10.1109/TII.2012.2187914.
- [222] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß, and M. Steinbrecher, "Computational intelligence: Eine methodische einföhrung in künstliche neuronale netze," *Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze Vieweg+ Teubner, Wiesbaden*, 2011.
- [223] L. Serrano, *Grokking Machine Learning*. Simon and Schuster, 2021.
- [224] F. von Tüllenbug and T. Pfeiffenberger, "Concepts for reliable communication in a software-defined network architecture," in *Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings 36*, Springer, 2017, pp. 173–186.
- [225] P. Bosshart, D. Daly, G. Gibb, et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.
- [226] J. Hyun and J. W.-K. Hong, "Knowledge-defined networking using in-band network telemetry," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2017, pp. 54–57.

- [227] A. C. Baktir, A. Ozgovde, and C. Ersoy, "Implementing service-centric model with p4: A fully-programmable approach," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–6.
- [228] F. Paolucci, F. Cugini, and P. Castoldi, "P4-based multi-layer traffic engineering encompassing cyber security," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, IEEE, 2018, pp. 1–3.
- [229] Y. Afek, A. Bremler-Barr, S. L. Feibish, and L. Schiff, "Detecting heavy flows in the sdn match and action model," *Computer Networks*, vol. 136, pp. 1–12, 2018.
- [230] F. H. M. Ali, R. Yunos, and M. A. M. Alias, "Simple port knocking method: Against tcp replay attack and port scanning," in *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, IEEE, 2012, pp. 247–252.
- [231] J. Aycock, M. Jacobson, et al., "Improved port knocking with strong authentication," in *21st Annual computer security applications conference (ACSAC'05)*. IEEE, 2005.
- [232] M. Nabi-Abdolyousefi and M. Mesbahi, "Network identification via node knockout," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3214–3219, 2012.
- [233] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: Dynamic access control for enterprise networks," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*, 2009, pp. 11–18.
- [234] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized? state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 1–6.
- [235] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.

- [236] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful sdn data planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017.
- [237] U. LACORE, "A review of port scanning techniques,"
- [238] MPICH. "High-performance portable mpi." (2008), [Online]. Available: <https://www.mpich.org/> (visited on 09/07/2023).
- [239] *Microsoft SEAL (release 4.0)*, Microsoft Research, Redmond, WA., Mar. 2022. [Online]. Available: <https://github.com/Microsoft/SEAL> (visited on 09/07/2023).
- [240] M. Albrecht, M. Chase, H. Chen, et al., "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., Nov. 2018.
- [241] A. Kim, Y. Polyakov, and V. Zucca, "Revisiting homomorphic encryption schemes for finite fields," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2021, pp. 608–639.
- [242] M. Chase, H. Chen, J. Ding, et al., "Security of homomorphic encryption," *HomomorphicEncryption.org, Redmond WA, Tech. Rep*, 2017.
- [243] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2021, pp. 648–677.
- [244] J. Cho, J. Ha, S. Kim, et al., "Transciphering framework for approximate homomorphic encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2021, pp. 640–669.
- [245] B. Li, D. Micciancio, M. Schultz, and J. Sorrell, "Securing approximate homomorphic encryption using differential privacy," in *Annual International Cryptology Conference*, Springer, 2022, pp. 560–589.

- [246] Y. Peng, Y. Qin, X. Tang, Z. Zhang, and L. Deng, "Survey on image and point-cloud fusion-based object detection in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22 772–22 789, 2022. doi: 10.1109/TITS.2022.3206235.
- [247] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020. doi: 10.1109/MSP.2020.2973615.
- [248] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust {lidar-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 877–894.
- [249] Y. Cao, J. Ma, K. Fu, R. Sara, and M. Mao, "Automated tracking system for lidar spoofing attacks on moving targets," *The Network and Distributed System Security*, 2021.
- [250] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711–1719.
- [251] M. M. Lau and K. H. Lim, "Review of adaptive activation function in deep neural network," in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, IEEE, 2018, pp. 686–690.
- [252] V. Verdhan, "Computer vision using deep learning," *Berkeley, CA: Apress*, 2021.
- [253] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [254] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *arXiv preprint arXiv:2008.05756*, 2020.

- [255] M. H. Ferris, M. McLaughlin, S. Grieggs, et al., "Using roc curves and auc to evaluate performance of no-reference image fusion metrics," in *2015 National Aerospace and Electronics Conference (NAECON)*, IEEE, 2015, pp. 27–34.
- [256] K. Klahr, *Performance-metriken des überwachten lernens für klassifikationsprobleme*, <https://www.saracus.com/blog/performance-metriken-klassifikation-2-2/>, Nov. 2018. (visited on 09/07/2023).
- [257] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.
- [258] C. Pan, S. Zhang, C. Zhao, H. Shi, Z. Kong, and X. Cui, "A novel active queue management algorithm based on average queue length change rate," *IEEE Access*, vol. 10, pp. 75 558–75 570, 2022. doi: 10.1109/ACCESS.2022.3189183.
- [259] C. D'apice, M. P. D'arienzo, A. Dudin, and R. Manzo, "Admission control in priority queueing system with servers reservation and temporal blocking admission of low priority users," *IEEE Access*, vol. 11, pp. 44 425–44 443, 2023. doi: 10.1109/ACCESS.2023.3273148.
- [260] Y. Chiang, C.-H. Hsu, G.-H. Chen, and H.-Y. Wei, "Deep q-learning-based dynamic network slicing and task offloading in edge network," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 369–384, 2022.
- [261] C. Xu and W. Song, "Intelligent task allocation for mobile crowdsensing with graph attention network and deep reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 1032–1048, 2023.
- [262] B. Lim and M. Vu, "Distributed multi-agent deep q-learning for load balancing user association in dense networks," *IEEE Wireless Communications Letters*, 2023.
- [263] W. Cheng, X. Liu, X. Wang, and G. Nie, "Task offloading and resource allocation for industrial internet of things: A double-dueling deep q-network approach," *IEEE Access*, vol. 10, pp. 103 111–103 120, 2022.

- [264] J. Zheng, Y. Pan, S. Jiang, Z. Chen, and F. Yan, "A federated learning and deep q-network based cooperative resource allocation algorithm for multi-level services in mobile edge computing networks," *IEEE Transactions on Cognitive Communications and Networking*, 2023.
- [265] Z. Tianqing, W. Zhou, D. Ye, Z. Cheng, and J. Li, "Resource allocation in iot edge computing via concurrent federated reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1414–1426, 2021.
- [266] Y. He, M. Yang, Z. He, and M. Guizani, "Computation offloading and resource allocation based on dt-mec-assisted federated learning framework," *IEEE Transactions on Cognitive Communications and Networking*, 2023.