University of New Mexico
UNM Digital Repository

Electrical and Computer Engineering ETDs | Engineering ETDs

Summer 6-25-2025

AI-Driven EMI Mitigation for Smart Healthcare: Generative Adversarial Networks and Edge Computing for Reliable Medical Systems

Mona Esmaeili

Follow this and additional works at: https://digitalrepository.unm.edu/ece_etds

 Part of the Electrical and Computer Engineering Commons

Recommended Citation

Esmaeili, Mona. "AI-Driven EMI Mitigation for Smart Healthcare: Generative Adversarial Networks and Edge Computing for Reliable Medical Systems." (2025). https://digitalrepository.unm.edu/ece_etds/728

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.

Mona Esmaeili

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Payman Zarsh-Ha

Chair

Dr. Sameer Hemmady

Co-Chair

Dr. Edl Schamiloglu

Member

Dr. Christos Christodoulou

Member

AI-Driven EMI Mitigation for Smart Healthcare: Generative Adversarial Networks and Edge Computing for Reliable Medical Systems

BY

Mona Esmaeili

B.S., Computer Science, Iran University of Science and Technology

M.S., Computer Science, University of Washington

DISSERTATION

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2025

DEDICATION

*T*o my parents, Mahnaz and Esmaeil, who steadfastly stood by me through every setback on my journey.

ACKNOWLEDGMENTS

First, I wish to express my profound gratitude to my esteemed advisor, Professor Payman Zarkesh-Ha, for entrusting me with the invaluable opportunity to advance my graduate studies. Professor Payman Zarkesh-Ha, your constant support, mentorship, and encouragement have been indispensable throughout my academic journey.

I extend my heartfelt thanks to the members of my committee for generously dedicating their time and providing invaluable feedback and suggestions to enhance this work.

AI-Driven EMI Mitigation for Smart Healthcare: Generative Adversarial Networks and Edge Computing for Reliable Medical Systems

by

Mona Esmaeili

B.S., Computer Science, Iran University of Science and Technology

M.S., Computer Science, University of Washington

Ph.D. Engineering, University of New Mexico, 2025

ABSTRACT

Electromagnetic interference (EMI) from radio frequency (RF) sources poses a major challenge to digital systems, especially in high-electromagnetic environments. Traditional electromagnetic compatibility (EMC) analysis often focuses on continuous wave (CW) interference and overlooks the effects of waveform modulation on EMI coupling. This thesis explores how modulated waveforms influence EMI coupling and introduces a Generative Adversarial Network (GAN)-based classification framework to distinguish between harmful and non-harmful EMI signals. The study extends conventional EMC analysis using machine learning (ML), showing that modulated EMI waveforms can increase coupling by up to 27% compared to CW signals. This highlights the need for waveform-aware EMI mitigation strategies. The proposed GAN-based framework offers an accurate, automated method for evaluating EMI risk. Beyond EMC, the research applies EMI mitigation to medical asset management and mobile edge computing. Using predictive analytics and reinforcement learning, it enhances hospital resource management and improves the reliability of cloud-connected systems, contributing to resilient and secure digital infrastructure.

TABLE OF CONTENTS

List of Figures xi

List of Tables xv

List of Publications xvi

List of Abbreviations xviii

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Research Objectives 2

2 Impact of Waveform Modulation from Electromagnetic Interference Sources on Coupling to Digital Electronic Interconnects **4**

2.1 PROBLEM STATEMENT 5

2.2 ANALYTICAL SOLUTION 7

2.2.1 Evaluation of Fundamental Frequency 8

2.2.2 Evaluation of Harmonic Phases 9

2.2.3 Evaluation of Harmonic Amplitudes 10

2.3 A PRACTICAL EXAMPLE 12

2.4 CONCLUSION 14

3 Advancing EMC Analysis with GAN-Driven Signal Classification and Waveform Modulation **16**

3.1 Related Work 18

3.1.1 EMC and Digital Interconnects 18

3.1.2 RF-Driven EMI Analysis 18

3.2 Dataset 19

3.3 Methodology 22

3.3.1 GANs: Training and Dataset Usage: 22

3.3.2 Signal Processing Techniques: Shifting and Scaling 23
3.4 Deep Learning for the Discrimination of Safe and Unsafe Signals 25
3.4.1 Architectural Insights: A Comparative Evaluation of CNNs in Image Classification 25
3.4.2 GAN Architecture and Training 31
3.5 Results 32
3.5.1 Confusion Matrix Analysis 32
3.5.2 Image Classification Based on Shifting and Scaling 35
3.5.3 GAN Evaluation 36
3.6 Conclusion 39

4 Advancements in AI-Driven Customer Service 41
4.1 Background 41
4.1.1 Motivation 41
4.1.2 Related Work 42
4.2 Problem Description & Datasets 44
4.2.1 Datasets 44
4.2.2 Evaluation Methods 44
4.2.3 Metrics 46
4.3 Experiments 46
4.3.1 Solution Architecture 46
4.3.2 Data Preprocessing 47
4.3.3 Algorithms 48
4.4 Results 51
4.5 Limitations and Conclusion 51

5 IoT-based Contact Tracing Framework in Smart Cities 53
5.1 IoT-enabled Contact Tracing Framework 55

5.1.1 Architecture 55
5.1.2 User-IoT device Distance Estimation using RSSI 57
5.1.3 IoT Device Scanning 59
5.1.4 Centralized Processing 60
5.2 Infection Risk Assessment using Machine Learning 62
5.2.1 Risk Classification 62
5.2.2 Classification Algorithms 64
5.3 Experimental Evaluation 66
5.3.1 Case Study 66
5.3.2 User-Device Distance Estimation Accuracy 67
5.3.3 System Parameter Evaluation and Overall Accuracy 69
5.3.4 Evaluation of risk classification 72
5.3.5 Impact of Window Size 74
5.4 Related Work 74
5.5 Conclusions 76

6 Multi Agent Based Traffic Prediction for ANMS 77

6.1 Overview of Network Automation and Enabling Technologies 81
6.1.1 Overview of Network Management Automation 81
6.1.2 Overview of the Multi-agent System 82
6.1.3 Agent as a Decision Making Element in Autonomic Networking . 83
6.1.4 Application of Traditional ML and DL as a Cognitive Component of Agents 84
6.2 Proposed Network Traffic Predictor and Traffic Classification Agents . . . 90
6.2.1 Proposed Network Traffic Predictor Agent Architectural Framework 92
6.2.2 Proposed Network Traffic Classifier Agent Architectural Framework 95
6.2.3 Communication Interface Between Agents 95
6.3 Implementation Scenario and Conceptual Framework 96

6.3.1 Implementation Conceptual Framework for MANA-NMS using NTPA . . . 97
6.3.2 Implementation Conceptual Framework for MANA-NMS using NTCA . . . 99
6.4 Performance Evaluation . . . 102
6.4.1 Accuracy of NTPA using Different DL Models . . . 102
6.4.2 Training Latency and Predicting Latency of NTPA using Different DL Models . . . 106
6.4.3 Computing Resource Utilization of NTPA Using Different DL Models 108
6.4.4 Accuracy of NTCA using Different ML Models . . . 109
6.4.5 Classification and Training Latency of NTCA using Different ML Models . . . 111
6.4.6 Computing Resource Utilization of NTCA using Different ML Models . . . 112
6.5 Conclusion and Future Work . . . 114

7 DRL for Online Latency Aware Workload Offloading in MEC 116
7.1 Related Work . . . 119
7.2 System Model . . . 120
7.3 Deep reinforcement learning based offloading decision and resource management . . . 123
7.4 Simulation Results . . . 126
7.5 Conclusion . . . 132

8 Hospital Asset Usage Prediction using Deep Learning 133
8.1 Literature Review . . . 135
8.2 Problem Description . . . 137
8.3 System Model . . . 139

8.3.1 Data Modeling-Time Series Analysis 139
8.3.2 IoT Infrastructure-Data Collection and Processing 140
8.3.3 Prediction Algorithms 141
8.4 Experiment and Numerical Results 143
8.4.1 Case Study 144
8.4.2 System Evaluation and Overall Accuracy 144
8.4.3 Range Prediction for Different Device Categories 147
8.4.4 Impact of Combining DL models with VMD 150
8.5 Conclusions 152

9 Conclusion **153**

9.1 Summary of Contributions 153
9.2 Future Research Directions 154
References 156

LIST OF FIGURES

Figure 2.1. A simple digital system subjected to EMI emission, where the noise margin is compared with the peak-to-peak value of the couple signal. 5

Figure 2.2. The representation of the digital system under EMI shown in Fig. 1 using the s-parameters for the electromagnetic coupling. 6

Figure 2.3. An arbitrary S21 frequency response. 8

Figure 2.4. Demonstration of constructive/destructive phases for harmonic 3. . . 9

Figure 2.5. Demonstration of constructive/destructive phases for harmonic 2. . . 10

Figure 2.6. Demonstration of constructive/destructive phases for harmonic 5. . . 11

Figure 2.7. A measured example of S21 frequency response. 12

Figure 2.8. The optimal input and output waveforms. 14

Figure 2.9. The single tone input and output waveforms. 15

Figure 3.1. Overview of the constructed dataset for the classification of electrical signals. The dataset includes labeled images representing 'safe' and 'unsafe' signals based on peak-to-peak voltage thresholds, with a maximum value of 0.55 V and classification boundary at 0.32 V. The dataset comprises 1500 images for each category, facilitating robust training and evaluation of the classification model. 21

Figure 3.2. Architecture of a Generative Adversarial Network (GAN). 23

Figure 3.3. The confusion matrix for the different models. 33

Figure 3.4. A comparison of the Area Under the Curve (AUC) scores for different deep learning models. 34

Figure 3.5. The original input signal and its corresponding output. 36

Figure 3.6. The shifted input signal and its corresponding output. 36

Figure 3.7. The scaled input signal and its corresponding output. 37

Figure 3.8. Distribution of peak-to-peak values for GAN-generated safe signals (accuracy: 92%) . . . 38
Figure 3.9. Distribution of peak-to-peak values for GAN-generated unsafe signals (Accuracy: 93%) . . . 38

Figure 4.1. Example of a dataset format. . . . 44
Figure 4.2. Solution architecture of our chatbot. . . . 46

Figure 5.1. Framework Architecture . . . 56
Figure 5.2. RSSI Estimation Using Kalman Filter [119] . . . 58
Figure 5.3. Trilateration . . . 61
Figure 5.4. Label-risk function . . . 64
Figure 5.5. Location of IoT Devices - Smart Space Topology [141] . . . 67
Figure 5.6. Average absolute distance error vs. real distance . . . 68
Figure 5.7. Contact occurrences vs. overlapping time and distance during a single day . . . 70
Figure 5.8. Overall number of weekly contacts vs (Left) distance, and (Right) overlapping time . . . 71
Figure 5.9. Classification performance through a Confusion-Matrix . . . 72
Figure 5.10. Accuracy vs. window size . . . 73
Figure 5.11. Accuracy vs. AUC . . . 73

Figure 6.1. A Simple MAS Architecture. . . . 82
Figure 6.2. An Example of a DT. . . . 89
Figure 6.3. An Illustration of a Hyperplane and Support Vectors. . . . 90
Figure 6.4. An Illustration of the K-NN Classifier Algorithm in action. . . . 91
Figure 6.5. Network Traffic Classifier Agent (top) and Network Traffic Predictor (bottom) Agent Architecture. . . . 94
Figure 6.6. Simple Architecture for the MANA-NMS. . . . 98

Figure 6.7. Implemented MANA-NMS Scenario. 101
Figure 6.8. Different DL Models Performance Comparison. 105
Figure 6.9. RMSE for Different Dataset Size. 105
Figure 6.10. MAPE for Different Dataset Size. 106
Figure 6.11. Comparison of NTPA's Training Latency. 107
Figure 6.12. Comparison of NTPA's Predicting Latency. 107
Figure 6.13. Comparison of NTPA's Memory Usage. 108
Figure 6.14. Comparison of NTPA's CPU Usage. 109
Figure 6.15. Comparison of NTCA's Training Latency 111
Figure 6.16. Comparison of NTCA's Classification Latency. 112
Figure 6.17. Comparison of NTCA's Memory Usage. 113
Figure 6.18. Comparison of NTCA's CPU Usage. 114

Figure 7.1. The MEC architecture. 117
Figure 7.2. Learning curve of DECENT. 128
Figure 7.3. Performance comparison. 128
Figure 7.4. Average weighted response time over average task arrival rate for different algorithms. 129
Figure 7.5. Average response time w.r.t λ and weights of the tasks. 129
Figure 7.6. Average weighted response time over μ. 130
Figure 7.7. Average weighted response time over α. 131

Figure 8.1. IoT Infrastructure [2]. 138
Figure 8.2. Structure of VMD-CNN [112]. 144
Figure 8.3. Device count evolution over time step using VMD-CNN: predicted vs target values for different devices. 146
Figure 8.4. Pulse Oximeter range prediction for different time horizons using VMD-CNN. 149

Figure 8.5. Accuracy comparison for combined CNN (top) and GRU (bottom) approaches. 151

LIST OF TABLES

3.1 Performance metrics for deep Learning models 34

4.1 Algorithm Performance 48

5.1 Key Notations 56

5.2 Case-Study Parameters 68

5.3 Relative distance error vs. real distance 69

5.4 Algorithm Performance 71

6.1 Hardware Configuration 96

6.2 Configuration Parameters 104

6.3 NTCA's Classification Accuracy 110

7.1 Simulation Parameters. 127

8.1 Case-Study Parameters 144

8.2 Evaluation of combined DL models with VMD method forecasting performance. 145

LIST OF PUBLICATIONS

1. Esmaeili M, Hemmady S, Noakoasteen O, Christoddoulou C, Schamiloglu E, Zarkesh-Ha P. Impact of Waveform Modulation from Electromagnetic Interference Sources on Coupling to Digital Electronic Interconnects Conference on 2024 International Symposium on Electromagnetic Compatibility–EMC Europe Accepted. IEEE. **(Chapter 2)**

2. Esmaeili M, Hemmady S, Noakoasteen O, Schamiloglu E, Christoddoulou C, Zarkesh-Ha P. Advancing EMC Analysis with GAN-Driven Signal Classification and Waveform Modulation Journal on IEEE Access Accepted. 2024 IEEE World AI IoT Congress (AIIoT). **(Chapter 3)**

3. Esmaeili M, Ahmadi M, Ismaeil M, Mirzaei S, Verdial J. Advancements in AI-Driven Customer Service. IEEE. **(Chapter 4)**

4. Akhavan Z, Esmaeili M, Sikeridis D, Devetsikiotis M. Internet of Things-enabled Passive Contact Tracing in Smart Cities Journal on Internt of Things Accepted. Elsevier. **(Chapter 5)**

5. Arzo ST, Akhavan Z, Esmaeili M, Devetsikiotis M, Granelli F. Multi-Agent-Based Traffic Prediction and Traffic Classification for Autonomic Network Management Systems for Future Networks.
   Future Internet. 2022 Aug;14(8):230. **(Chapter 6)**

6. Akhavan Z, Esmaeili M, Badnava B, Yousefi M, Sun X, Devetsikiotis M, Zarkesh-Ha P. Deep Reinforcement Learning for Online Latency Aware
   Workload Offloading in Mobile Edge Computing. InGLOBECOM 2022-2022 IEEE Global Communications Conference 2022 Dec 4 (pp. 2218-2223).
   IEEE. **(Chapter 7)**

7. Esmaeili M, Akhavan Z, Nasiri H, Worku YM, Arzo ST, Stavropoulos A, Devetsikiotis M, Zarkesh-Ha P. Medical Asset Management: Deep Learning Based Asset Usage Prediction in a Hospital Setting using Real Data. International Conference on Machine Learning and Applications(ICMLA) Accepted. IEEE. **(Chapter 8)**

LIST OF ABBREVIATIONS

IoT: Internet of Things

BLE: Bluetooth Low Energy

RSSI: Received Signal Strength Indicator

ANMS: Autonomic Network Management Systems

ML: Machine Learning

DL: Deep Learning

DRL: Deep Reinforcement Learning

MEC: Mobile Edge Computing

DECENT Algorithm: Deep reinforcement lEarning based offloading deCision and rEsource managemeNT

PAR: Periodic Automatic Replenishment

Electromagnetic compatibility : EMC

Continuous wave : CW

Generative Adversarial Networks : GAN

Reinforcement Learning : RL

Electromagnetic Compatibility : EMC

Radio Frequency : RF

Power Spectral Density : PSD

Electromagnetic Interference : EMI

Pulse Width Modulation : PWM

Chapter 1

Introduction

1.1 Background and Motivation

Modern electronic systems are increasingly vulnerable to electromagnetic interference (EMI), especially in high-density environments where analog, digital, and wireless circuits operate in close proximity. Traditional EMI mitigation techniques such as shielding, grounding, and filtering are often insufficient when dealing with complex, time-varying interference sources. This challenge is compounded by the growing complexity of modern computing systems and the proliferation of high-speed digital interconnects and wireless technologies.

Traditional electromagnetic compatibility (EMC) analysis typically relies on continuous wave (CW) interference testing. However, recent studies show that modulated EMI signals such as those generated using spread-spectrum or chaotic modulation can exhibit significantly higher coupling efficiency, necessitating more sophisticated mitigation strategies.

In parallel, recent advances in machine learning (ML) have opened new avenues for intelligent EMI analysis. For example, Li et al. [91] employed deep reinforcement learning to optimize pin-map configurations for EMI immunity. Similarly, Zhao et al. [176] and Bian et al. [31] demonstrated convolutional neural network (CNN)-based approaches for EMI cancellation in RF-sensitive applications like MRI. These works highlight how ML enables real-time interference prediction and adaptive mitigation capabilities that are challenging to achieve using traditional rule-based techniques.

Moreover, the increasing digitalization of modern infrastructure including smart healthcare, cloud-edge systems, and mobile edge computing (MEC) demands robust EMI aware system design. In such environments, EMI not only affects performance but can also com-

promise safety and data integrity. ML driven EMI classification and mitigation strategies are thus gaining attention as scalable, data-driven solutions.

This dissertation bridges EMC engineering and ML-based methods to develop intelligent, EMI resilient frameworks for a variety of applications. By leveraging Generative Adversarial Networks (GANs) for signal classification and deep reinforcement learning (DRL) for optimization, the research proposes novel tools for EMI risk assessment and system-level mitigation across healthcare, edge computing, and smart infrastructure domains.

1.2 Research Objectives

This dissertation is guided by the following objectives:

- **Analyze the Impact of Waveform Modulation on EMI Coupling:** Extend traditional EMC models to investigate how different modulation schemes influence EMI coupling in digital interconnects.
- **Develop a GAN-Based EMI Classification Framework:** Design a deep learning-based GAN model for automated classification of EMI waveforms, enabling accurate identification of harmful interference.
- **Optimize EMI Mitigation in Medical Applications:** Investigate EMI-induced disruptions in hospital infrastructure and propose ML-driven forecasting models for asset tracking and equipment reliability.
- **Enhance Mobile Edge Computing (MEC) Through EMI-Aware Optimization:** Implement deep reinforcement learning (DRL) techniques for workload offloading in MEC environments, mitigating EMI-induced disruptions.

- **Integrate EMI Mitigation into Smart City Infrastructure:** Apply ML-driven EMI classification and mitigation strategies to smart healthcare, IoT-based transportation, and energy networks to enhance system reliability.
- **Incorporate Recommendation System Principles into EMI Mitigation:** Leverage ranking and retrieval-based models commonly used in recommendation systems to suggest optimal EMI mitigation strategies based on historical interference patterns, system context, and classification results.

Chapter 2

Impact of Waveform Modulation from Electromagnetic Interference Sources on Coupling to Digital Electronic Interconnects

Electromagnetic compatibility (EMC) engineering of commercial digital electronics is a well-known problem, which assumes continuous wave (CW), single-tone radiation across a given bandwidth [41]. However, the electromagnetic interference waveform can exhibit time-varying power- spectral density (PSD) profiles. In this paper, we study how such EMI waveform modulation can impact the coupling to digital interconnects and address the specific question of "Are there EMI waveform modulations which optimally couple to digital interconnects and, thereby, lead to enhanced interference that is not predicted by randomly chosen CW EMI modulations?". Addressing this critical EMI/EMC question will provide guidance to EMC engineers to design optimal shielding techniques to consumer electronics that operate in arbitrarily varying EMI environments.

Figure 2.1 illustrates a simple digital system in an arbitrary EMI environment, where the interconnect acts as the receiving antenna coupling the radiated signal onto the digital data stream. For our analysis, we assume that the main criterion that quantifies the adverse effect of EMI in this system is the peak-to-peak value of the externally coupled EMI signal onto the interconnect. Generally, once the peak-to-peak value becomes larger than the noise margin of the digital logic, it will induce a logic-fault. With faster clock speeds, modern-day consumer electronics utilize reduced scaled CMOS transistors with reduced noise margins, which exacerbates the system's sensitivity to EMI [79]. Therefore, it is imperative to properly model the coupling effect of EMI in a real system where the EMI waveform is not purely a single-tone CW emission.

In [99], it was shown that the multi-resonant impedance profile of the power distribution network (PDN) can induce conditions whereby a specifically modulated waveform can

induce voltage waves of exceedingly high peak values ("rogue waves"). Here, we posit that the similar conditions can be manifested when considering coupling profiles onto interconnects, which can be nonlinear with frequency. The main objective of this paper is to demonstrate that the modulation of an EMI waveform can affect the peak-to-peak value of the coupled signal and, thereby, can have significant impact on the coupling effectiveness of the EMI on digital circuits.

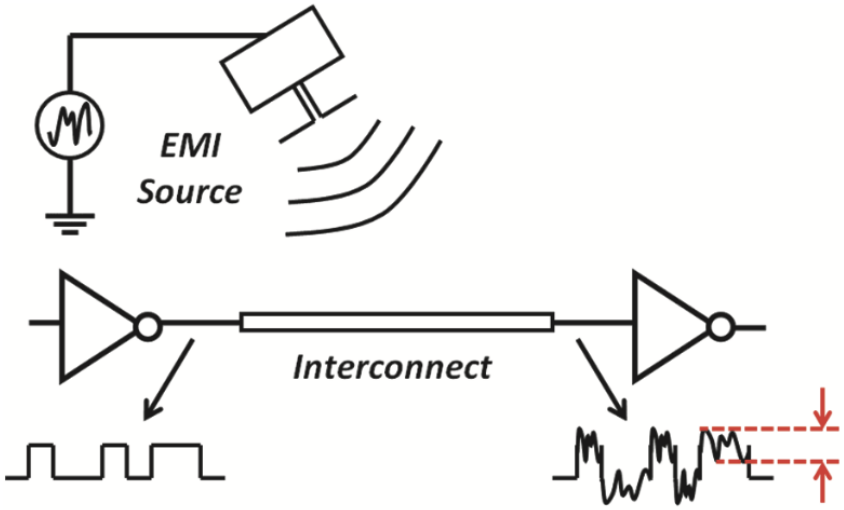


Figure 2.1: A simple digital system subjected to EMI emission, where the noise margin is compared with the peak-to-peak value of the couple signal.

An analytical optimization method is also presented to find the optimal EMI waveform modulation that maximizes the coupling effectiveness to a digital interconnect using the Fourier series approximation approach. The paper continues with the problem statement in Section II. Next, the analytical solution is presented in Sections III. Section IV presents a practical example to maximize the peak-to-peak value of the coupled signal. Finally, the conclusions are discussed in Section V.

2.1 PROBLEM STATEMENT

The EMI system in Fig. 2.1 can be represented with the block diagram shown in Fig. 2.2 Since coupling profiles are linear in nature due to the passive nature of the interconnects,

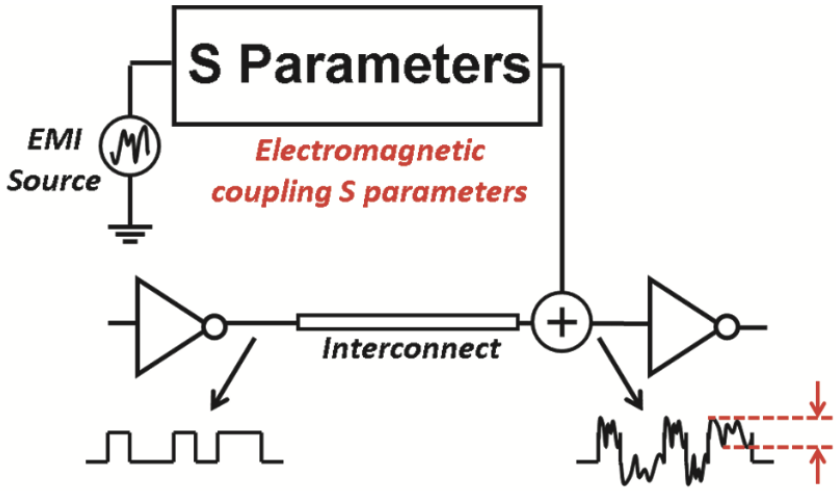


Figure 2.2: The representation of the digital system under EMI shown in Fig. 1 using the s-parameters for the electromagnetic coupling.

the EMI coupling to the interconnect can be modelled with the s-parameters frequency response, namely S21. Evidently, if the EMI source is a CW single-tone signal, the maximum peak-to- peak occurs at the frequency at which S21 has the highest peak. However, we pose the question, "Can a specific EMI waveform induce higher peak-to-peak signal at the interconnect as compared to the randomly chosen CW tones?" If this is true, a randomly chosen CW single-tone response will establish a conversative approach to EMC engineering analysis, instead of being accurate. To solve this problem, we assume that the time-series optimal waveform at the EMI source can be represented by a Fourier series as 2.4,

$$x(t) = \sum_{n=1}^{N} a_n \sin(2\pi n f_0 t + \phi_n) \tag{2.1}$$

where f_0 is the fundamental frequency, and $a_1, \phi_1, a_2, \phi_2, a_3, \phi_3, \ldots, a_N, \phi_N$ are the amplitudes and phases of harmonics 1 to N, respectively. The optimal waveform for $x(t)$ can be achieved by optimizing all the parameters in (2.1), such that the peak-to-peak value of the output signal, $y(t)$, is maximized for a constant input RF power (the constraint),

where

$$y(t) = \sum_{n=1}^{N} b_n \sin(2\pi n f_0 t + \theta_n), \tag{2.2}$$

and

$$b_n = a_n \cdot |S_{21}(2\pi n f_0)|, \tag{2.3}$$

$$\theta_n = \phi_n + \angle S_{21}(2\pi n f_0). \tag{2.4}$$

The RF power of the EMI source signal, $x(t)$, needed for the constraint, can be calculated by the RMS values of its harmonics as

$$P_{\text{Source}} = \frac{V_{\text{RMS}}^2}{Z_0} = \frac{\sum_{n=1}^{N} a_n^2}{2Z_0}, \tag{2.5}$$

where Z_0 is the radiation impedance of the system at the input port, which is assumed to be 50 Ω in our analysis. Equation (2.5) can be easily proven since all harmonics are orthogonal functions.

This optimization problem can be solved numerically by sweeping all Fourier series parameters in equation(2.1) and then using an exhaustive search to select the parameters that maximize the output peak-to-peak signal in equation (2.2), for a given constant input power in equation (2.5). However, due to the large number of variables, this approach is computationally expensive. In this paper, we develop an analytical solution to identify the optimal waveform.

2.2 ANALYTICAL SOLUTION

The optimal solution for $x(t)$ can be determined by finding the Fourier series parameters in equation (2.1), which includes the fundamental frequency, f_0, the phase of each harmonic, ϕ_n, and the amplitude of each harmonic, a_n. In this Section, we develop an analytical solution to find these parameters.

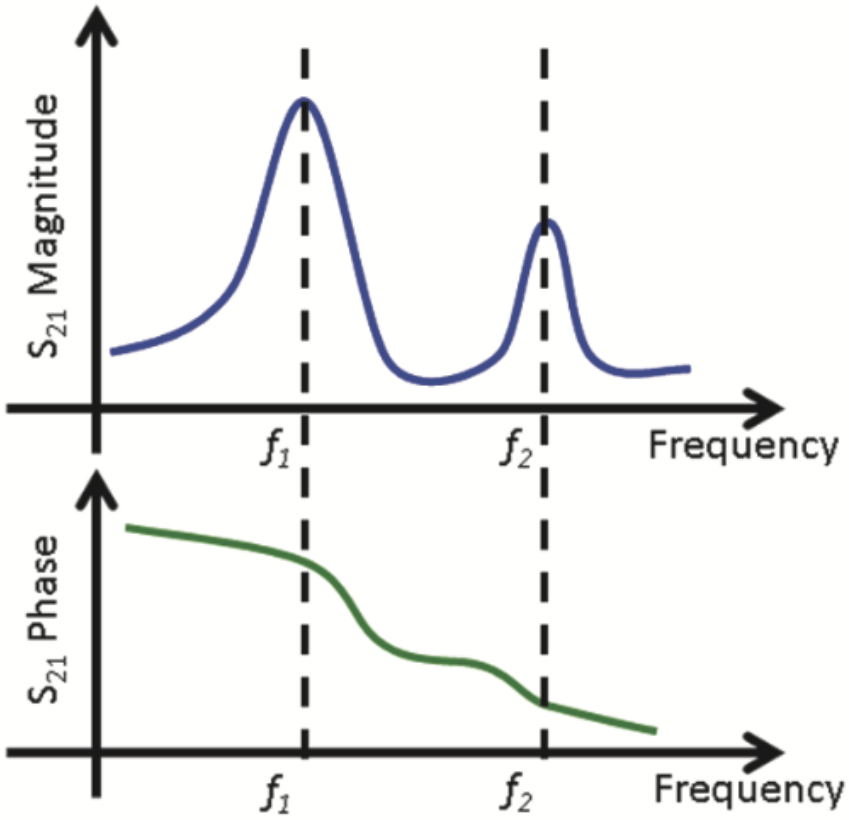


Figure 2.3: An arbitrary S21 frequency response.

2.2.1 Evaluation of Fundamental Frequency

Figure 2.3 illustrates an arbitrary S_{21} frequency response with two largest peaks at frequencies of f_1 and f_2. For the input signal, $x(t)$, to effectively pass through the system in Fig. 3, two of its harmonic frequencies must lie on f_1 and f_2. Therefore, the fundamental frequency, f_0, must be such that f_1 and f_2 become integer multiples of f_0. As an example, for integer numbers m and k, we have $f_1 = m \cdot f_0$ and $f_2 = k \cdot f_0$. Since no other harmonics besides m and k can pass the system, only a_m and a_k must be non-zero to avoid unnecessary power consumption, as seen in equation (2.5). Therefore, in this example, $x(t)$ can be written as:

$$x(t) = a_m \sin(2\pi m f_0 t + \phi_m) + a_k \sin(2\pi k f_0 t + \phi_k). \tag{2.6}$$

Note that if S_{21} has a flat response, multiple harmonics can be chosen within the given bandwidth as long as they can be assigned to integer multiples of a fundamental frequency.

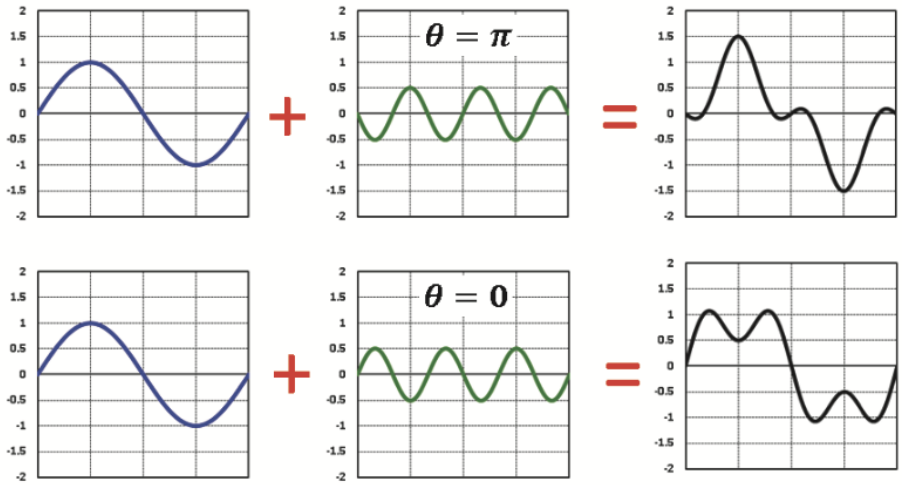


Figure 2.4: Demonstration of constructive/destructive phases for harmonic 3.

2.2.2 Evaluation of Harmonic Phases

For the output signal, $y(t)$, to have the maximum peak-to-peak value, the harmonic signals must be aligned such that it produces constructive interference. As an example, Fig. (2.4) illustrates the interference between harmonics 1 and 3. As shown, when $\theta = 0$ the peaks of both harmonics are aligned, which produces a maximum possible peak-to-peak signal when added together. However, when $\theta = \pi$, the peaks of the two harmonics are destructive, resulting in a lower peak-to-peak value. Therefore, the phase of harmonic 3 must be π with respect to harmonic 1. A similar analogy can be applied for harmonics 5 and 1, where the two harmonics will add constructively, only when the phase of harmonic 5 is 0 with respect to harmonic 1, as shown in Fig. (2.5).

In summary, assuming the phase of the first harmonic is 0, the optimal phase is π for harmonics 3, 7, 11, 15, ..., and the optimal phase is 0 for harmonics 5, 9, 13, 17, ..., in order to align the harmonics constructively. Under this condition, the overall

peak-to-peak value of the signal becomes:

$$y_{\text{peak-to-peak}} = 2 \times \sum_{\text{Odd } n} b_n, \tag{7}$$

where b_n is the amplitude of harmonic n as described in equation (2.2). While the odd harmonics affect the high and low peaks the same way (either constructive or destructive), the impact of even harmonics is asymmetric. For example, Fig. (2.6) illustrates the interference between harmonics 1 and 2, where the interference is constructive for the high peak but destructive for the low peak when $\theta = -\frac{\pi}{2}$. As a result, the even harmonics do not contribute to improving the output peak-to-peak and therefore can be eliminated.

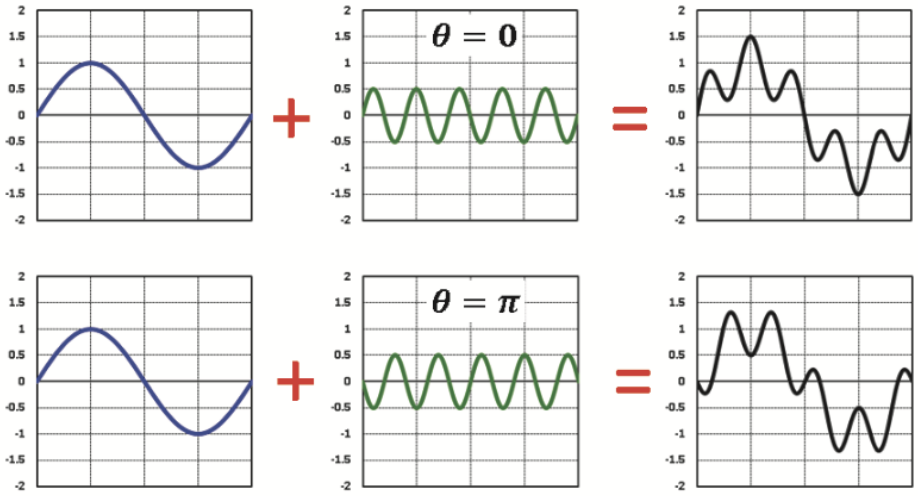


Figure 2.5: Demonstration of constructive/destructive phases for harmonic 2.

2.2.3 Evaluation of Harmonic Amplitudes

Assuming the phases of the harmonics in $y(t)$ are aligned constructively and using equations (2.3) and (7), the peak-to-peak value of the output signal can be calculated by

$$y_{p2p} = 2 \times \sum_{n=1}^{N} b_n = 2 \times \sum_{n=1}^{N} a_n \cdot |S_{21}(2\pi n f_0)|, \tag{2.7}$$

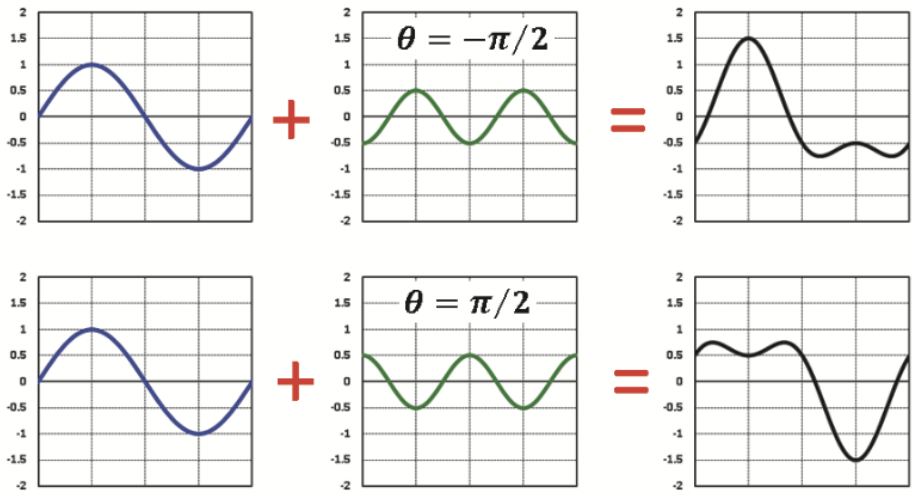


Figure 2.6: Demonstration of constructive/destructive phases for harmonic 5.

where a_n and b_n are the harmonic amplitudes of the input and output, respectively. Also, $a_n = b_n = 0$ for even harmonics, n. Moreover, using Equation (2.5), the constant input power constraint dictates that

$$P \times 2Z_0 = \sum_{n=1}^{N} a_n^2 = \text{Constant} \tag{2.8}$$

Equation (2.7) can be seen as the inner product of vector $\mathbf{a}$ and S_{21}, under the condition in Equation (2.8) that the sum of a squares is constant. It can be mathematically proved that under the given constraint, the inner product, y_{p2p} is maximized when the vector $\mathbf{a}$ is fully correlated with vector S_{21}, or

$$a_n = \gamma |S_{21}(2\pi n f_0)|, \tag{2.9}$$

where γ is the scaling factor and n is the odd harmonics, associated with the peak frequencies of the S_{21}, for instance, as represented by m and k in Equation (2.6). The scaling factor, γ, is used to normalize the amplitudes of harmonics for a given input RF power, P_{Source}, as shown in Equation (2.5).

2.3 A PRACTICAL EXAMPLE

To demonstrate an application of the waveform optimization analysis, a practical example is presented in this section. The measured S_{21} frequency response of an RF circuit is shown in Figure 2.7, where there are three largest peaks at 136 MHz, 370 MHz, and 1.07 GHz. Given these three frequencies, the fundamental frequency of $f_0 = 19.48$ MHz puts the three peaks at harmonics 7, 19, and 55, respectively. Considering that all three harmonics are odd numbers, they all can contribute to the optimal waveform.

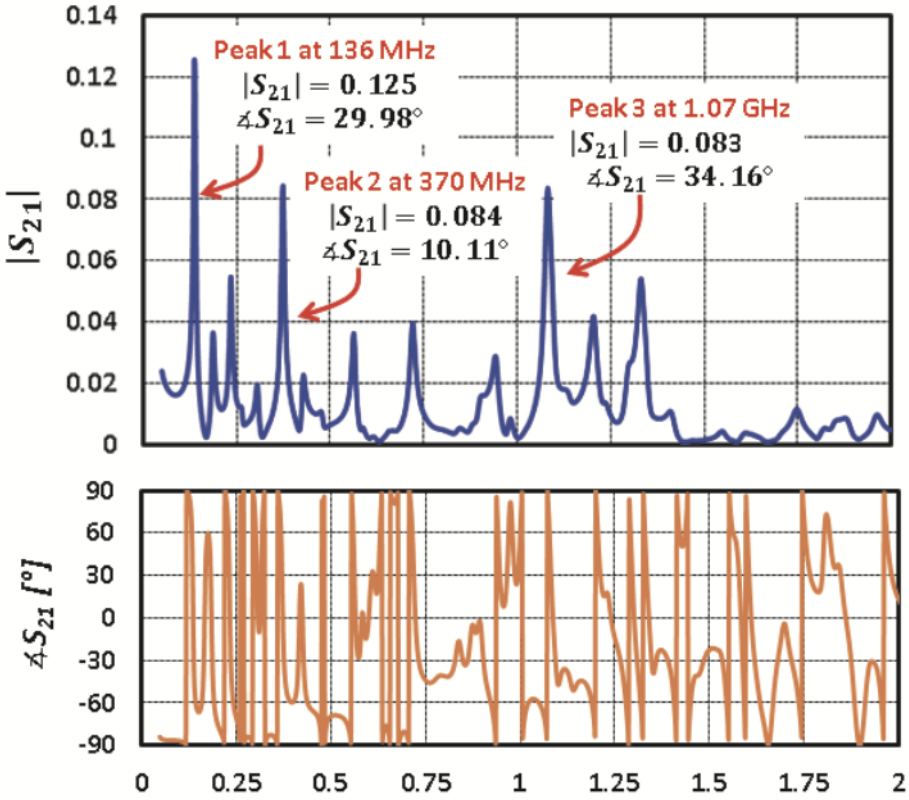


Figure 2.7: A measured example of S21 frequency response.

To ensure that all three harmonics are aligned to constructively produce the largest peak-to-peak output signal and based on the rule described in Section III-B, the phase of harmonics 7, 19, and 55 in the output must be $\theta = 0$, π, and π, respectively. Consid-

ering that the phase of S_{21} for harmonics 7, 19, and 55 are 29.98°, 10.11°, and 34.16°, respectively Fig.2.7, Equation (2.3) yields the input phase of $\phi = -29.98°$, 169.89°, and 145.84° for each harmonic, respectively. Finally, the amplitude of each harmonic at the input can be calculated using Equation (2.9), which results in the amplitudes of 0.125γ, 0.084γ, and 0.083γ for harmonics 7, 19, and 55, respectively. Assuming the input RF power is 10 dBm and the system's radiation impedance is 50 Ω, the scaling factor, γ, becomes 5.806 V. Therefore, the optimum input waveform signal to obtain maximum output peak-to-peak using three harmonics is:

$$\begin{aligned} x(t) = {} & 0.726\sin(2\pi \times 7 \times 19.48 \times 10^6 t - 29.98°) \\ & + 0.488\sin(2\pi \times 19 \times 19.48 \times 10^6 t + 169.89°) \\ & + 0.485\sin(2\pi \times 55 \times 19.48 \times 10^6 t + 145.84°). \end{aligned} \tag{2.10}$$

Using equation (2.2), the output signal can be calculated as

$$\begin{aligned} x(t) = & y(t) = 0.0907\sin(2\pi \times 7 \times 19.48 \times 10^6 t + 0°) \\ & + 0.0410\sin(2\pi \times 19 \times 19.48 \times 10^6 t + 180°) \\ & + 0.0404\sin(2\pi \times 55 \times 19.48 \times 10^6 t + 180°). \end{aligned} \tag{2.11}$$

The input and output waveforms, $x(t)$ and $y(t)$, for this example are shown in Fig. 2.8, which results in the peak-to-peak output signal of 0.317 V. To compare this result to a single-tone sinewave with the same input power of 10 dBm, the output signal is calculated and shown in Fig. 2.9, where the peak-to-peak of the output with a single tone injection is 0.249 V. Therefore, in this example, the single tone injection model results in a 27% underestimation of the peak-to-peak output signal, compared to a three harmonic signal injection with the same RF power. The coupling effectiveness impact is even more pronounced with added harmonics.

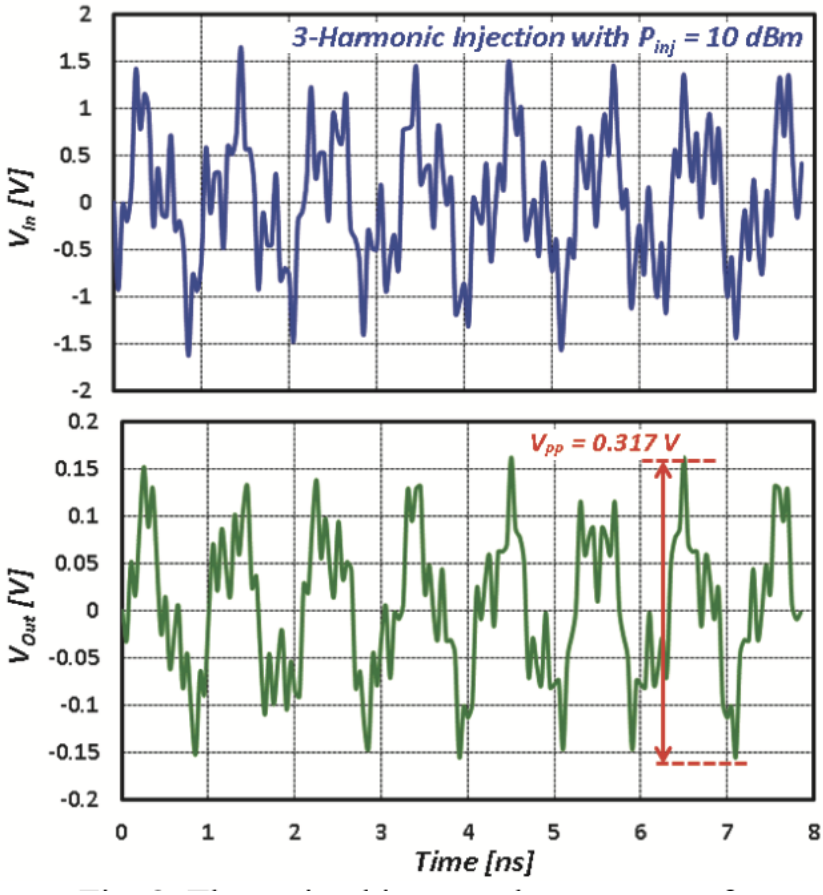


Figure 2.8: The optimal input and output waveforms.

2.4 CONCLUSION

It is shown that the modulation shape of the EMI waveform can drastically influence its impact on digital electronics. For our example, the coupling effectiveness of an optimally modulated waveform is shown to be 27% higher than that of a single-tone EMI, as measured by the peak-to-peak voltage induced in the interconnect. This presented analytical solution in finding the optimal waveform for maximum coupling can be utilized by EMC engineers to develop robust shielding margins for various classes of EMI signals beyond a single-tone approximation.

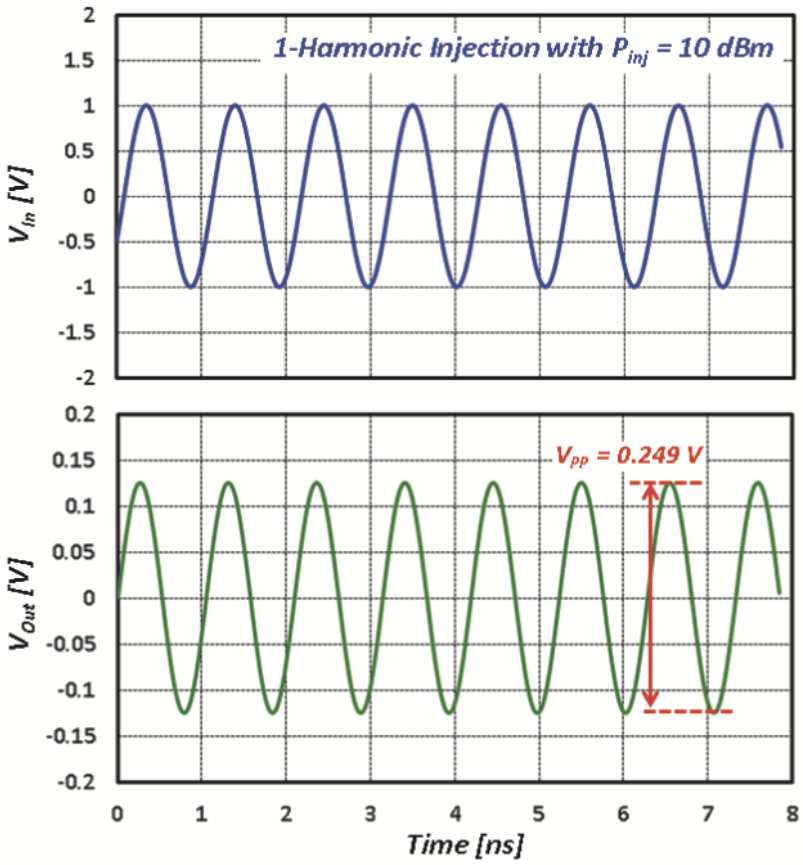


Figure 2.9: The single tone input and output waveforms.

Chapter 3

Advancing EMC Analysis with GAN-Driven Signal Classification and Waveform Modulation

The ubiquity of digital electronic devices in contemporary society is a testament to the rapid advancement of technology. However, as the density of these devices in both commercial and personal spaces increases, the issue of Electromagnetic Compatibility (EMC) becomes a focal point of concern. Ensuring that electronic systems operate harmoniously within their electromagnetic environment, without causing or falling prey to interference, is essential for maintaining the reliability and functionality of our increasingly interconnected world.

Electromagnetic Interference (EMI), especially from external Radio Frequency (RF) sources, poses a significant threat to the integrity of digital electronic interconnects. These interconnects are the lifelines of electronic systems, enabling components to communicate and function as a cohesive unit. As such, understanding and mitigating the adverse effects of EMI is of paramount importance. Traditional analyses of EMI have predominantly focused on the impacts of Continuous Wave (CW) signals, providing limited insights into scenarios where waveform modulation significantly influences coupling efficiency and system behavior.

In this pioneering study, we present an approach that integrates waveform modulation into EMC analysis, moving beyond the confines of CW signal methodologies. Our research builds upon and extends the findings presented in Impact of Waveform Modulation from Electromagnetic Interference Sources on Coupling to Digital Electronic Interconnects [57], and draws on the chaotic pulse-width modulation (PWM) technique of Kim and Song [118] and the spread-spectrum EMI analysis by Crovetti and Musolino [42]. By

exploring modulation techniques, we reveal how specific waveform modulations alter EMI coupling efficiency, surpassing traditional CW approaches. Our work is characterized by methodological innovations that include the utilization of S-parameter data, the application of consistent input RF power, and a concentration on steady-state responses, thus obviating the need for time-domain conversion. Furthermore, we introduce an analytical solution that integrates waveform modulation into the EMI coupling analysis, revealing that specific modulations markedly enhance coupling efficiency compared to their CW counterparts.

A cornerstone of our work is the introduction of a cutting-edge image-based classification system designed to assess signal safety. Leveraging advanced image processing techniques, this system evaluates representations of user-supplied signals, categorizing them as 'safe' or 'unsafe' based on their steady-state response characteristics. To further enhance classification accuracy, we employ a Generative Adversarial Network (GAN) trained on 'safe' and 'unsafe' signal images. The GAN effectively learns to generate realistic signal patterns, enabling the system to better distinguish between safe and unsafe states, especially in scenarios where empirical data may be limited. Through rigorous validation against various image manipulations, we affirm the resilience and precision of our GAN-augmented classification model.

By incorporating waveform modulation into EMC analysis, this study provides practical insights for addressing challenges in high-RF-interference environments and extends the framework established by previous researchers in this domain. This advancement not only elevates the analytical practices within the EMC field but also plays a crucial role in fortifying the reliability and security of electronic devices amidst environments rife with RF interference.

3.1 Related Work

3.1.1 EMC and Digital Interconnects

The related works in the domain of EMC and digital interconnects encompass various innovative analyses and methodologies. One study delves into the decompositional electromagnetic signal analysis of digital interconnects [138], emphasizing the necessity of high-quality models for precise time-domain analysis and the need for broadband materials in transmission lines. Another work highlights the effect of CW electromagnetic interference on radar frequency equipment [49]. It specifically addresses the linear and nonlinear responses of radar systems to CW interference, detailing the experimental setup and results of CW electromagnetic effect tests on radar equipment. The study provides insights into the basic properties of frequency equipment sensitive to different frequencies of electromagnetic radiation and the fundamental law of electromagnetic interference effect. Additionally, research on the effects of Spread Spectrum (SS) modulated EMI on digital data links underlines how SS-modulated EMI can exacerbate EMI-induced capacity loss compared to non-modulated EMI, particularly under specific conditions.These studies collectively contribute to the evolving understanding and mitigation strategies of EMC challenges in electronic systems. [42]

3.1.2 RF-Driven EMI Analysis

The exploration of RF-driven EMI analysis ventures beyond the conventional Continuous Wave (CW) centric methodologies, marking a significant shift towards addressing the intricate challenges of electromagnetic interference in digital and RF systems. This subsection elaborates on seminal contributions that have advanced the field.

Active EMI Reduction Using Chaotic Modulation: A notable study, "Active EMI Reduction Using Chaotic Modulation in a Buck Converter with Relaxed Output LC

Filter," presents a groundbreaking approach to EMI mitigation in DC-DC buck converters—a staple in portable electronic devices. By harnessing chaotic pulse-width modulation (PWM) techniques, this research, published in MDPI, offers a sophisticated method to curtail EMI effectively without the reliance on external filtering mechanisms, thereby aligning with the compact design requirements of SoC or SiP applications. [118]

Interference of Spread-Spectrum EMI and Digital Data Links: In another critical piece of research, "Interference of Spread-Spectrum EMI and Digital Data Links under Narrowband Resonant Coupling," the focus shifts to the effects of spread-spectrum (SS) modulation on EMI dynamics and its implications for digital communication systems. Published in Electronics, this study illuminates the profound impact of SS modulation, especially those leveraging chaos theory, on the modulation of EMI characteristics. It posits that SS-modulated EMI may exacerbate EMI-induced capacity loss under certain scenarios, thus challenging the traditional perceptions of SS modulation's efficacy in EMI mitigation. [42]

Our research extends these foundational studies, offering a comprehensive RF-centric analysis of EMI effects on digital interconnects and introducing an image-based classification system for signal safety assessment. By emphasizing modulation effects in EMI coupling, we distinguish our work from traditional EMC methodologies. These innovative approaches mark a significant progression in EMC, setting the stage for more robust and secure electronic systems in RF-intense environments.

3.2 Dataset

In our study, we curated a dataset founded on waveform optimization analysis of a test case S-parameter. The initial data were obtained from S-parameter measurements, specifically focusing on S_{21} as shown in the frequency response plot. This parameter

was chosen for its relevance in representing the forward transmission between two ports in a network, which is crucial for our study on signal integrity and waveform optimization.

The optimization process was governed by a Fourier series [109], [110] aimed at achieving the maximum peak-to-peak output. The algorithm adjusted the input waveform by selectively incorporating harmonics in this case, harmonics 1, 19, 53 and 189 and by tuning the input power to a specified level of 10 dBm. These harmonics were specifically chosen based on their alignment with the resonance peaks of the S_{21} frequency response. This alignment ensures constructive interference, thereby enhancing the signal amplitude. These harmonics also satisfy the requirement of being integer multiples of the fundamental frequency, which is critical for effective coupling. These criteria were established to ensure optimal signal enhancement through waveform modulation. [57]
The raw data underwent a series of processing steps to filter, align, and normalize the waveforms, resulting in a dataset of optimal input waveforms. Each waveform was then passed through the test case network, where the output waveforms were measured and analyzed. The key metric of interest was the peak-to-peak voltage of the output signal, which, for example, was found to be 0.55 V for the optimized case.
The dataset is structured to include both the optimized input waveforms and the resultant output waveforms, along with their corresponding time-domain representations. The dataset allows for a comprehensive analysis of the effects of waveform optimization on signal characteristics, providing a peak-to-peak voltage measurement for each waveform to facilitate comparative studies.
To ensure the reliability and validity of the dataset, each optimized waveform was subjected to a verification process that compared the empirical results with theoretical expectations based on the S-parameter model. This rigorous validation approach confirms the dataset as a reliable basis for further investigation into advanced waveform optimization techniques.

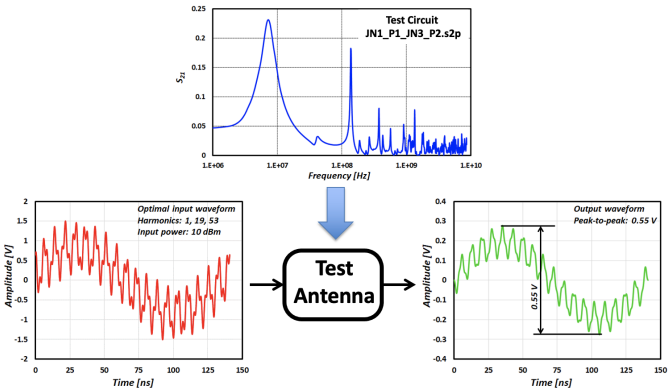


Figure 3.1: Overview of the constructed dataset for the classification of electrical signals. The dataset includes labeled images representing 'safe' and 'unsafe' signals based on peak-to-peak voltage thresholds, with a maximum value of 0.55 V and classification boundary at 0.32 V. The dataset comprises 1500 images for each category, facilitating robust training and evaluation of the classification model.

In this study, we have constructed a dataset essential for the classification of electrical signals (see Figure 3.1). This dataset was developed following the analytical solution that established the maximum peak-to-peak voltage value at 0.55 V. For classification purposes, signals were labeled as 'unsafe' if their peak-to-peak voltage exceeded 0.32 V (Label 0) and 'safe' if their value was 0.32 V or less (Label 1). Our dataset comprises a balanced set of 1500 safe images and 1500 unsafe images, providing a solid foundation for training the GAN-based classification model.

To ensure robust model evaluation, the dataset in classification was divided into 80% training/validation and 20% testing sets. To achieve a fair comparison across models, we applied consistent training and validation splits using a K-fold cross-validation strategy with 3 folds for GoogLeNet and ResNet50 and 4 folds for CNN and VGG16. This approach ensures each model was rigorously evaluated on multiple splits, minimizing bias from a single train/test division.

By training our GAN on the dataset, we enable the generation of realistic signal patterns that aid in distinguishing between safe and unsafe signals.

3.3 Methodology

3.3.1 GANs: Training and Dataset Usage:

In traditional machine learning methods, datasets are typically split into training and test sets to evaluate the model's performance. This process helps ensure that the model generalizes well to unseen data. However, with Generative Adversarial Networks (GANs), this approach differs slightly. Since GANs aim to generate new, realistic data rather than solely predict outcomes, the dataset is only used for training; a separate test set is not required.

GANs consist of two networks, the Generator and the Discriminator, which train together in a competitive manner. The Generator learns to create data samples that resemble the training data, while the Discriminator tries to distinguish between real and generated data. This adversarial process pushes both networks to improve iteratively. As a result, the concept of a test set does not apply in the same way as it does with traditional models. Instead, the quality of the GAN is assessed through the realism and diversity of the generated data.

In this study, a GAN was trained on a dataset comprising labeled signal categorized as 'safe' or 'unsafe.' The GAN generated realistic signal patterns that enhanced the robustness and generalization of our classification model.

The architecture of our Generative Adversarial Network (GAN) consists of two neural networks: a Generator and a Discriminator, which engage in a competitive training process. The Generator transforms random noise into synthetic data samples, while the Discriminator attempts to differentiate between real and synthetic data. Through adversarial training, the Generator progressively improves its ability to generate data that closely resembles real samples, while the Discriminator enhances its classification accuracy. Figure 3.2 illustrates the overall structure of our GAN model.

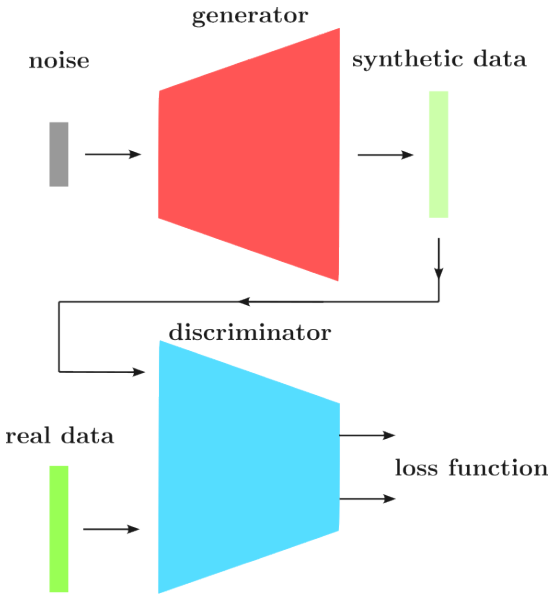


Figure 3.2: Architecture of a Generative Adversarial Network (GAN).

3.3.2 Signal Processing Techniques: Shifting and Scaling

The classification of signals as 'safe' or 'unsafe' is crucial in systems where the integrity of the signal is paramount for dependable functioning. This study focuses on assessing how scaling and shifting signals influence their safety status, a key consideration for maintaining operational reliability. We employ two main signal processing techniques: scaling and shifting. These methods are instrumental in modifying signal characteristics, which is essential for our analysis. Specifically, we aim to evaluate the safety of signals based on established criteria, determining the impact of these alterations on their classification. This approach allows us to understand better the relationship between signal manipulation and its implications for system safety.

- **Shifting Signals**

  Signal shifting involves displacing a signal along the time axis, a technique crucial for temporal analysis and comparison. This operation is mathematically represented as $S'(t) = S(t + \Delta t)$, where $S(t)$ is the original signal, $S'(t)$ is the shifted

signal, and Δt is the shift amount. In our implementation, we use Python's 'np.roll' function from the NumPy package to cyclically shift signal values by a specified amount, enabling controlled adjustments in the signal's temporal alignment. The shift amount, denoted as 'shift amount', is predefined and applied uniformly across all signals.

This process is instrumental in examining phase variations and temporal alignments within signal components, aiding in our comprehensive safety analysis.

- **Scaling Signals**

  Scaling, on the other hand, adjusts the amplitude of the signal. This operation is critical for normalizing signal intensity or adjusting signal strength for safety assessments. Mathematically, it's expressed as $S''(t) = \alpha \cdot S(t)$, where $S''(t)$ represents the scaled signal, and α is the scaling factor. The scaling factor, 'scale factor', is a crucial parameter in our study, influencing the safety classification of the signal.

  Signals are deemed 'safe' or 'unsafe' based on their peak-to-peak voltage ('Vp-p'), calculated after scaling. A signal is classified as 'safe' if its 'Vp-p' after scaling remains below a predetermined threshold, specifically 0.32 in our criteria. This classification underscores the significance of the scaling factor in our safety analysis, as it directly affects the 'Vp-p' value and, consequently, the safety label assigned to each signal.

  Our experimental setup utilized a 'scale factor' of 0.5 for demonstration purposes, illustrating how even a moderate scaling can alter signal characteristics significantly enough to impact safety classifications. This aspect of our methodology highlights the delicate balance between signal integrity and safety requirements, underscoring the importance of meticulous parameter selection in signal processing applications.

3.4 Deep Learning for the Discrimination of Safe and Unsafe Signals

Our research initiative focused on the development of an intelligent framework capable of categorizing signals into 'safe' or 'unsafe' based on their peak-to-peak voltage values. The core objective of this study is to leverage deep learning models to automate the process of signal analysis that traditionally requires manual examination and threshold-based decision-making.

Our approach is grounded on the premise that signals carrying a peak-to-peak voltage exceeding a certain safety threshold are indicative of potential hazards and therefore must be classified as 'unsafe.' Conversely, signals that fall below this threshold are considered 'safe' and generally do not pose a risk. The critical threshold has been analytically determined to be 0.55 V, with a further classification boundary set at 0.32 V.

To operationalize this, we deploy several advanced convolutional neural network (CNN) architectures, including ResNet50, GoogleNet (Inception), VGG16, and custom CNNs. These networks have been trained to recognize and differentiate between 'unsafe' and 'safe' signals using labeled image data. Images of signals that exhibit a peak-to-peak voltage greater than 0.32 V are labeled as 'unsafe' (Label 0). In contrast, images representing signals with a peak-to-peak voltage of 0.32 V or less are labeled as 'safe' (Label 1).

3.4.1 Architectural Insights: A Comparative Evaluation of CNNs in Image Classification

- **CNN:** In a convolutional layer, the network performs a convolution operation between the input data and a set of learnable filters or kernels. This operation is mathematically represented as

$$F(x, y) = (I \cdot K)(x, y) = \sum_{m} \sum_{n} I(m, n) \cdot K(x - m, y - n), \tag{3.1}$$

where F is the output feature map, I is the input image, and K is the kernel. The purpose is to highlight features in the input such as edges or textures. Pooling layers follow convolutional layers and are used to reduce the spatial dimensions (width and height) of the input volume. The most common form is max pooling, where the maximum value within a certain window is taken as the output. This operation is described by

$$P(x, y) = \max_{(a,b)\in W} I(a, b), \tag{3.2}$$

with W being the window over the input I, and P the pooled output, effectively down-sampling the input feature map.Fully connected layers, appearing towards the end of the network, aim to classify the image based on the features extracted by the convolutional and pooling layers. Each neuron in a fully connected layer has full connections to all activations in the previous layer, and the output is computed as a weighted sum of these activations, followed by a non-linear activation function. The entire network is trained end-to-end with a backpropagation algorithm, using a loss function suitable for the task (e.g., cross-entropy for classification) to update the weights in a way that minimizes the loss. During this process, the network learns to emphasize features that are important for distinguishing between the different classes in the task at hand, effectively learning a hierarchy of increasingly complex features at each layer.

In essence, CNNs automate the feature extraction process, eliminating the need for manual feature design and allowing the network to adaptively learn the features directly from the data, leading to state-of-the-art performances in many visual tasks. [88], [82], [160]

- **ResNet50:** is short for Residual Network with 50 layers, is a deep convolutional neural network architecture designed to address the vanishing gradient problem during training. It introduced the concept of residual blocks, which allow for the training of very deep networks.

  In a standard neural network, as the number of layers increases, it becomes challenging to optimize the network effectively. The vanishing gradient problem occurs because gradients during backpropagation become extremely small as they are propagated back through many layers, making it difficult to update the weights of earlier layers effectively.

  ResNet50 tackles this issue by introducing residual blocks. These blocks contain skip connections, also known as shortcut connections, that allow the gradient to flow directly without diminishing significantly. The output of a residual block can be mathematically represented as:

$$F(x) = \mathrm{H}(x) + x, \tag{3.3}$$

  where:

  $F(x)$ is the output of the residual block.

  $H(x)$ represents the transformation applied by the block's layers.

  x is the input to the block.

  By adding the input x to the transformed output $H(x)$, ResNet enables the network to learn residual functions, i.e., the difference between the desired output and the current prediction. This makes it easier to train very deep networks effectively.

ResNet50 consists of multiple residual blocks, each containing several convolutional layers and batch normalization. These blocks progressively extract hierarchical features from the input data. The final fully connected layers of ResNet50 are responsible for the classification task, producing the output probabilities for various classes.

In summary, ResNet50 is a deep neural network architecture that overcomes the vanishing gradient problem by using residual blocks with skip connections, allowing for the training of very deep networks. These residual blocks learn the difference between the desired output and the current prediction, making it easier to optimize the network during training. [153], [165], [135]

- **GoogLeNet** is a deep convolutional neural network architecture designed for image classification tasks. It introduced the concept of inception modules, which significantly improved model efficiency and performance.

  Inception modules are the key innovation in GoogLeNet. Instead of using a single convolutional layer, they employ multiple convolutional operations of different filter sizes (1x1, 3x3, and 5x5) and pooling operations (max pooling) in parallel. These operations capture features at various scales and help the network learn a richer set of features. The output of an inception module can be mathematically represented as:

$$O = \text{concat}(\text{C}_{1x1}, \text{C}_{3x3}, \text{C}_{5x5}, \text{MxM}), \tag{3.4}$$

where:

O is the concatenated output.

C_{1x1} represents the result of a 1x1 convolution operation.

C_{3x3} represents the result of a 3x3 convolution operation.

C_{5x5} represents the result of a 5x5 convolution operation.

MxM represents the result of a max-pooling operation.

The inception module allows GoogLeNet to efficiently capture both fine-grained and large-scale features. Additionally, GoogLeNet uses 1x1 convolutions to reduce the number of channels and computational complexity, making the network deeper without significantly increasing the computational cost.

GoogLeNet also incorporates auxiliary classifiers at intermediate layers during training. These auxiliary classifiers provide additional gradient paths and regularization, helping with the vanishing gradient problem and improving convergence.

Overall, GoogLeNet's inception modules and auxiliary classifiers contribute to its effectiveness in image classification tasks, achieving state-of-the-art results with a relatively efficient architecture. [157], [154], [71]

- **VGG16** is a convolutional neural network architecture notable for its deep structure characterized by an increased number of convolutional layers. The architecture's key feature is the use of very small (3x3) convolutional filters throughout the network, which allows it to capture fine-grained details from the input image.

  The VGG16 network consists of 16 layers with trainable weights: 13 convolutional layers and 3 fully connected layers. Here's an explanation of how it works, suitable for inclusion in your paper:

VGG16's convolutional layers use small (3x3) filters with a stride of 1 pixel and padding to preserve the spatial resolution of the image. Given an input image or feature map X, the convolution operation in these layers is defined as:

$$(F \cdot X)(i,j) = \sum_{m=0}^{2} \sum_{n=0}^{2} \sum_{k=1}^{K} F(m,n,k) \cdot X(i+m-1, j+n-1, k), \tag{3.5}$$

where F is the 3x3 filter, K is the number of input channels, and i, j index the spatial position on the output feature map.

Each convolutional layer in VGG16 is followed by a Rectified Linear Unit (ReLU) activation function to introduce non-linearities into the model, making it capable of learning more complex patterns. The ReLU function is applied element-wise:

$$\text{ReLU}(x) = \max(0, x) \tag{3.6}$$

VGG16 uses max pooling layers to reduce the spatial dimensions (width and height) of the output feature maps from the convolutional layers. A typical max pooling operation with a 2x2 window and stride of 2 is applied:

$$\text{MaxPooling}(X)(i,j) = \max_{0 \leq m,n < 2} X(2i+m, 2j+n) \tag{3.7}$$

This operation reduces the dimensionality of the feature maps, decreasing the computational complexity and the number of parameters.

After several convolutional and max pooling layers, VGG16 has three fully connected layers. The first two have 4096 nodes each, and the third performs classification, typically with 1000 nodes for ImageNet. The operation in these layers for an input vector x is:

$$y = Wx + b, \tag{3.8}$$

where W is the weight matrix, b is the bias vector, and y is the output vector.

The final layer in VGG16 uses the softmax function to classify the input image into various categories. For an input z (output of the last fully connected layer), the softmax function is:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{3.9}$$

where K is the number of classes, and i indexes a particular class. This function outputs a probability distribution over K classes.

VGG16 is trained using backpropagation and an optimization algorithm like stochastic gradient descent (SGD). The network employs a loss function, typically cross-entropy for classification tasks, to calculate the error in prediction. The weights are updated to minimize this loss over multiple training iterations. [38], [170]

3.4.2 GAN Architecture and Training

Generative Adversarial Networks (GANs) [65] are models that can learn the data-generating distribution of the training set and produce new samples from it. As such, GANs are used to generate data that resembles a given dataset. The main component of a GAN is its Generator network, which is capable of mapping random noise to desirable data.

In the training phase, a Discriminator network classifies the Generator's output as either belonging to the training set or not. As the training progresses, the samples produced by the Generator become similar to those of the training set to such a degree that the Discriminator is unable to distinguish between them.

The original GAN suffers from training stability issues. The Wasserstein GAN (WGAN) was proposed to alleviate those issues [20]. The WGAN leverages the Wasserstein distance instead of the Jensen-Shannon divergence to measure the difference between the generated and real data distributions. Furthermore, it enforces the Lipschitz continuity

of the gradients in the Discriminator network through weight clipping.

Further improvement in the training process was made possible by the introduction of the Gradient Penalty (WGAN-GP) [66]. In this variant of the WGAN, Lipschitz continuity is imposed through a penalty on the norm of the gradient rather than weight clipping. The WGAN-GP implemented in this work is based on the one described in our previous work [120].

Both the Generator and the Discriminator are based on fully connected layers. We train a separate WGAN-GP for each category of 'safe' and 'unsafe'.

3.5 Results

The objective of our experimental analysis was to evaluate the efficacy of various deep learning models in classifying signal images as safe or unsafe. This evaluation entailed a comparative analysis of four advanced models—CNN, ResNet50, GoogLeNet, and VGG16—across standard performance metrics. In addition, we rigorously tested the robustness of these models against image transformations, specifically shifting and scaling, to ensure the reliability of signal safety classifications under varying image conditions.

3.5.1 Confusion Matrix Analysis

To provide a deeper understanding of each model's classification capabilities, we analyzed the confusion matrices for CNN, GoogLeNet, ResNet50, and VGG16. The confusion matrices illustrate the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classifications, offering insight into each model's performance.

Performance Metrics

Based on the confusion matrices in Figure 3.3, we calculated key performance metrics for each model, including accuracy, precision, recall, and F1-score, as shown in Table 3.1.

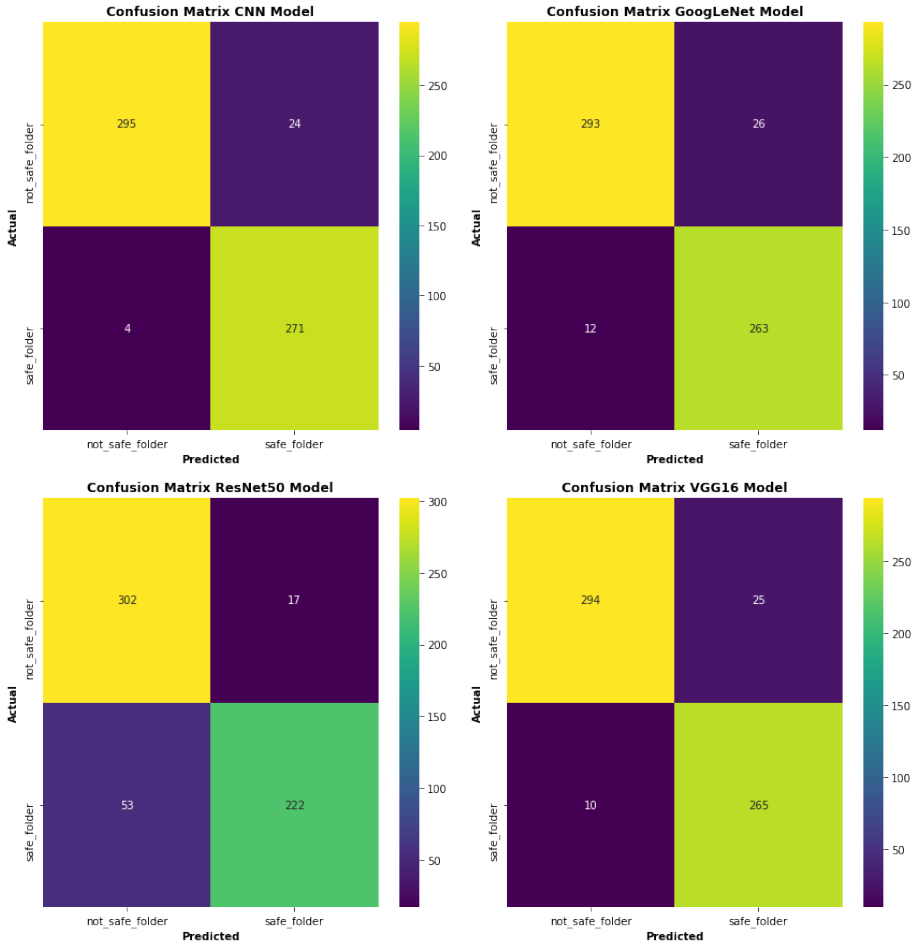


Figure 3.3: The confusion matrix for the different models.

These metrics provide a detailed understanding of each model's strengths and weaknesses in the classification task.

The results demonstrate that CNN achieves the highest F1-score, indicating its superior capability in balancing precision and recall for signal classification. VGG16 follows closely, with strong performance across all metrics. GoogLeNet also shows robust results, while ResNet50, despite its high precision, has a lower recall due to a higher false negative rate. This analysis highlights the trade-offs between model complexity and performance in signal safety classification.

Table 3.1. Performance metrics for deep Learning models

Model	Accuracy	Precision	Recall	F1 Score
CNN	0.9443	0.9187	0.9854	0.9510
GoogLeNet	0.9314	0.9106	0.9564	0.9330
ResNet50	0.8398	0.9288	0.8072	0.8635
VGG16	0.9367	0.9137	0.9636	0.9380

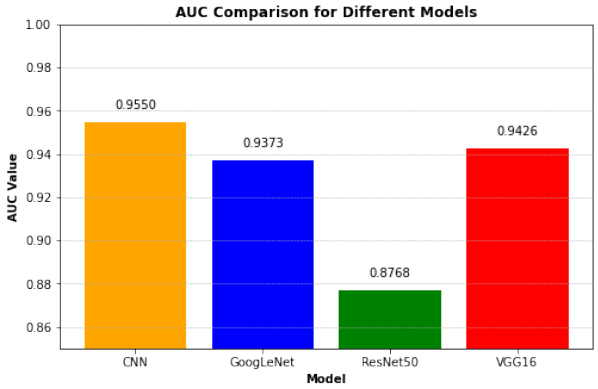


Figure 3.4: A comparison of the Area Under the Curve (AUC) scores for different deep learning models.

AUC Analysis

To further assess the classification performance of the models, we compared the Area Under the Curve (AUC) values for each model: CNN, GoogLeNet, ResNet50, and VGG16. AUC is a key performance metric that evaluates a model's ability to distinguish between positive and negative classes, with higher values indicating better discriminatory power.

As shown in Figure 3.4, the CNN model achieved the highest AUC score of 0.9550, indicating its superior capability in distinguishing between 'safe' and 'unsafe' signals. VGG16 followed with an AUC score of 0.9426, reflecting strong performance. GoogLeNet achieved an AUC of 0.9373, demonstrating robust classification capabilities. ResNet50, while achieving a slightly lower AUC of 0.8768, still demonstrates effective classification. The AUC results align with the models' performance in other metrics, as shown in Table 3.1, and provide further insight into each model's ability to handle signal classification

tasks.

3.5.2 Image Classification Based on Shifting and Scaling

Our analysis extended to testing the resilience of the classification system when subjected to image transformations. Our system prompts users to upload images representing the signals. Upon receipt, our algorithm analyzes these images, utilizing a dataset trained to recognize visual patterns correlated with safety classifications. The algorithm is designed to maintain the classification consistency for shifted and scaled versions of the signals.

Shifting: The first transformation involves a time shift in the original signal 3.5. The shifted signal displays a clear deviation in phase, as evidenced by the offset between the input and output signals over time 3.6. This phase shift can lead to misalignment in time-sensitive applications, where the precise timing of signal features is critical. The deviation introduced by the shifting process can result in misinterpretation or errors in signal processing, leading to the classification of the shifted signal as 'unsafe.' This designation underscores the potential hazards associated with using this transformed signal in applications where timing accuracy is paramount.

Scaling: The second transformation involves scaling the amplitude of the original signal 3.5. The scaled signal maintains the original frequency and phase characteristics but alters the amplitude. Upon examination of the scaled signals 3.7 based on the scale factor 0.5, it is observed that the input and output maintain a consistent pattern, with the output mirroring the input's dynamics across the time domain. The scaling process does not introduce any phase shift or distortion in frequency, which implies that the integrity of the signal is preserved. Consequently, the scaled signal is classified as 'safe,' indicating its suitability for further processing or interpretation without the risk of misrepresentation. However, it's crucial to differentiate between 'safe' and 'unsafe' adjustments, which

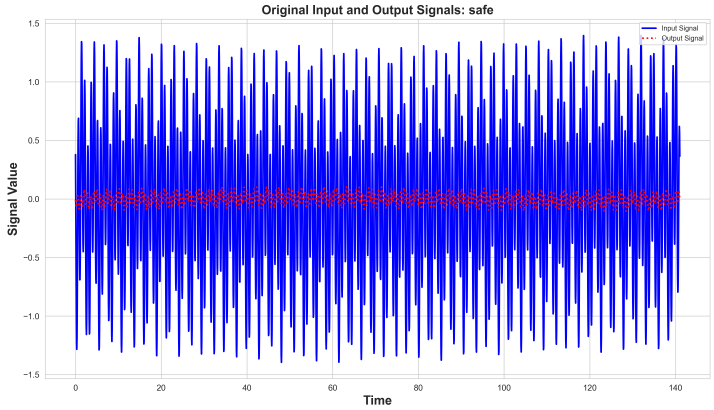


Figure 3.5: The original input signal and its corresponding output.

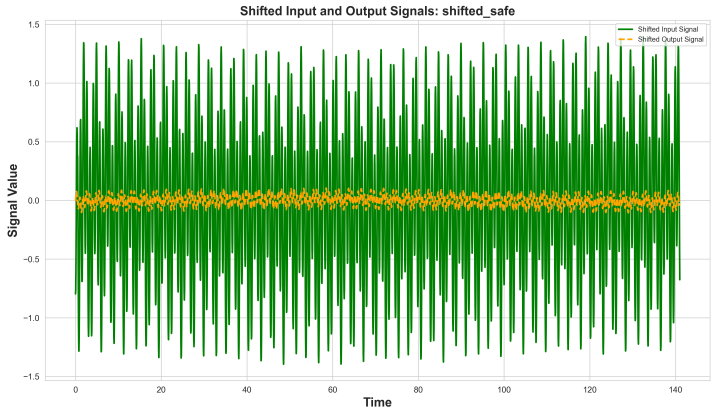


Figure 3.6: The shifted input signal and its corresponding output.

hinge on the scale factor's impact on the signal's utility and interpretability.

3.5.3 GAN Evaluation

To assess the GAN's effectiveness in generating realistic safe and unsafe signals, we evaluated the peak-to-peak value distributions for both categories. The GAN model was trained on a dataset containing 1500 safe and 1500 unsafe signals, optimizing parameters A_2, A_3, A_4, ϕ_1, ϕ_2, ϕ_3, and ϕ_4, with a fixed T value. The trained GAN achieved **92% accuracy** for safe signals and **93% accuracy** for unsafe signals in classification, as shown in the distribution histograms below.

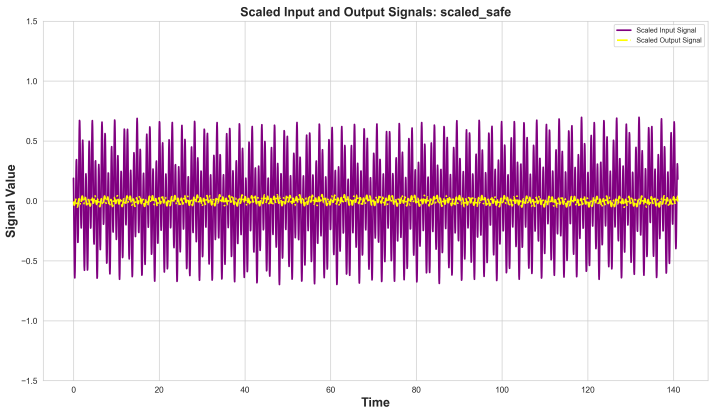


Figure 3.7: The scaled input signal and its corresponding output.

Safe Signal Results

In the histogram for safe signals (Figure 3.8), the distribution of peak-to-peak values centers primarily around the range of 2.6 to 3.0 V, with the highest frequency at approximately 2.8 V. This distribution indicates that the GAN effectively learned the characteristic peak-to-peak range for safe signals. The classification model achieved an accuracy of **92%** in identifying safe signals, demonstrating the GAN's success in replicating the core characteristics of safe signals in its synthetic data. The high classification accuracy for safe signals underscores the GAN's effectiveness as a data augmentation tool, helping to improve the robustness of the classification model even when real-world data is limited.

Unsafe Signal Results

For unsafe signals (Figure 3.9), the distribution of peak-to-peak values spans a broader range, from approximately 3.2 to 4.2 V, with prominent frequencies around 3.4 to 3.8 V. This broader distribution aligns with the more variable nature of unsafe signals, which generally exhibit higher peak-to-peak values. The classifier achieved an accuracy of **93%** in identifying unsafe signals, further validating the GAN's capability to generate realistic variations within the unsafe category. The high accuracy in classifying unsafe signals demonstrates that the GAN effectively captured the key characteristics of this category,

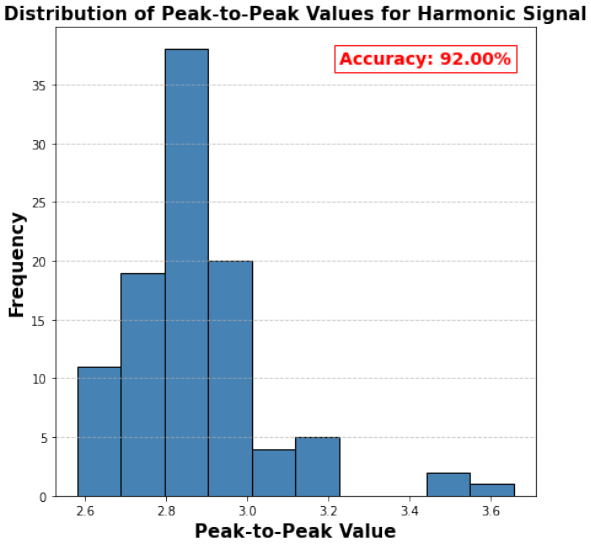


Figure 3.8: Distribution of peak-to-peak values for GAN-generated safe signals (accuracy: 92%)

enhancing the model's ability to generalize and accurately identify signal states under diverse conditions.

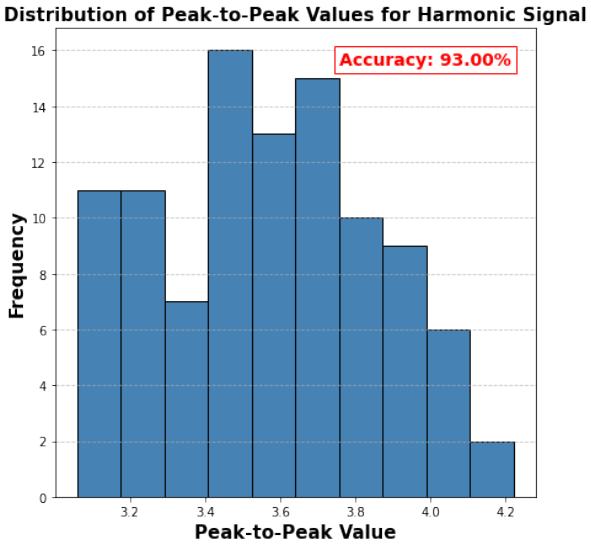


Figure 3.9: Distribution of peak-to-peak values for GAN-generated unsafe signals (Accuracy: 93%)

3.6 Conclusion

In this study, we introduced an advanced methodology for Electromagnetic Compatibility (EMC) analysis that combines image-based classification with waveform modulation analysis to rigorously assess signal safety in digital interconnect systems. This approach leverages S-parameter data and consistent RF power to yield a highly accurate evaluation of steady-state responses, surpassing traditional Continuous Wave (CW) methods in precision and relevance to real-world EMI conditions.

Our developed image-based classification system effectively discriminates between 'safe' and 'unsafe' signals, addressing a key limitation of traditional EMC approaches: limited empirical data. By constructing a comprehensive dataset, this system offers a more reliable assessment of signal safety, which is crucial for the design of robust digital interconnects capable of withstanding electromagnetic interference (EMI). Moreover, our analysis of waveform modulation reveals its significant role in optimizing EMI coupling efficiency, providing new insights into designing digital interconnects that are both resilient and high-performing under various RF interference scenarios.

The practical implications of our research extend to enhancing the reliability and security of electronic devices, particularly in high-interference environments where signal integrity is critical. By providing a robust framework for analyzing EMI effects, our approach supports the development of electronic systems that can maintain stable performance and reliability, even in challenging RF conditions.

A key innovation in our study is the integration of Generative Adversarial Networks (GANs) to overcome data scarcity, enabling synthetic generation of realistic 'safe' and 'unsafe' signal images. This augmentation significantly enriched our dataset, enhancing both the accuracy and resilience of our classification model. Through this comprehensive GAN-based augmentation, our model achieves reliable generalization across diverse conditions, demonstrating the potential of synthetic data generation to

advance EMC analysis. The inclusion of GANs in our methodology paves the way for more robust and adaptable signal safety frameworks, marking an important step toward creating resilient electronic systems in RF-intensive environments.

Chapter 4

Advancements in AI-Driven Customer Service

The chatbot [55] is a software tool powered by artificial intelligence, developed to maintain conversation and find relevant responses to address users' inquiries. Chatbots can be used across a variety of domains, such as managing personal bank accounts, checking the weather, and shopping. As the popularity of digital assistants grows, it becomes increasingly common for users to utilize these tools to gain information about products [52]. Consequently, in this paper, we have created an Amazon QA chatbot capable of answering users' queries by retrieving the most pertinent response from a pool of available answers.

Delving more into detail, the input to our system is a natural language question about a product, and the output is the top-ranked answer selected from a large dataset of potential answers. To enhance manageability, our research specifically targets the electronics category, which encompasses more than 300,000 question-and-answer pairs, in addition to other metadata. For this purpose, we have developed multiple algorithms and models to optimize and improve the accuracy of the chatbot. Furthermore, we plan to develop a web demo that will enhance the direct front-end interface of our agent [128].

4.1 Background

4.1.1 Motivation

Digital assistance is witnessing significant expansion. Its usage ranges from managing smart homes to teaching social skills to children. Nonetheless, there is a noticeable gap in providing solutions for answering product-related inquiries due to various technical and theoretical hindrances.

Challenges in natural language understanding (NLU) involve developing insights into a specific domain and its unique characteristics, as well as handling noisy inputs from users' speech and accents [101]. Additionally, effective information retrieval (IR) requires writing extensive rule sets or translating the user's natural language and intent into a format that can be comprehended computationally [101]. In both approaches, it is difficult to deliver accurate answers. On the psychological front, users may not always be clear or may struggle to articulate their inquiries. These reasons make the question-answering system both challenging and interesting at the same time [134].

4.1.2 Related Work

Considering the significant impact of conversational agents capable of providing relevant responses to questions, the development of these chatbots has been a captivating and dynamic area of research for many years [35]. By reviewing some previous papers related to this field, specifically focusing on the use of information retrieval for answering questions, we aim to highlight the importance of this area.

The study *An Information Retrieval Approach to Short Text Conversation* [179] addresses the fundamental challenges of conversational systems, focusing on ensuring that the computer provides reasonable responses to human input through information retrieval. For this purpose, they formalize the input as a search problem and then apply an information retrieval strategy, specifically a RankingSVM, to provide answers. The retrieval-based model functions more intelligently when applied to social media data [164]. Moreover, *Chatbot engines using unstructured documents* [167] introduced the DocChat method within the chatbot engine to explore unstructured information to answer human questions, demonstrating the efficacy of the retrieval-based method in finding the most relevant question and answer pairs directly [167]. On the other hand, generation-based methods focus on creating new answers. Previous approaches, whether search or generation-focused, depend heavily on extensive dialogue sentence pairs for training

data. In specific fields, obtaining a large dialogue corpus is more difficult, while ordinary text is more readily available. This paper proposes an efficient method for generating informative queries [124].

Transitioning towards statistical methodologies, the study "*Data-Driven Response Generation in Social Media*" utilizes Statistical Machine Translation (SMT) to generate responses [16]. This approach resolves the challenges associated with the broad spectrum of possible answers and the significant portion of unaligned words and phrases. The assessment, conducted using BLEU scores, demonstrates that SMT outperforms conventional Information Retrieval (IR) methods. Remarkably, SMT results are preferred over human responses in 15% of the cases [92] [178].

In recent research, more neural-based methods are being studied. The "Neural Responding Machine for Short-Text Conversation" introduces a neural network-based response generator for short-text interactions, known as the Neural Responding Machine (NRM) [93]. This method employs an encoder-decoder framework, formalizing the generation of responses as a decoding task. Both the encoding and decoding processes utilize Recurrent Neural Networks (RNNs), specifically Gated Recurrent Units (GRUs). The NRM models are capable of generating grammatically and contextually correct responses in more than 75% of the inputs [102]. Additionally, *A Neural Conversational Model* introduces a seq2seq-based language model that can be used to train a conversational engine [159]. This approach involves a recurrent neural network that processes the input sequence one token at a time and similarly predicts the output sequence token by token. During training, the actual output sequence is provided, enabling learning via backpropagation. The model's goal is to optimize the cross-entropy of the correct sequence given its context [128]. In inference, since the actual output sequence cannot be observed by users, the model uses the predicted output token to forecast the subsequent output, thus enhancing its predictive accuracy over time [121] [134].

4.2 Problem Description & Datasets

The objective of this paper is to respond to consumer queries in text format, typically consisting of a sentence or two, utilizing an Amazon knowledge-based chatbot. To accomplish this, we initially embed both the query and all potential questions into a common vector space. Subsequently, we identify the question that most closely aligns with the query and present the answer associated with that question. In parallel, we explore the Direct-to-Answer approach. This method involves encoding the consumer's query and directly matching it with available answers to select the most appropriate one. Although this approach is more straightforward, it may require specialized machine reasoning.

4.2.1 Datasets

Our dataset originates from the Amazon Q&A collections created by the McAuley team at UCSD and is limited to the Electronics category. It comprises 314,263 instances. The format of the dataset is outlined below:


```
{
  "asin": "B000050B6Z",
  "questionType": "yes/no",
  "answerType": "Y",
  "answerTime": "Aug 8, 2014",
  "unixTime": 1407481200,
  "question": "Can you use this unit with GEL shaving cans?",
  "answer": "Yes. If the can fits in the machine it will despense hot
gel lather. I've been using my machine for both , gel and traditional
lather for over 10 years."
}
```


Figure 4.1: Example of a dataset format.

4.2.2 Evaluation Methods

Upon reviewing the data, we noted that many questions lack clarity because a single inquiry can be interpreted in multiple ways and still yield the same response. Therefore, to enhance the quality of the user experience, our chatbot needs to process a large variety of queries, even if they are not worded as in our dataset. It is unlikely that users will

phrase a question exactly as it appears in our dataset. To address this problem, we must manually annotate subsets of the data and revise questions.

To elucidate this issue further, let us consider an example. Imagine a user poses the question, "How is the warranty for this monitor?" If our dataset contains this exact question, then matching algorithms will easily identify the precise answer. However, if this query is worded differently in the chatbot's dataset, despite the question's concept being the same, finding the answer becomes less straightforward.

It is more likely that the input queries will not exactly match those in our dataset; hence, our evaluation must account for these differences. Our algorithms are designed to handle such deviations. However, without a collection of varied modified queries, assessing the performance of the algorithms is difficult. Although developing this project is time-consuming, we have devoted significant effort to creating such a dependable test set.

As part of this result, we have developed a set of 1200 manually edited queries that serve as paraphrases of the original inputs. For each example query, there is an accompanying set of 100 questions for evaluation, consisting of 99 unrelated samples and one original question corresponding to the query. Ideally, our algorithm should be able to match the original question with this set of 100 using the revised query to find the best answer. In the Direct-to-Answer method, approximately 45,000 answers are available for evaluation, having been selected from the entire dataset based on specific criteria. More specifically, we filter examples that exhibit any of the following characteristics:

label=() The type of question is Yes/No.

lbbel=() Answers contain fewer than 5 tokens.

lcbel=() Answers exceed 50 tokens in length.

Lastly, in the Raw Answers approach, the comprehensive set of over 300,000 answers forms the evaluation candidate set.

4.2.3 Metrics

C. Evaluation Metrics

We assess our models' effectiveness by calculating their overall accuracy. Specifically, each case provides the algorithm with one opportunity to accurately identify a match, a process repeated 1200 times. Therefore, the accuracy rate is determined by the percentage of instances in which the algorithm efficiently selects the correct match, serving as a direct measure of the algorithm's capability. Additionally, we perform manual evaluations of the responses for a qualitative analysis of their performance.

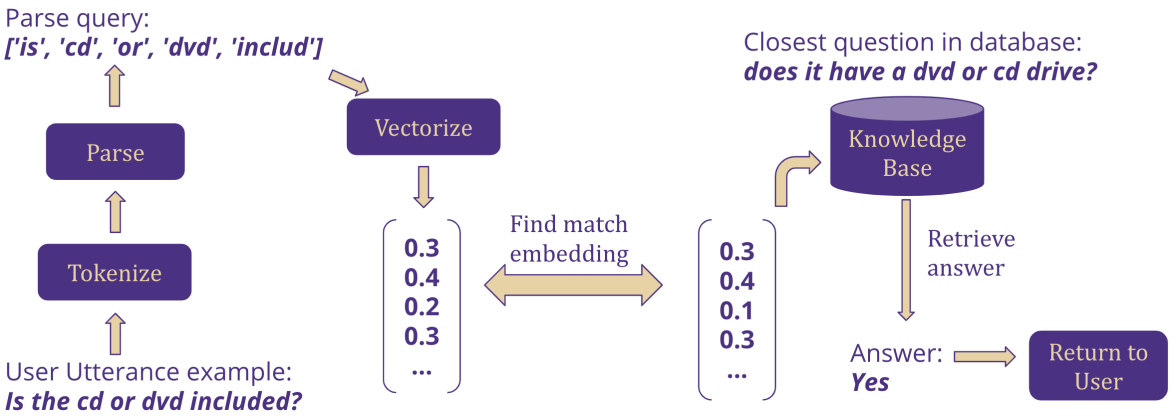


Figure 4.2: Solution architecture of our chatbot.

4.3 Experiments

4.3.1 Solution Architecture

Fig. 2 shows our design for the solution architecture of the chatbot. Once a user utterance comes in, it is tokenized and parsed. The *parse* step includes removing stopwords, stemming and switching from numerical values to special digit tokens. For instance, if the utterance is *Is the cd or dvd included?*, then the parsed query would be like *['is', 'cd', 'or', 'dvd', 'includ']*. The tokens will be further vectorized into an array of numbers.

The closest matching embedding will then be found by comparing the vector to each

vectorized question in the knowledge base. As a result, the user will receive the response that corresponds to this closest embedding. The closest response to the updated query is given back straight in the Direct-to-Answer scenario, without being matched to the question.

4.3.2 Data Preprocessing

We performed several preprocessing steps on the dataset before using it. Following the preparation of the data, we statistically analyzed the corpus. In user utterances, the average length of a question is 14.1 tokens, the average length of a response is 31.9 tokens, and 40.8% of queries have a number in them. We have a total of 14,460,956 tokens in our vocabulary—42,389 unique tokens at the beginning, before delexicalization and filtering, and 19,246 unique tokens at the end. The data preprocessing steps are carried out through three main phases:

- **Cleaning** consists of eliminating unnecessary textual elements, like capitalization and stopword elimination. To aid in tokenization, we also had to substitute punctuation on a case-by-case basis. This is significant because electronic products often include references to items that include versions such as `'Kindle v2.0'` which include punctuation, but should not be split.

- **Normalization** entails mapping concepts inside the scheme or lemmatizing and stemming words to reduce language. For these reasons, we choose to use the default NLTK stemming package. Also, we swap out any incredibly uncommon words with the `<unk>` token. For example, *fujifilm xm* in "Would the lens fit on a *fujifilm xm*?" would be considered as rare.

- **Delexicalization** consists of substituting numbers to their underlying forms represented by the number of digits. So for example, 640 would be replaced with `<3-digit>` and 20 would be replaced with `<2-digit>`. This step essentially

Table 4.1. Algorithm Performance

Retrieval Style	**Algorithms**			
	BM25	***GloVe***	***word2Vec***	***FastText***
Raw Answers	0.94%	Out of Memory	Out of Memory	Out of Memory
	6 hours 38 min	20 hours 14 min	32 hours 24 min	25 hours 30 min
Filtered Answers	8.00%	2.25%	2.25%	3.83%
	57 minutes	2 hours 52 min	5 hours 41 min	3 hours 35 min
Raw Questions	87.85%	72.25%	70.67%	83.67%
	1.5 minutes	1.8 minutes	2.4 minutes	1.9 minutes

replaced nearly 10,000 unique number tokens with just 9 special tokens, as a result of the vocabulary's exponential growth.

4.3.3 Algorithms

We applied four major algorithms including BM25, Word2Vec, GloVe and FastText, which are introduced as below. All four types are effectively sentence embeddings which can map any given query or answer into the same vector space for comparison.

- **BM25** is the state-of-art TF-IDF-like algorithm for information retrieval. The inverse-document-frequency part is derived from a binary independence model, and the term-frequency part is a saturation function that approximates a complicated Poisson distribution model that takes into account the document length [83].

  Given a query Q, which contains keywords $q_1, q_2, ..., q_n$, the BM25 score for certain document can be calculated as below:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k + 1)}{f(q_i, D) + k \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \tag{4.1}$$

  where $f(q_i, D)$ is the term frequency of q_i in document D. $|D|$ is the length of

document D, while *avgdl* is the average length of all documents. k and b are parameters and usually $k \in [1.2, 2.0]$ and $b = 0.75$. The $IDF(q_i)$ is the IDF part(inverse-document-frequency), which is usually given as below:

$$\text{IDF}(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{4.2}$$

where N is the total number of documents, and $n(q_i)$ is the number of documents that contains q_i.

- **Word2Vec** is a model produces word embedding from corpus. It has two major implementations that are complements of each other. The first is the Continous Bag Of Words (CBOW) in which given surrounding words, the model aims to predict the missing word [83]. For example, given `Today is a really --- day.` the model would be expected to produce `beautiful`, `nice` or `wonderful`. It is normalized by:

$$V_{w_{I,c}}^{(new)} = V_{w_{I,c}}^{(old)} - \eta \cdot \frac{1}{C} \cdot EH$$

The second method is called the Skip-gram method. In this case, given the center word, learn to predict surrounding words. For example, given `beautiful`, the model might be expected to produce the tokens [a, really, day, .] for a context window of length 2. It is normalized by:

$$V_{w_{I,c}}^{(new)} = V_{w_{I,c}}^{(old)} - \eta \cdot \frac{1}{C} \cdot EH$$

The overall prediction is done with a typical softmax:

$$p(w_O|w_I) = \frac{\exp\left((v'_{w_O})^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left((v'_w)^\top v_{w_I}\right)}$$

Then, the weights of the trained network become the embeddings to use in the future.

- **GLoVe** stands for a *G*lobal and *Lo*cal *Ve*ctors. Its creation starts by making a large term-frequency matrix, similar to TF-IDF which is the global view. The local view, which is each token's window context, is taken into account when building this matrix. After that, rank reduction from millions of dimensions to a more manageable number, like 100 or 300 dim, is accomplished via global matrix factorization. We employed the 300 dim pre-trained embeddings for our purposes [53].

  In more detail, global matrix factorization aka latent semantic analysis (LSA) can be reduced via singular value decomposition (SVD).

$$M = U\Sigma V^*$$
$$UU^* = I_m$$
$$VV^* = I_n$$

  Since GLoVe looks at local context like Word2Vec, and global context like BM25, it is theoretically a more powerful embedding model.

- **FastText** is an algorithm that dissects each token into its constituent subwords in addition to examining the token's embedding [32]. A token like `school`, for instance, would have the trigram breakdown "sch, cho, hoo, ool," each of which would have its own embedding and score. Specifically, the scoring mechanism works as:

$$score(w, c) = \sum_{g \in G_w} \mathbf{z}_g^\top \mathbf{v}_c$$

  Then, The final token is calculated by averaging the sum of the n-gram embeddings and the whole token embedding, `school`. The capacity to handle unfamiliar words

is the main advantage of this method because it is highly probable that new words will contain known subword embeddings. This is particularly pertinent to our project because, as was already noted, there are a lot of uncommon product names that are only seen once or never in training.

4.4 Results

In this section, we present the evaluation and comparison of the four major algorithms—BM25, Word2Vec, GloVe, and FastText—for their effectiveness in generating sentence embeddings. The experiment performed on the datasets show that BM25 achieves the optimal performance in terms of both accuracy and time cost (Table I) compare to conventional models. On average the best performance have FastText and Word2Vec models (0.60). Comparing queries to raw questions, we can see that BM25 can achieve 87.85% accuracy, which is very high compared to GloVe and word2vec, and only slightly lower than FastText's 83.67% accuracy. The time cost is also the lowest for BM25. We believe that this is due to the fact that the neural embedding matrix multiplications take up a lot of time.

For direct-to-answer approach using filtered answers, BM25 has the highest accuracy of 8.00% with the shortest time of 57min. However, this very low accuracy indicates that this approach is not suitable for this project. Therefore, we recommend the nearest query method. Finally, if you use raw answers, all approaches will have memory issues and you will not be able to complete the task. This is because you need to create a matrix of 1200x314x300 dimensions. The 314 comes from the size of the dataset, the 1200 is the number of examples and the 300 is the word dimension size.

4.5 Limitations and Conclusion

The major limitation is the lack of computing power. We come across out-of-memory issues when trying to compare a 310,000 x 300 dim matrix. Also in terms of time cost,

it takes hours of computation required to loop through 10k+ examples. Thus we tried to mitigate the issue by precomputing tokenization and then caching. Also, the results from the Direct-To-Answer approach show that the questions have very limited word overlap with answers. After all, intuitively, the words in the answer might not match the words in the question. For example, for the question-answer pair `\What is the weather like?"` and `\Cloudy with a chance of rain"`, it's much easier to match with the revised queries like `\What will the weather be today?"` but not answers due to the unigram overlap from question to query.

In terms of algorithmic performance, comparing embeddings is still very fast compared to neural networks or Gradient-boosting algorithms. Also since we wrap the code into an independent class for each algorithm, we can easily transfer it into new embedding types for comparison. Also, our work is easy to interpret because we are just finding similar sentences. However, our algorithms do not have the processing power of neural networks to learn from data and can be quite slow since all candidates must be processed during every query. In conclusion, we have learned a great deal about the trade-offs in word embeddings and would recommend taking a simple approach before diving into more complex neural methods.

Chapter 5

IoT-based Contact Tracing Framework in Smart Cities

Recently, smartphone-based contact tracing has emerged as a practical way to trace someone's social exposure with many public-safety-related applications including infectious disease tracking [36]. Such systems can monitor peer-to-peer interactions between citizens and retroactively provide alerts to users that were in contact with someone that has been diagnosed or tested positive for an infectious disease. A case in point for the importance of such systems is the recent COVID-19 pandemic where many governments have been looking for effective ways to relax restrictions, resume industry operations and bring back daily routines without risking dangerous outbreaks [61]. Therefore, effective contact tracing can be added to the relatively short list of outbreak-preventive measures that also includes regular hand washing, face covering, and temperature checks.

Traditionally, contract tracing is a manual process that requires the collaboration of multiple authorized entities and personnel resulting in a time-consuming operation [51]. Recently, multiple parties have been developing contract tracing applications that rely on peer-to-peer (P2P) architectures that track interactions between individuals through their smartphones or other smart wearables (e.g., smartwatches [115]). The majority of those applications detect either the location or the proximity of two users, and rely mostly on well-known positioning tools that include GPS [4] and more prominently the Bluetooth Low Energy protocol [45, 100, 115, 116, 155]. Specifically, the list of contact tracing apps that use BLE as their main technology include the BlueTrace protocol -the basis behind Singapore's TraceTogether app [29]-, Apple/Google's decentralized CT protocol [100], the Pan-European Privacy-Preserving Proximity Tracing (PEEP-PT) [155], Privacy-Preserving Automated Contact Tracing (PACT) [129] among others. The logic behind such applications is simple: User devices' transmit and detect BLE advertise-

ment packets, and utilize them to exchange the users' IDs and calculate their proximity. These encounters are logged either on the device or on a remote central authority to be used in case one of the two parties tests positive (in the case of infectious diseases) or is picked out for any reason. Then, using history logs other contracts of that person will be identified and alerted either through the device itself or by other means.

While the aforementioned approaches are promising in terms of performance, the overall designs tend to ignore the impending fact that a massive Internet of Things' infrastructure is actually embedded within urban areas and smart cities. Such Internet of Things (IoT) infrastructure supports city-wide communications [103, 144, 147] and consists of computationally powerful devices including Electric Vehicles, smart locks, cameras, drones, various sensors, smart furniture, and WiFi hotspots [18,77,81,145]. Utilizing such networks of interconnected and computationally powerful devices can lead to the development of alternative contact tracing applications or (most probably) reinforce the operation of existing P2P contact tracing and significantly improve their performance and accuracy.

In this work [14], we specifically address this gap by presenting an IoT-assisted correlative contact tracing framework that relies on passive interactions between mobile users and static IoT devices that are already part of a wider smart city ecosystem. Specifically, we:

- Design a distributed framework that enables IoT devices to harvest passive advertising packets and act as contact tracing anchors to detect and log interactions between two smart space occupants.

- Deploy a centralized solution able to combine information from multiple IoT devices concurrently towards identifying more accurate contact tracing information including duration of interaction, and distance.

- Utilize machine learning methods to accurately classify the infection risk between

two users given specific attributes of their interaction.

- Evaluate our design on a real-world case study where 46 participants interact with an IoT infrastructure of 32 devices as part of their daily routine using the Bluetooth Low Energy protocol.

5.1 IoT-enabled Contact Tracing Framework

5.1.1 Architecture

The proposed system utilizes existing IoT devices with multiple wireless interfaces as permanent wireless packet scanners. We will denote as $\mathcal{I}$ the set of $|N|$ IoT devices randomly placed in our smart setting (i.e., following the random placement of IoT devices such as cameras, smart locks, sensors, etc.). We will also denote as $\mathcal{U}$ the set of $|M|$ mobile or static users that are frequent occupants on the smart space and carry their respective personal devices. Our design accounts for any kind of personal smart devices, such as smartphone, smartwatch, or other wearables, as well as any device with connectivity capabilities handed to the users by the facility/city administrators (e.g., smart id cards). Such devices can emit non-intrusive advertising packets by any of their communication interfaces, with the most common being Bluetooth Low Energy (BLE) advertisement packets and WiFi probe request frames. Our design utilizes such passive user-IoT device interactions to calculate the proximity of a user to an IoT device and document their interaction with other users through the common edge device with which they are both associated (common anchor). Each personal device can be anonymously linked to its owner from any wireless interface that might utilize for the contact tracing capability.

Each IoT device reports all the recorded interactions between users to a centralized processing unit (e.g., a local server or cloud environment [143]) that combines information from all sightings and produces a final contact trace between every pair of users. For each pair, our system can calculate the duration of the contact, as well as produce

an estimation of the distance between the two users during the contact. Based on the information above, our system outputs an infection risk indicator that will be the outcome of infection probability modeling (performed by a specialist) and machine learning models as described in this work. Figure 5.1 shows the overall architecture of the proposed framework, while Table 5.1 summarizes all the different tuning parameters of the proposed system as found in this section.

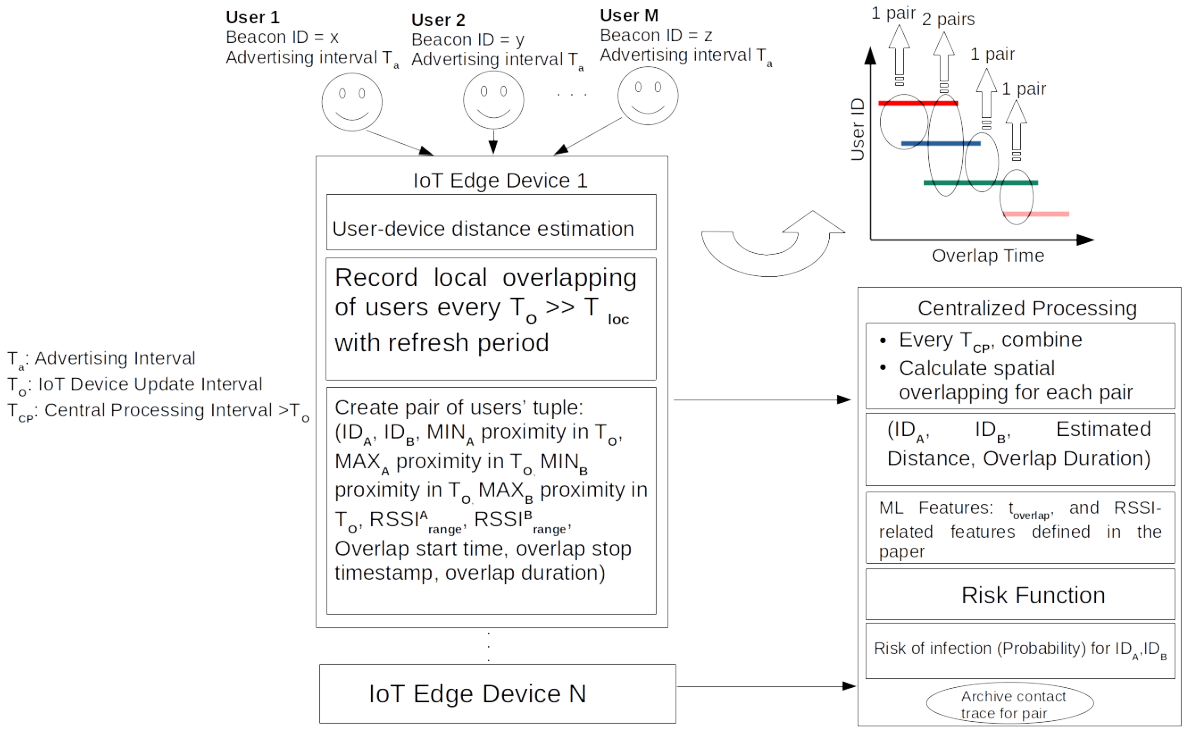


Figure 5.1: Framework Architecture

Notation	Definition
$\mathcal{I} = \{i_0, ..., i_n, ..., i_{\|N\|}\}$	Set of IoT devices
$\mathcal{U} = \{u_0, ..., u_m, ..., u_{\|M\|}\}$	Set of users
T_a	User Device Advertising interval
T_o	Update interval of IoT device
$T_{CP} > T_o$	Centralized processing interval
d_{ij}	Estimated distance between user i and user j
t_{ij}	Overlapping time duration for user i and user j

Table 5.1. Key Notations

5.1.2 User-IoT device Distance Estimation using RSSI

Given the constant stream of advertising packets from the users to the IoT devices, our framework's first step is to estimate the user-IoT device distance at any time. This process takes place locally within each IoT device. In what follows, we explain the required data processing steps to estimate the user-device distance.

First, we assume that every personal user device transmits advertising packets every T_a seconds (advertising interval). These packets are picked up by multiple IoT devices, and their Received Signal Strength Indicators (RSSI) are used to calculate the user-IoT device distance locally within each device. To do so, the following process takes place periodically every T_o seconds across all available IoT devices. First, we perform Exponential Smoothing [7] to smooth the RSSI values and remove high bandwidth noise as follows:

$$\hat{rssi}_{t+1|t} = \alpha rssi_t + (1-\alpha)\hat{rssi}_{t|t-1} \tag{5.1}$$

where $\hat{rssi}_{t+1|t}$ is the estimated RSSI at time $t+1$ is calculated using the weighted average of the most recent $rssi_t$ and the previous estimate $\hat{rssi}_{t|t-1}$. Parameter α controls the weight of the past and recent observations over time (e.g. it associates smaller weights to the older data and greater weights to the recent data).

Next, we apply Kalman filter to provide a better estimation of RSSI values. It consists of two parts: 1) Measurement update (Bayes rule product) and 2) Prediction (total probability convolution). In the update step, two Gaussian distributions used are called prior and measurement. The prior distribution has a mean of m_1 and a variance of ${v_1}^2$, and the measurement distribution has a mean of m_2 and a variance of ${v_2}^2$. We multiply two prior and measurement distributions of RSSI values and obtain the new

RSSI measurement using m' and $v^{2'}$. The update equations are as follows:

$$\begin{aligned} m' &= \frac{{v_2}^2 m_1 + {v_1}^2 m_2}{{v_2}^2 + {v_1}^2} \\ v^{2'} &= \frac{1}{\frac{1}{{v_2}^2} + \frac{1}{{v_1}^2}} \end{aligned} \tag{5.2}$$

In the prediction step, we add up the old mean m_1 and the motion x, and for the variance, we add up the old variance ${v_1}^2$ and ${v_2}^2$. The prediction equations are as follows: x is denoted as motion or estimated RSSI (e.g., if the user moves over to another place with a different received RSSI).

$$\begin{aligned} m' &\leftarrow m_1 + x \\ v^{2'} &\leftarrow {v_1}^2 + {v_2}^2 \end{aligned} \tag{5.3}$$

Figure 5.2 is a visualization of above equations and illustrates the RSSI estimation using two Gaussian distributions. Thus, in this step, we forecast the users' RSSI to the nearby devices.

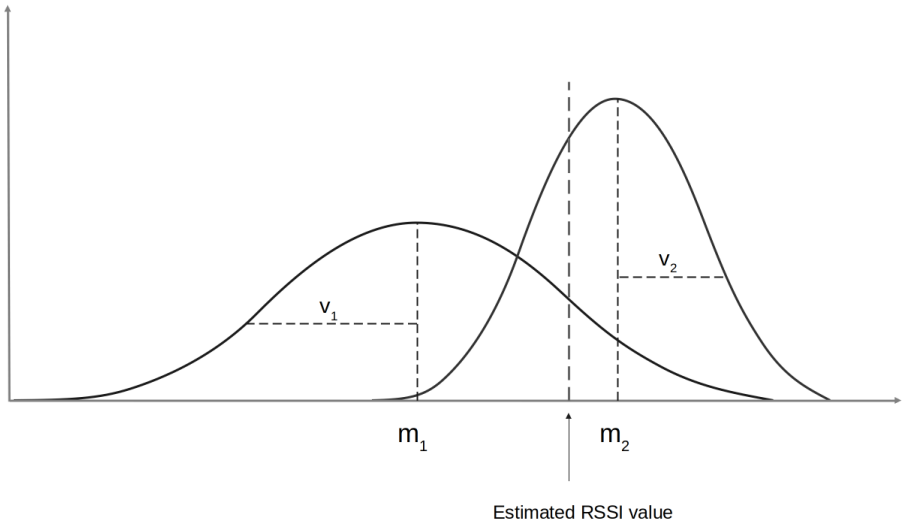


Figure 5.2: RSSI Estimation Using Kalman Filter [119]

Finally, the Kalman filter outputs the RSSI estimate values for the specific user.

Each RSSI estimation is converted to distance. To do so, we use the following path loss model [84, 140] to calculate the corresponding distance value d:

$$p_d = p_{d0} - 10\ \beta\ log(\frac{d}{d_0}) \tag{5.4}$$

Where p_d is the observed RSSI value (dBm) d meters away from the IoT device and p_{d0} is the RSSI value (dBm) observed at a reference distance d_0.

5.1.3 IoT Device Scanning

Every IoT device gathers distance readings from each user in its proximity within the T_o interval. Next, each IoT device locally calculates the overlap time between each pair of users and logs the minimum and maximum distance observed for each of them during T_o. This information is sent at the end of every T_o period to the centralized unit (see Figure 5.1). Specifically, each IoT device reports:

- ID_A: ID of the pair's first user
- ID_B: ID of the pair's second user
- $t_{overlap}$: Duration of the pair's encounter
- min_A: Minimum distance between user A and IoT device within T_o
- max_A: Maximum distance between user A and IoT device within T_o
- min_B: Minimum distance between user B and IoT device within T_o
- max_B: Maximum distance between user B and IoT device within T_o
- $RSSI^A_{range}$: Range of RSSI readings for user A
- $RSSI^B_{range}$: Range of RSSI readings for user B

5.1.4 Centralized Processing

Combining observations from multiple IoT devices for the same pair

Each user pair's information (overlap time, and distances to the device) may have been collected by multiple nearby IoT devices. Thus, each device will generate alternative values for the same user pair describing the same time interval (T_o). This information is sent to a centralized unit where our framework combines the given distributed information to produce a holistic view for each pair. This process in the central unit happens also during specific time intervals T_{CP} where $T_{CP} > T_o$.

Assuming that one or multiple devices have reported that a specific pair of users A and B are in close proximity, the centralized unit logs the IDs of the interacting users, and their combined duration of interaction:

$$t^{AB}_{overlap} = max({}^{1}t^{AB}_{overlap}, {}^{2}t^{AB}_{overlap}, ..., {}^{n}t^{AB}_{overlap}),\ with\ n <= |N| \tag{5.5}$$

In addition, for each user m, we calculate the average RSSI range from all IoT devices:

$$\overline{RSSI^{m}_{range}} = \frac{\sum_{i=1}^{n} {}^{i}RSSI^{m}_{range}}{n},\ with\ n <= |N| \tag{5.6}$$

Regarding the estimated distance between users, we first estimate the distance of each user m from an IoT device n as $r^m_n = \frac{max_m + min_m}{2}$. Next, we identify three possible scenarios depending on the number of IoT devices that intercepted the pair during the T_{CP} interval:

1) A single IoT device n has identified the AB user pair: In this case we estimate their distance d_{AB} as:

$$d_{AB} = |r^A_n - r^B_n| \tag{5.7}$$

2) Two IoT devices n_1 and n_2 have identified the AB user pair: In this case we rely on

the IoT device that was closer to the interaction (e.g., n_1 in case $min(min_A^{n_1}, min_A^{n_2}, min_B^{n_1}, min_A^{n_2}) = min_A^{n_1}$ *or* $min_B^{n_1}$) to estimate their distance d_{AB} as in Eq. 5.7.

3) Three or more IoT devices have identified the AB user pair: In this case, we rely on the known locations of each IoT device (x_n, y_n) and calculate the exact location of each user m (x_m, y_m) using trilateration [6, 132]. If more than three IoT devices have intercepted the pair, we use the measurements from the three devices that present the lower RSSI readings. Figure 5.3 shows the trilateration process. A user's smart device has been identified by three IoT devices. The distance between the smart device and each IoT device represents the radius of a circle showing all possible locations of the mobile smart device. The user m (x_m, y_m) coordinates are obtained using equations for the three circles in the two-dimensional Cartesian coordinate system and by solving the following equations:

$$\begin{aligned}(-2x_1 + 2x_2)x + (-2y_1 + 2y_2)y &= {r_1}^2 - {r_2}^2 - {x_1}^2 + {x_2}^2 - {y_1}^2 + {y_2}^2 \\ (-2x_2 + 2x_3)x + (-2y_2 + 2y_3)y &= {r_2}^2 - {r_3}^2 - {x_2}^2 + {x_3}^2 - {y_2}^2 + {y_3}^2\end{aligned} \tag{5.8}$$

Next, given the estimated locations of the two users A (x_A, y_A) and B (x_B, y_B) their distance is given by:

$$d_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{5.9}$$

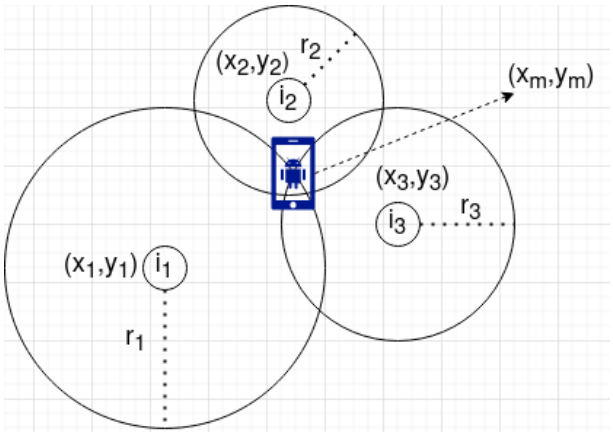


Figure 5.3: Trilateration

Finally, the framework outputs the combined interaction information for each pair of users A, and B within the interval T_{CP}:

- ID_A: ID of the pair's first user
- ID_B: ID of the pair's second user
- $t^{AB}_{overlap}$: Overlapping time of pair
- d_{AB}: The estimated distance between the two users
- $\overline{RSSI^{A}_{range}}$: Average RSSI range for user A
- $\overline{RSSI^{B}_{range}}$: Average RSSI range for user B

5.2 Infection Risk Assessment using Machine Learning

5.2.1 Risk Classification

Next, our framework estimates the infection risk following the interaction of two users. To do so, we consider the RSSI interactions between the user pair and the IoT devices, and the users' estimated overlapping time to identify this interaction between users as low, medium, or high risk. We adopt a similar risk classification logic, as presented in [116]. We categorize the risk prediction function as a hypothesis test. We suppose x as a feature vector and R as a risk prediction function $R : (x) \rightarrow \{1, 2, 3\}$, where 1 shows low, 2 medium, and 3 high risk. Then we have the following three hypotheses, as shown below:

$$\begin{aligned} H_0 &= R(x) = 1 \\ H_1 &= R(x) = 2 \\ H_2 &= R(x) = 3 \end{aligned} \tag{5.10}$$

where H_0 represents a low risk, H_1 the average risk, and H_2 the high risk. Under this scenario, a false positive is an error in the classification in which a hypothesis incorrectly shows the interaction as low risk. However, this classification is high-risk, while in a false negative error, the hypothesis incorrectly shows the high-risk user to be low-risk.

The exact function that accurately translates the interaction between two parties to infection risk is a variable that relates to the exact nature of the infection, and specific analysis made by the field specialists [15] [28] [173]. Therefore, the proposed framework can be retrofit to any past or future pandemic cases. We will also utilize a supervised machine-learning approach that requires a set of labeled interactions before being able to classify the interaction risk successfully. This labeling can be initially made by the specific infection risk function. However, after this system is available online and more and more interaction cases are identified, they can be labeled accurately by their outcome. For instance, if a user has been identified as infectious, his contacts will be identified. Then, suspicious interactions will be labeled as high risk.

Since, neither a specific risk function or after-the-fact data are currently available to us, we define an alternative labeling procedure, to showcase the applicability of this method. Our manual label classification considers the interaction distance and time overlap duration between any two individuals. We consider specific conditions on d_{ij} and t_{ij} for calculating the labels as shown below and in Figure 5.4:

- High-risk = 2, { $d_{ij} < 2$ m and $t_{ij} \geq 60$ s } or { 2 m $\leq d_{ij} \leq 6$ m and $t_{ij} \geq 3600$ s } or { 6 m $\leq d_{ij} \leq 10$ m and $t_{ij} \geq 3600$ s }
- Medium-risk = 1, { $d_{ij} \leq 2$ m and $t_{ij} < 60$ s } or { 2 m $\leq d_{ij} \leq$6 m and $t_{ij} < 3600$ s }
- Low-risk = 0, { 6 m $\leq d_{ij} \leq 10$ m and $t_{ij} < 3600$ s}

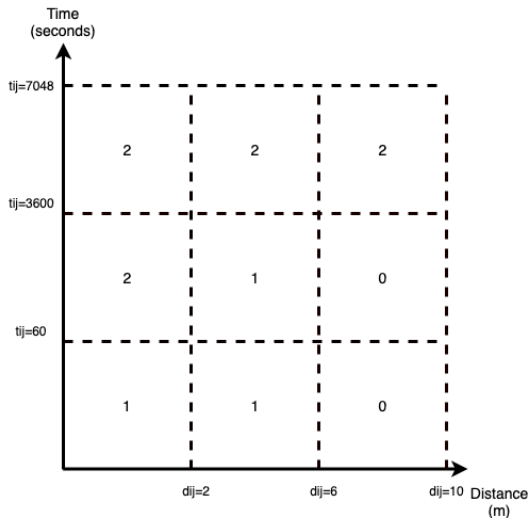


Figure 5.4: Label-risk function

5.2.2 Classification Algorithms

Our machine-learning approach utilizes five input features to classify a user pair interaction at a specific risk level. These features are the minimum and maximum RSSI values observed by each user, their estimated position user, the average RSSI observed for each user, and their respective average RSSI ranges. These features provide information on the environmental noise conditions that accompanied each interaction. In addition, we compared different classification methods. We applied multi-class classification (One-vs-all) in all models where we have three classes of labels equaling high risk, medium risk, and low risk. In the following, we will briefly describe these classification models:

1. Decision Tree(DR): Decision Tree is a supervised and nonlinear machine-learning model. The DR uses a training set and splits the feature input into smaller subsets and at the same time creates a related tree. The Gini impurity measurement is used in DR to decide the optimal split for the nodes. The formula for calculating the Gini impurity of a feature is as follows:

$$G = \sum_{i} P(i) * (1 - P(i)) \tag{5.11}$$

where P(i) is the likelihood of a certain classification i, for the training data set [76].

2. Extreme Gradient Boosting (XGBoost): XGboost is an excellent technique to improve the performance of the model by reducing overfitting [37]. This model is high-speed and scalable. The objective function of XGBoost is illustrated as below:

$$\begin{aligned} N(\theta) = &\quad L(\theta) + \Omega(\theta) \\ \Omega(\theta) = &\ \gamma T + \frac{1}{2}\lambda \|w\|^2 \end{aligned} \tag{5.12}$$

where γ controls the learning rate and set to a value greater than 0 and less than 1, w is the weight of the leaves, and $\Omega(\theta)$ is a parameter which controls the regularization term. T is the number of leaves in a tree and γT prevents overfitting. $L(\theta)$ calculates the loss function and fits the data for our model. The equation for calculating $L(\theta)$ is illustrated as follows:

$$L(\theta) \approx \sum_{i=1}^{n} l(y_i, \hat{\mathrm{y}}_i^{(t-1)}) + \mathrm{g}_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) + \Omega(\theta). \tag{5.13}$$

where l is loss function, f_t is the t^{th} tree output. g_i and h_i are the first and second derivatives of the loss function as given below:

$$\begin{aligned} g_i &= \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}(t-1)} l(y_i, \hat{y}^{(t-1)}) \end{aligned} \tag{5.14}$$

The Equation 5.14 shows y_i and $\hat{y}_i$ as target label and predicted label accordingly. From Equation 5.12 we find the best splitting point for each tree and minimize the objective function and the goal is to create the tree and minimize the target label.

3. Bootstrap aggregating (Bagging): Bagging builds the learning algorithm on a different subset from training data. The algorithm collects the groups of data and

trains each bag separately. The new predictions are built by averaging the predictions together from the base learners. This is shown in Equation 5.15 where $\hat{g}_{bag}$ is the bagged prediction, x is the record for which we need to produce a prediction, and $\hat{g}_1(x) + \hat{g}_2(x) + ... + \hat{g}_h(x)$ are the predictions from the particular base learners.

$$\hat{g}_{bag} = \hat{g}_1(x) + \hat{g}_2(x) + ... + \hat{g}_h(x) \tag{5.15}$$

The aggregation method reduces the variance of the individual base learner.

4. K-nearest neighbors (KNN): KNN is a non-parametric supervised learning method based on distance rule-based techniques. KNN tries to classify a data point to a given category with the training set [163]. KNN uses k samples of group data and finds the distances between a query and all the data's training information. By choosing a specific number for K, the algorithm uses a majority vote for finding the most frequent label. Therefore, K is the tuning parameter that affects the performance of KNN.

5.3 Experimental Evaluation

The proposed framework is evaluated using the Bluetooth Low Energy protocol as the main means of passive packet advertising, and our real-world trial-generated dataset of user-IoT device interactions, the BLEBeacon dataset [141, 142].

5.3.1 Case Study

Our case study environment is comprised of multiple BLE advertising devices (held by 46 smart space occupants) and a sensing infrastructure of IoT-like devices [141], which in this case are 32 Raspberry Pis (RPi). Note that the locations of the IoT devices were dictated by the need for outlets, and therefore are distributed in a non-uniform fashion in the physical space. This setting emulates a real-world installation of multi-

purpose IoT devices very closely. The trial participants (college students and faculty following their usual routines in the indoor environment of a university building) carry BLE beacons (Gimbal Series 10) that produce advertising packets with a transmission rate of 1 Hz ($T_a = 1\ second$). The generated packets are collected by the RPis, thus emulating interactions between mobile users following their routine (for one month) and IoT devices. Figure 5.5 illustrates the RPis' exact locations and Table 8.1 shows the exact parameter values that were used to evaluate our framework in this real-world scenario.

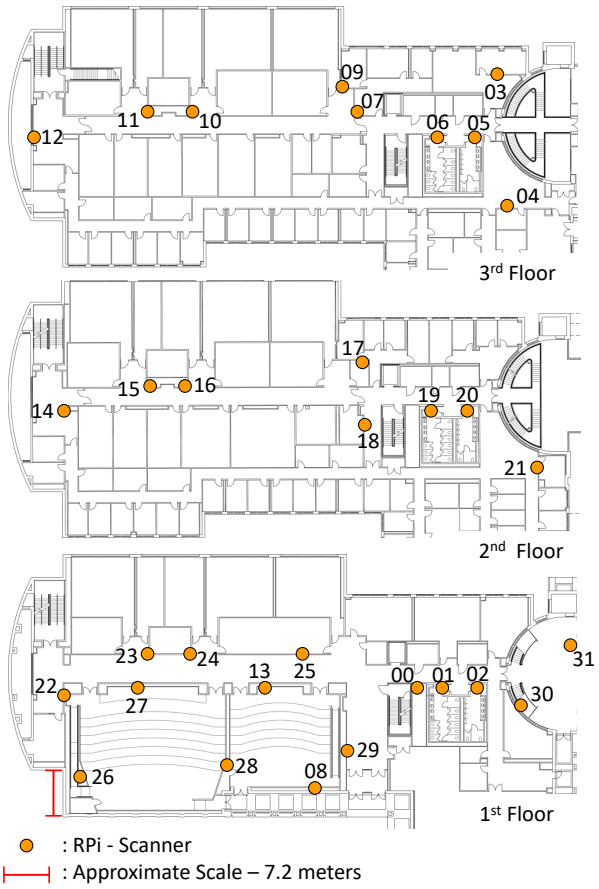


Figure 5.5: Location of IoT Devices - Smart Space Topology [141]

5.3.2 User-Device Distance Estimation Accuracy

First, we evaluate the framework's capability to estimate the distance between the user and the IoT device using BLE advertising packets. To do so, we have conducted the following experiment. A user equipped with a BLE beacon was kept static at different

Parameter	Value
Number of IoT devices (N)	32
Number of Beacons/Participants (M)	46
Advertising interval (T_a)	1 *second*
Update interval of IoT device (T_o)	$10, 15, 20$ *seconds*
Centralized processing interval (T_{CP})	$40\ s, 1$ *minute*, 2 *minutes*
Smoothing parameter (α)	0.8
Measurement uncertainty (v_2)	4
Motion uncertainty (constant)	2
Initial estimate (m_1)	0
Initial uncertainty (v_1)	10000

Table 5.2. Case-Study Parameters

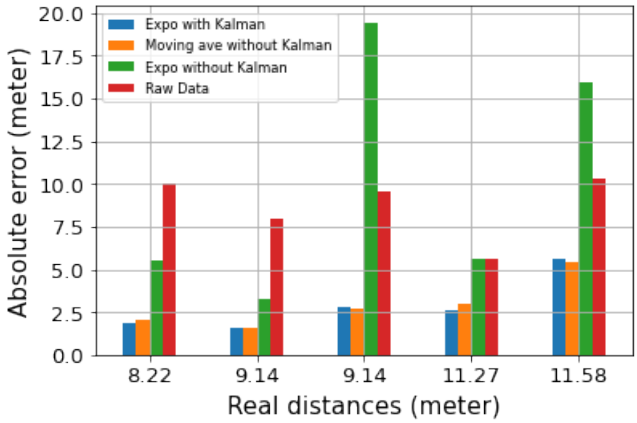


Figure 5.6: Average absolute distance error vs. real distance

distances from an IoT device with the same hardware specifications used in the real-world case study. The device collected packets for approximately 15 minute periods for each distance and logged the RSSI values from each packet. From then on, we evaluated different approaches for filtering the RSSI values for noise reduction. Specifically, we tested the proposed filtering (exponential filter and Kalman) against more lightweight methods, namely simple exponential filtering, moving average filtering [122], and using the raw RSSI values for the distance estimation with Eq. 5.4 (p_{d0} = -64.63 for d_0 1m). Regarding the exponential filter we tested different values for α (see Equation 5.1) such as 0.2, 0.4, 0.6, 0.8 and lowest average absolute error was achieved with $\alpha = 0.8$.

Fig. 5.6 and Table 5.3 show the average absolute and relative distance estimation error (compared to the real measured distance) for different user-IoT device distances,

	Relative Error (%)				
Real Distance (m)	***8.22***	***9.14***	***9.14***	***11.27***	***11.58***
Expo with Kalman	22%	16%	31%	23%	48%
Simple Moving Average	25%	16%	29%	26%	47%
Simple Exponential Filter	67%	36%	212%	49%	138%
Raw Data	121%	86%	105%	49%	88%

Table 5.3. Relative distance error vs. real distance

respectively. Results show that the combined approach followed in our framework that utilizes both exponential and Kalman filtering vastly outperforms the alternative. Specifically, the distance error of the filtered RSSI is reduced by approximately 6-8 meters in comparison to the raw RSSI approach.

5.3.3 System Parameter Evaluation and Overall Accuracy

Next, we evaluate the operation of our system using the real-time-generated data from the BLEBeacon dataset [140, 142]. Every entry corresponds to a single interaction between a user and an IoT device (in the form of a BLE advertising packet - see [74, 140]). Our analysis considers how our framework would operate during a single day (09/22/2016) and a single week (09/22/2016-09/29/2016).

First, we consider a single day of the experiment [74] and evaluate tuning parameters of our system, namely the update interval of each IoT device (T_o), and the centralized processing server interval (T_{CP}), as seen in Table 8.1. As mentioned earlier, the centralized processing unit outputs contact tracing results every T_{CP} seconds. We further merge the contact information for those users who have appeared in multiple consecutive (with the less or equal 1-second lag) batches and calculate their total overlapping

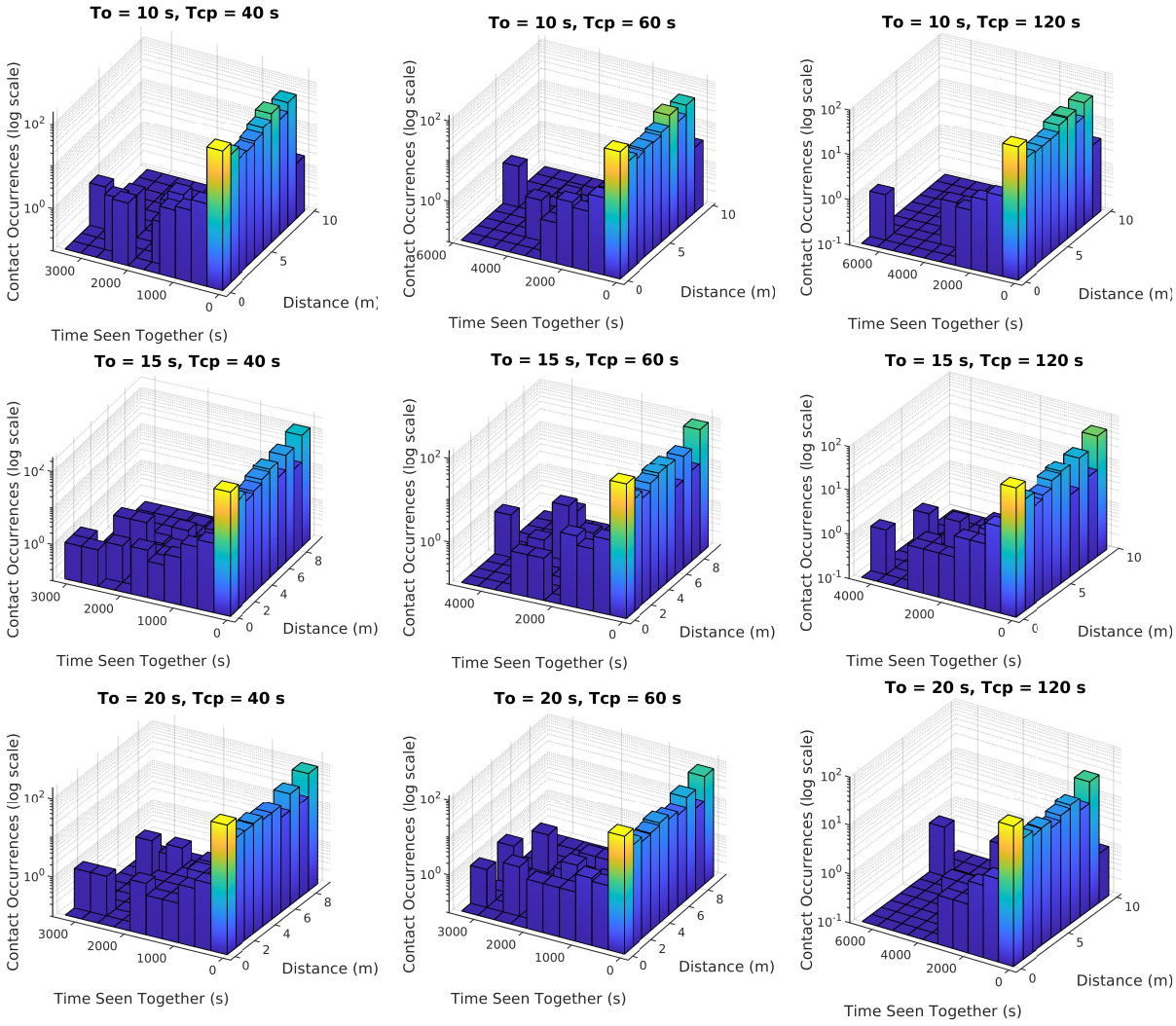


Figure 5.7: Contact occurrences vs. overlapping time and distance during a single day

time duration and average estimated distance. Figure 5.7 shows how adjusting the (T_o), and (T_{CP}) values affects the overall accuracy of our framework. Results show a similar number of occurrences regardless of the distributed systems' update/processing periods, attesting to our framework's capability to detect mobile users' contact. Therefore, the system operator will be able to adjust the T_o and T_{CP} parameters in order to optimize network traffic and computational efficiency for both the IoT devices and the centralized unit without losing the ability to detect user contacts (time duration and distance) accurately.

Next, we extend the period under observation to one week. We use an IoT device

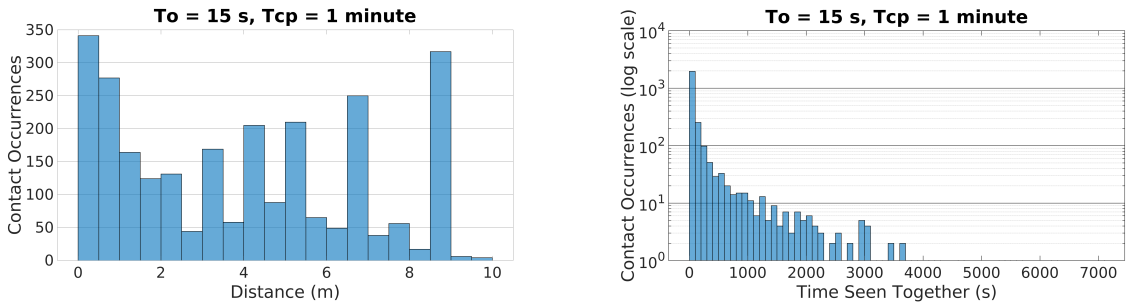


Figure 5.8: Overall number of weekly contacts vs (Left) distance, and (Right) overlapping time

Classifier	**Performance**			
	Accuracy	***Precision***	***Recall***	***F1-score***
XGBoost	0.85	0.81	0.87	0.87
	(0.83,0.87) 95% CI	(0.79,0.83) 95% CI	(0.85,0.89) 95% CI	(0.86,0.89) 95% CI
Bagging	0.81	0.79	0.84	0.86
	(0.79,0.84) 95% CI	(0.77,0.81) 95% CI	(0.82,0.86) 95% CI	(0.84,0.88) 95% CI
Decision Tree	0.77	0.75	0.83	0.83
	(0.75,0.79) 95% CI	(0.73,0.78) 95% CI	(0.81,0.86) 95% CI	(0.81,0.85) 95% CI
KNN	0.50	0.45	0.50	0.53
	(0.47,0.52) 95% CI	(0.43,0.48) 95% CI	(0.47 , 0.52) 95% CI	(0.50,0.56) 95% CI

Table 5.4. Algorithm Performance

update period of $T_o = 15$ seconds and central unit update interval of $T_{CP} = 1$ minute and apply the framework to a week-long portion of the BLEBeacn dataset. Figure 5.8 shows the overall number of contacts against the observed distance and their overlapping time. The overlapping time results show a strong power law distribution behavior, which follows previous results on the statistical behavior of dwell time for human mobility patterns [177].

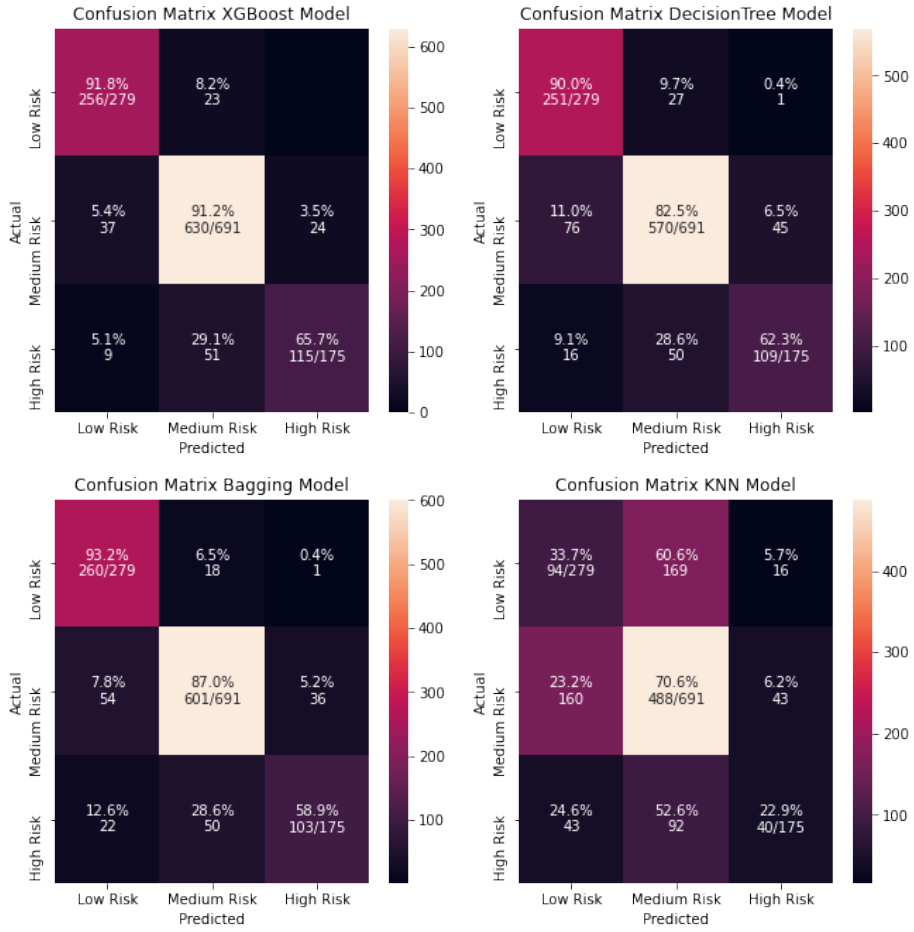


Figure 5.9: Classification performance through a Confusion-Matrix

5.3.4 Evaluation of risk classification

The interactions (contacts of user pairs and their data e.g., RSSI ranges and user estimated location) that were generated by the week-long study were used to evaluate our machine-learning algorithms for the infection risk classification part of our framework. We split the collected interaction data into a training (80%) and a test (20%) set. We first fit the parameters and weights of the hypothesis function into the training dataset and assess the learned model by using the test data that predicts user-risk. The labeling for this specific evaluation along with the utilized features is explained in Section 5.2. For all our testing we utilize the Python Scikit-Learn library.

We evaluate four machine-learning classifiers: Bootstrap aggregating (bagging), Deci-

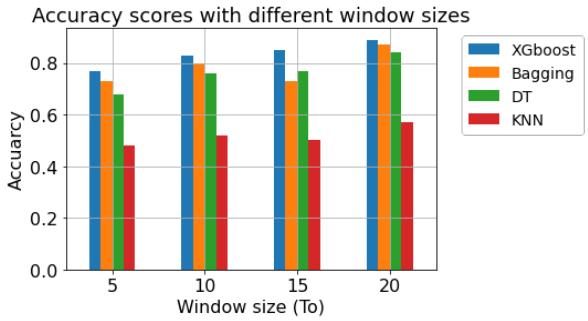


Figure 5.10: Accuracy vs. window size

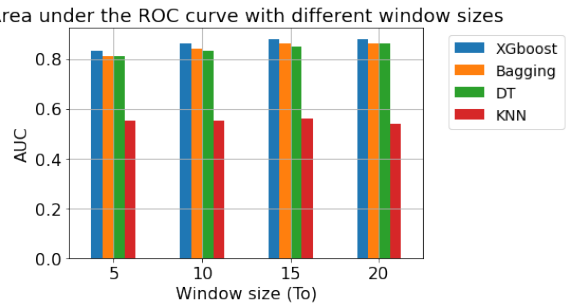


Figure 5.11: Accuracy vs. AUC

sion Tree (DR), Extreme Gradient Boosting (XGBoost), and K-nearest neighbors (KNN). Our study uses KNN as a baseline. We draw the confusion matrix to show each method's performance in Figure 5.9. We observe that XGBoost wrongly categorizes high-risk as low-risk 5.1% of the time and as average-risk 29.1% of the time. However, it correctly categorizes high-risk cases at a 65.7% rate. On the other hand, bagging wrongly categorizes high-risk as low-risk 12.6% of the time and as average-risk 28.6% of the time. The correct high-risk characterization is at 58.9%. The decision tree and KNN approaches perform significantly worse than the ones mentioned above. It is very important for the specific scenario of contact tracing that people who are at high risk are not classified as low risk. Therefore, in that aspect, XGboost outperforms all other models.

To calculate the confidence intervals for our models, we iterated the process 100 times and at each iteration randomly divided the data into 80% for training and 20% for the test. From Table 5.4, we observed that XGboost has 81% precision and 87% recall, while Bagging presents 79% precision and 84% recall. On the other hand, the DT presents only a 75% precision and 83% recall. In contrast, the worst-performing KNN has only 45% precision and 50% recall, which makes the approach inappropriate for this application. High precision and recall are translated to low false positive rates and low false negative rates. Therefore, XGboost has a better performance compared to other models.

5.3.5 Impact of Window Size

Finally, we examined the impact of the T_o window size on accuracy and area under the ROC curve (AUC). The outcome is illustrated in Figure 5.10. We can observe that DT has profited more by the window increase, and achieved a higher performance increase than other approaches (DT accuracy was raised from 68% to 84%). Additionally, as seen in Figure 5.11 the AUC score of XGboost is higher than any other method, while the AUC score of KNN presented no significant change. In total, we observed an increase in the accuracy of all models as the T_o interval grew. This behavior is attested to the fact that a larger window leads to larger local samples of RSSI measurements, and therefore a better risk estimation for the whole system. All related data and code is open-sourced and available in [5].

5.4 Related Work

Due to the 2019/2020 pandemic [61], a lot of contact tracing systems leveraging BLE technology have been studied, mainly due to its broad availability in wearable devices and its low consumption profile [12, 33, 123]. The work in [107] designed an algorithm estimating the distance between users in a Personal Area Network (PAN) through RSSI. Based on the estimation distribution and predefined zones (e.g., safe, moderate, and high-risk range), the system sends the appropriate alert to the users depending on their proximity. The authors in [116] proposed a BLE based contact tracing method where each user estimates his distance to nearby users using RSS measurements on their smartphones. The proximity measured by the users' smartphones is evaluated based on low and high-risk contact using five different machine learning classifiers. To protect users' sensitive information, the work uses non-connectable BLE transmission to disallow access to other users' smartphones. To harden the privacy, they propose a signature protocol where the BLE packet is encrypted by 31 bytes of information obtained from the

nearby devices. This signature will be unique and is unlikely to get duplicated on other occasions. However, this signature requires computation to be generated, preventing scalability, and potentially imposing constraints on network traffic. The work in [115] leverages smartwatches to measure the proximity between users for contact tracing purposes. The proposed application also leverages BLE technology to make contact tracing more convenient for users that do not always carry their smartphones.

The authors in [72] discuss the impact of enhanced contact tracing using sensing technologies. The work proposes an IoT-based contact tracing architecture consisting of three components called user endpoint (UE), facility endpoint (FE), and object endpoint (OE) that measure the infection risk over time. The work in [156] also focuses on an IoT-enabled architecture for contact tracing and focuses on privacy and security issues. Their proposed generic architecture delegates technical contact tracing tasks as well as privacy/security tasks to the deployed IoT devices and health authorities towards avoiding the computational and power limitations of mobile devices carried by users in public places.

Finally, we have recently observed an explosion of the use of blockchain technologies in smart city applications that span across multiple fields from local energy markets [39], and smart grid infrastructure security [139] to e-voting systems [11]. Thus, multiple works discuss and investigate the use of blockchain technology for secure data sharing in hospitals [86], and even for COVID-19 detection using CT Imaging [85]. The work in [78] discusses various COVID-19 pandemic challenges and investigates the applicability of blockchain to face them along with the expected performance. Similarly, the authors in [8] investigate practical blockchain applications that attempt to mitigate COVID-19 challenges. The work in [166] presents BeepTrace which is a privacy-preserving blockchain-enabled contact tracing framework. The focus is on reinforcing privacy and security and allows for the decentralization of user identification and location-based information. Finally, the authors in [97] propose a contact tracing framework that uses blockchain

technology to decouple any connection between the ownership of on-chain location information and user identity. The proposed framework is evaluated using Bluetooth 5.0 with the focus being on storage/CPU usage, observed delay, energy consumption, and mainly security vulnerabilities.

5.5 Conclusions

Fast and accurate contact tracing systems can assist in preventing the spread of highly infectious diseases like Covid-19. This requires a cohesive and complex system. We propose a passive contact tracing framework that relies on passive interactions between users and an IoT infrastructure. The framework relies on proximity sensing while performing risk infection classification using well-known machine-learning techniques. We have also performed a case study of the system using a real-world dataset of BLE-based interactions. Our experimental results suggest that our design can substantially help monitor the contact occurrences between people in smart cities constantly and accurately both as a standalone system or as a complementary system for other contact tracing applications.

Chapter 6

Multi Agent Based Traffic Prediction for ANMS

Traditional networks consist of several devices including switches, routers, servers, deep packet inspection, firewalls, and hosts. These devices are traditionally implemented using hardware. A human administrator manages such small networks [96, 133], through configuration and management interfaces. Typically, an administrator is necessary to configure and make the required changes. This is feasible only for relatively small size networks, even though with limited performance and flexibility in terms of resource utilization, energy efficiency, service admission, latency, reliability, etc.

Nevertheless, within the last two decades, communication networks have tremendously evolved. It is now possible to interconnect a massive number of devices and applications including traditional users, IoT devices, and machines. For instance, in the wireless domain, 4G and 5G have shown a multi-fold increase in the network size. The advent of IoT and other machine-to-machine communication paradigms also hugely increases the network size. The legacy network implementation has an enormous capital expenditure (CAPEX) since such networks are hardware-based and manually operated, which is inefficient and inflexible. As the network grows larger, it is increasingly difficult and expensive to manually administer its complexity, implying an increase in operational expenditure (OPEX). This is mainly because a massive number of technical personnel and experts are needed and sometimes humanly challenging and inefficient to perform, resulting in degrading network performance.

Moreover, it is expected that Beyond 5G (B5G) and 6G will increase in size and become more complex. Furthermore, as indicated in 6G deliverable [68, 69], intelligence and autonomy are expected to be the core principles that drive future networks B5G and 6G. In this direction, an approach for in-network intelligence and automation is required

to address the expected network growth as well as stringent requirements of various services. Currently, software-defined networking (SDN) and network function virtualization (NFV) are enabling network softwarization, subsequently increasing network flexibility and opening the door to autonomic networking. SDN provides a programmable network while NFV softwarizes network functions, decoupling the hardware-based implementation to general-purpose and cloud-based deployment [98].

Autonomic networking is the process of enabling the network system to have a self-X property; X can be managing, configuring, healing, and protecting the network [105]. This originally aimed at creating autonomic computing by IBM, [30,133]. One of the reasons for having automated network management systems, is the idea of dynamic network resource allocation [63, 67] based on the service traffic arrival statistics. A self-managing network should be able to autonomously predict, diagnose, reconfigure, monitor, and circumvent failures while providing service processing as per the user traffic demands. This means, an autonomous network management system (ANMS) can collect, analyze, act, while adapting to the network and user dynamic and their evolving behavior. Using ANMS, networks are capable of self-management and autonomous decisions both in unsupervised and supervised manner [158]. This guarantees a simplified control, management, and operation of complex and vast networks. These research works are classic approach for network automation that we developed before the introduction of the softwarized networking which transformed the network management, design and configurations.

Recently, MANA-NMS architectural framework has been proposed in [21, 23], using a multi-agent system (MAS). Defining atomic decision-making units, the authors discussed the concept of network function atomization. Such atomic units could replace the virtual network functions (VNFS) with atomized intelligent agents. The approach is a service-oriented architectural (SOA) design for flexible, loosely coupled, and intelligent system development for future generation networks. Moreover, it is a competing approach for microservice-based design with proactive and reactive intelligence. MANA-NMS used a

MAS due to its advantage over microservices in terms of autonomy and intelligence. In the proposed ANF, the agents are autonomous with adaptive functionality in performing a given task. An ANMS can be built composing these service agents. Alternatively, MANA-NMS framework can be taken as a splitting of a monolithic system into agents and recomposing them to have autonomous system: e.g., SDN controller Evolved Packet Core (EPC), and Network Management System (NMS). Currently, network function such as EPC, SDN, and NMS are monolithic systems that can be split into the smallest possible functions [25, 64]. The decomposed units can be implemented as microservices [25] or agents [21]. This is an appealing paradigm in the current trend of softwarization, service function virtualization or containerization, cloudification, and orchestration. This helps deploy network functions (possibly autonomous agents) as loosely coupled, elegant, and lightweight applications in a virtualized/containerized, distributed, and cloud/edge environment with maximum re-composition freedom. Both papers [21] [23] only discussed a theoretical model and the performance evaluation is using mathematical evaluation.

A similar approach with a focus on agent based algorithm is proposed in [62]. The authors presented a software agent to execute a set of computation loads in parallel, which enables a bottom up systematic organization of the overall system. The paper discussed IoT objects to be an agent that is autonomous and able to collaborate with other peer agents. However, the approach is focused on high level objects and didn't consider protocol level and function level functionalities. Moreover, it is needed consider service design approaches in comparison with microservices showing how to incorporate ML/DL algorithms in the agent design.

In this paper [22], using the design guideline from [21], we implemented the internal composition of an NTPA and NTCA. We briefly introduce the ML and DL algorithms that are used as the cognitive component of NTCA and NTPA. We also proposed and implemented possible communication interface technologies. In addition to the legacy agent communication language (ACL), we suggested Restful-API, gRPC, and Websocket.

This paper could be considered as a continuation towards implementation of MANA-NMS, the automation architecture proposed in [21] and an extended version of the work in [136]. The authors in [136] have only considered classification agent design with limited evaluation that doesn;t consider the resource consumption of agents. This is crucial in that agents are mostly deployed in a containerized environment with a resource constraint environment, especially as the complete system may need a number of agents to be instantiated.

In [21], the authors used mathematical models and evaluation and left the implementation as future work. Therefore, this article tries to implement and simulate the ML/DL-based NTPA and NTCA and evaluate critical measures such as accuracy and latency. The benefit and advantage of designing NTPA and NTCA agents are to use these agent as a service unit in the multi agent system. In the simulation section, an example of the complete MANA-NMS system architectures are provided incorporating traffic classifier agent and traffic predictor their role in the overall system as an example. The contributions of this work are:

- Design the internal architecture of NTPA and NTCA.
- Implement and simulate both agents.
- Evaluate the performance of NTPA and NTCA using accuracy prediction and classification, training latency, prediction and classification latency, and resource consumption, respectively.

The rest of the paper is organized as follows. Section 6.1 presents an overview of the network automation and enabling technologies. The prediction and classification agents design architecture along with agent communication technologies is discussed in Section 6.2.1. The implementation scenario for the proposed conceptual framework and dataset description are presented in Section 6.3. Section 6.4 presents the performance evaluation of the proposed NTPA and NTCA. Lastly, Section 6.5 concludes this work.

6.1 Overview of Network Automation and Enabling Technologies

This section provides a review of important concepts for network automation, including the applied ML/DL algorithms and multi-agent systems.

6.1.1 Overview of Network Management Automation

Autonomic network management is proposed as one of the solutions for the management of large and complex networks. As indicated above, it comes from autonomic computing, a manifesto started with IBM in 2001 [23, 105, 133]. IBM introduced the concept of self-management and self-adaptation of networks. Self-management implies that a network can make decisions on its own while self-adaptation means that the network is aware of its environment and can adapt the performance according to the changes in the environment. The autonomic systems described by IBM are equipped with four self-management properties [23] such as self-configuration, self-optimization, self-protection, and self-healing. These properties are discussed in details below.

- Self-configuration offers the opportunity to discover new or evolving devices on a network, and to automatically establish routes and other configurations required to seamlessly connect the device.

- Self-optimization is the ability of the network to change its settings to match its actions better with the system's user-defined purpose.

- Self-protection requires a network to protect itself from a possible attack such as denial-of-service (DoS) attacks or modification of firewall rules based on alleged malicious traffic.

- Self-healing is the capacity to correct, by itself, any issues that may occur, whether or not from the attempted behavior of the device.

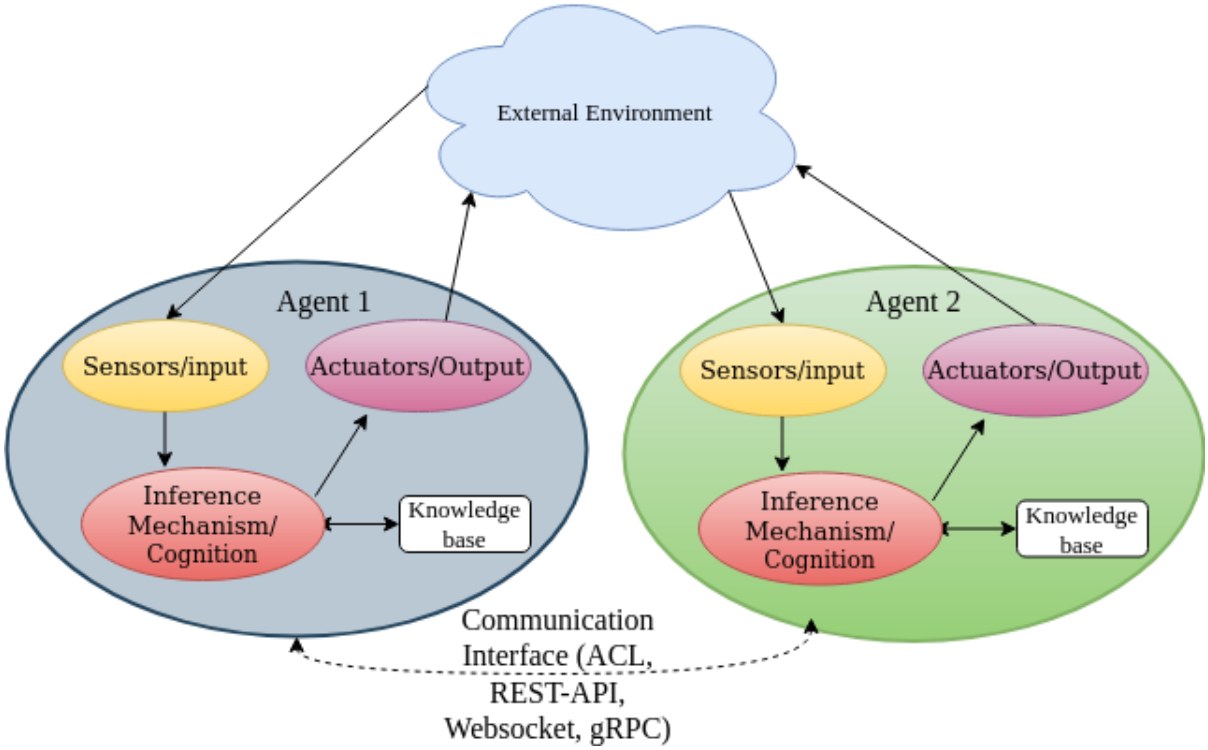


Figure 6.1: A Simple MAS Architecture.

6.1.2 Overview of the Multi-agent System

Problem solving in complex environments requires distributed approaches. As a solution *Distributed Artificial Intelligence* (DAI) is proposed where the system consists of many intelligent agents interacting with each other to achieve the goal [47, 137]. DAI, traditionally, suggests two types of solutions. The first one is to break a problem into sub-problems. Then, micro solutions are provided to solve each sub-problem individually. The second solution is MAS where autonomous agents cooperate with each other to provide a service.

Figure 6.1 illustrates the general architecture of the MAS. Similar to distributed problem solving using microservice solutions, the agent work with other agents or individually to perform a given task. The agent also measures the environment continuously to obtain updated information about the environment [46]. The core characteristics of the MAS will be discussed in section 6.2.1 in context of our agent definition .

6.1.3 Agent as a Decision Making Element in Autonomic Networking

Agents have different definitions based on their applications in different environments [46]. Before defining network agents, we present a general definition incorporating all characteristics of the agents. *Agent: An object that observes an environment and uses various parameters to execute an appropriate action at each step and achieve the goal.* This definition is made of four keywords that are further explained below.

1. *Object*: This refers to the kind of agent in different contexts. For example, a software agent, a hardware agent, etc.

2. *Environment*: The environment receives the agent's actions and updates itself accordingly. As a result, the agent observes the changes and adjusts its actions.

3. *Parameters*: These are the information that agents need to know to take the appropriate action.

4. *Action*: The agent executes actions that result in changes in the environment. The action space can be continuous or discrete, .

Each agent decides to perform its action with regard to time and resource constraints [46]. In order to perform a good action, the agent needs to interact with the environment as well as communicate with other agents through ACL, REST-API, gRPC, Websocket, as illustrated in Figure 6.1 to learn about the information. This information contains the previous agent's experience from interacting with the environment. This helps the agent with its decision-making process [130].

In this work, we present the MAS as an automated network management system where the agents are known as the network decision elements (DEs).

6.1.4 Application of Traditional ML and DL as a Cognitive Component of Agents

As indicated in the introduction section that will be further discussed in the next section, one of the internal elements of an agent is the cognitive or reasoning component which enables the agent to perceive, reason, and decide. This would enable designing agents as the smallest atomic element with autonomous and intelligent capability in performing a given task. Using such elements, we build the complete system. This is in line with the future service-based network design in a softwarized, containerized, and intelligent network environment. Incorporating in-network intelligence in the network functions is the prime target in 6G [68–70]. Hence, we require to incorporate ML/DL to create an intelligent agent such as NTPA, NTCA.

In the next paragraph we discuss the DL definition and its advantages and disadvantages over ML methods.

DL is a sub-field of ML that enables learning and decision capability in the agent, training the agent to learn like a human brain. DL statistics and predictive modeling enable agents to process massive datasets. It uses a layered structure and applies a non-linear transformation to its input iteratively to learn the features and create a statistical model as output. There are several DL techniques that help with creating accurate predictive models from a large amount of unlabeled and unstructured data. We will describe some of the techniques as follows:

- Learning rate decay: is a hyperparameter in DL settings controlling the learning process of the model every time the weights get updated to avoid unstable training and long training process.
- Transfer learning: this method uses an existing network trained previously. It feeds the new data along with previously unknown samples to the existing network and

once the adjustments are made it feeds the new samples to be classified. This method requires much less data, thus reducing computation time.

- Training from scratch: This method requires a large amount of labeled data to train the network. However, this technique may be impractical for some applications as it needs a tremendous amount of data to be collected and make the training process last too long.

- Dropout: Training a neural network on supervised learning tasks may cause overfitting. To avoid this problem, the dropout method drops units and their connection randomly from the neural network.

Although various neural network architectures have been proposed such as RNN, CNN, and feedforward neural networks, they all function in similar ways. They benefit from a layered architecture where input is given and the model figures out itself whether it has made the right decision about the data element by a trial-and-error process. Next, we list some advantages and disadvantages of the DL methods and compare it with ML algorithms.

Unlike the general ML, DL does not need to be given the feature set to make decisions. It learns the features by interacting with the environment and builds the feature set incrementally. Therefore, DL methods do not require the programmer to specify the features the computer should be looking for. In other words, DL is able to learn without human intervention. While this makes the DL methods take so much time to train, they are much quicker in testing than the ML algorithms. ML test time increases as the data size grows. However, there are some issues with DL models making data scientists choose traditional ML over DL. Its neural networks rely on the trial-and-error process. In any case, we used different ML and DL models for the NTCA and NTPA design and tested their performance.

LSTM-based Network Traffic Predictor Agent

LSTM is an improved form of the traditional Recurrent Neural Networks (RNNs) addressing the vanishing and exploding gradients problems using connected memory blocks in the layers. Each block consists of multiple cells containing three units: the input, output, and forget gates. Unlike the standard RNNs, this block design enables LSTM to capture long-term temporal dependencies in sequence learning problems. The network interacts with the cells through their gates. The role of the input and output gates is to achieve a constant error flow and avoid the irrelevant memory contact, respectively. However, an unstable error flow is observed through the backpropagation in existing RNNs [125, 168].

GRU-based Network Traffic Predictor Agent

GRU is a variant of LSTM and the improved version of standard RNN [168]. A GRU cell consists of two gates: update gate and reset gate to solve the vanishing gradient problem in recurrent neural networks. First, the reset gate is used to filter out the irrelevant information from the past time steps and create the memory content at the current time step. Next, the update gate is used to create the final content of the memory by deciding how much information still needs to be kept and passed to the output layer [125,168]. We describe the usage of the reset and update gates with mathematical notation below: First, the reset gate (similar to the forget gate in the LSTM setup) filters out the irrelevant past information as shown in equation 6.1.

$$m'(t) = tanh(Wx_t + r_t \odot W'm_{t-1}) \tag{6.1}$$

Where $m'(t)$ represents the current memory content. The element-wise product between the reset gate and weighted previous memory content ($r_t \odot W'm_{t-1}$) determines what information needs to be removed from the past time steps.

Next, the update gate determines what information from the current memory content and the previous time steps need to be passed through the network for computing the final result as shown in equation 6.2 where u_t is the update gate and controls the memory content.

$$m_t = u_t \odot m_{t-1} + (1 - u_t) \odot m'_t \tag{6.2}$$

MLP-based Network Traffic Predictor Agent

MLP is a fully connected feedforward artificial neural network composed of many perceptrons (neurons) that are organized into at least three layers, an input layer, a hidden layer, and an output layer [114,161]. Each neuron in the layer except the input layer utilizes a non-linear activation function to learn the non-linear data. The learning process in an MLP is described as, when the data is given to the MLP, it goes through its layers. Each node in the layer computes its error by comparing its output and the expected result. The goal is to minimize the error and then adjust the node weights [161]. The error is calculated by the following equation:

$$e_j(n) = d_j(n) - s_j(n) \tag{6.3}$$

Where d_j is the target value and s_j is the value generated by the node j. The error for processing data point n by the nodes is obtained by the equation below:

$$E(n) = \frac{1}{2} \sum_j e_j^2(n) \tag{6.4}$$

Next, weight adjustment is calculated using gradient descent:

$$\Delta w_{ji}(n) = -\alpha \frac{\partial E(n)}{\partial a_j(n)} s_i(n) \tag{6.5}$$

Where s_i is the output of the previous node and α is the learning rate to ensure that the weight corrections converge quickly, and a_j is the induced local field or activation

potential.

CNN-based Network Traffic Predictor Agent

CNN is a kind of feed-forward neural network that has major application in image classification [114]. It has also applications in time series [**?**] where we can utilize it for network traffic prediction. It consists of an input layer, convolution layer, pooling layer, fully connected layer (as hidden layers), and an output layer. The convolution layer is a good substitute for a fully connected layer as it is scalable to massive datasets. The role of the convolution layer is to reduce the dimension of input images without losing critical features, allowing the network to be deeper and learn more easily.

A part of a convolution layer is the kernel/filter that has smaller dimensions than the input image and moves over the image and performs matrix multiplication until the entire image is traversed. This process produces convoluted feature output extracting high-level features of the image such as edges. The next layer is called the pooling layer. The pooling layer is in charge of reducing the spatial size of convoluted feature output, even more, using two operations: max pooling and average pooling. Max pooling returns the maximum value of the calculated convoluted features corresponding to a portion of the image. Average pooling returns the average of all values generated by the kernel/filter. Finally, a fully connected layer (multilayer perceptrons) is added to classify the images using the Softmax classification technique.

Decision Tree-Based Network Traffic Classifier Agent

We have presented four possible ML solutions for NTCA design in [25] in more details. For the readers' convenience, we briefly discuss these traffic classifier ML algorithms.

A non-parametric-based supervised learning approach utilized for classification is a DT. DTs learn from knowledge to approximate a sine curve using a sequence of if-then-else decisions. The longer the tree, the more complex the rule of decision in fitting a

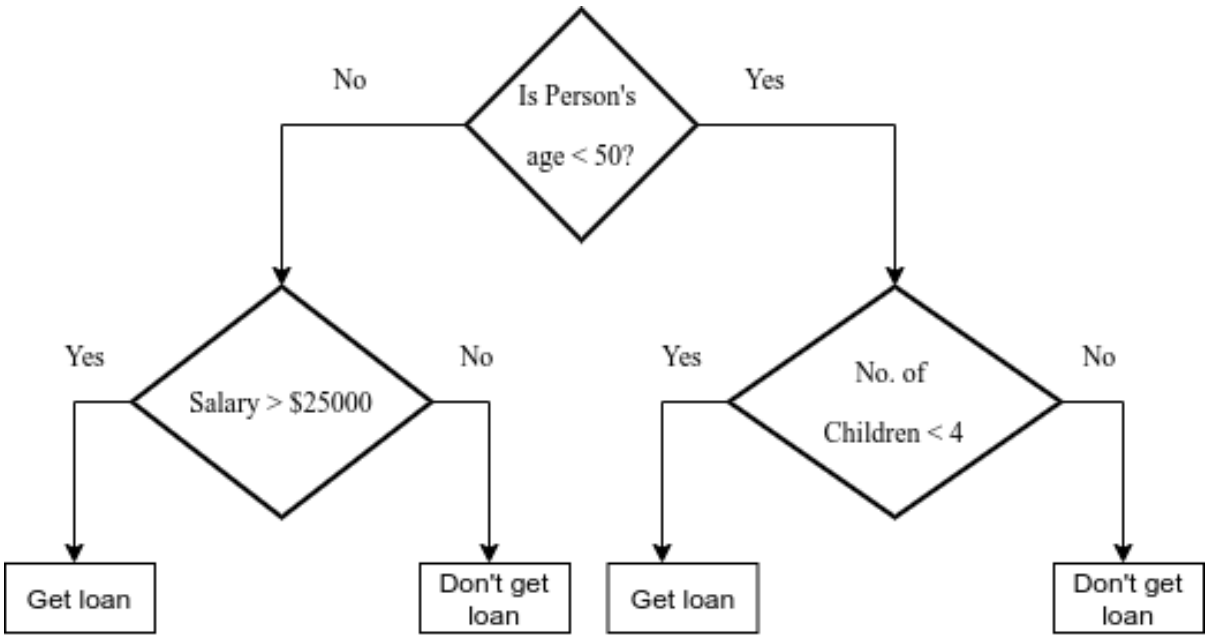


Figure 6.2: An Example of a DT.

given model [113, 127]. Figure 6.2 illustrates the DT diagram.

Naive Bayes based Network Traffic Classifier Agent

NB is a classification technique that given a finite set of class labels, assigns the labels to the data samples using a probability distribution [1]. It assumes each feature value independently builds the probability specifying the data sample's class regardless of the correlations between the features.

Support Vector Machine Based Network Traffic Classifier Agent

SVM attempts to find a hyperplane (decision area) having the maximum distance between the two categories of data points in high or infinite dimensional space [1, 3, 127]. Figure 6.3 illustrates the SVM diagram.

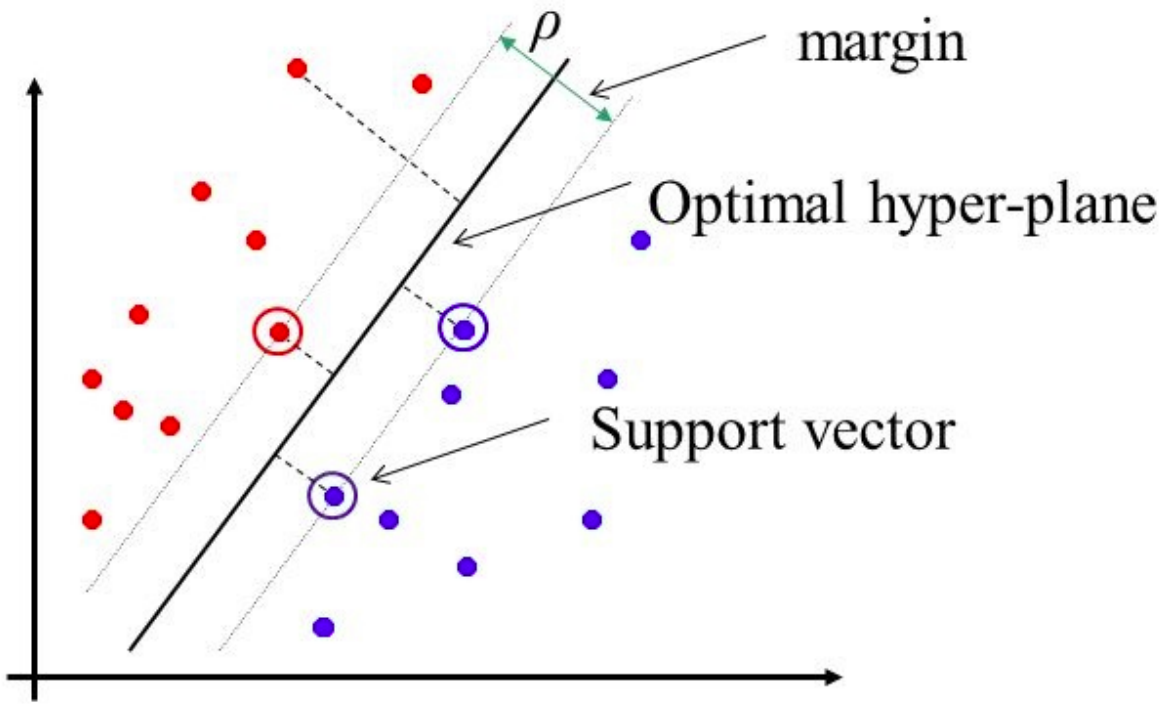


Figure 6.3: An Illustration of a Hyperplane and Support Vectors.

K-Nearest Neighbours Based Network Traffic Classifier Agent

K-NN classifies the data point based on the most common class among its k nearest neighbors [127]. Figure 6.4 shows how K-NN can help one to identify the class of a new data point.

6.2 Proposed Network Traffic Predictor and Traffic Classification Agents

In this section, we discuss the proposed NTPA and NTCA frameworks. First, let us re-define and contextualize the agents. The agent has input, output, and communication tools to observe the environment, provide the output or execute the action, and communicate with other existing agents in the MAS. Once an agent is employed in a given environment and it observes the environmental variables, it can make decisions. The decision is based on the agent's final goals, which could be to predict the incoming traffic

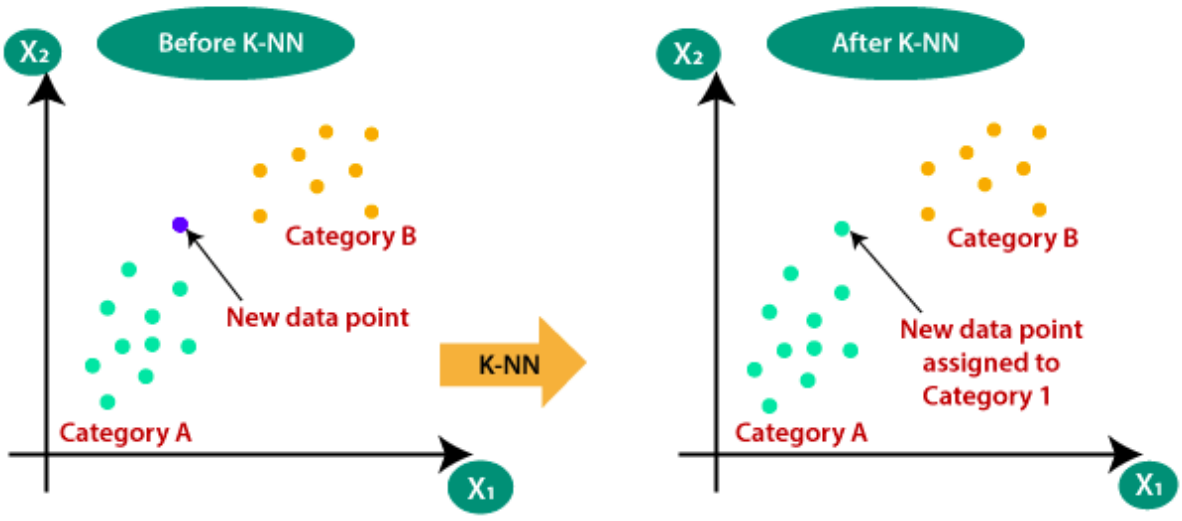


Figure 6.4: An Illustration of the K-NN Classifier Algorithm in action.

and classify them accordingly. By measuring the incoming traffic over a given link, it could analyze and use it for load balancing, resource slicing, or resource allocation. The important characteristics of an agent are:

- Situatedness: that means agents use sensors or perception capabilities as an interaction with the environment, and it uses actuators to do actions on the environment. In the NTPA and NTCA context, it means taking incoming data as an input and providing prediction and classification results as an output, respectively.

- Autonomy: means the agents' internal state is protected from any external disturbance from other agents in the MAS. NTPA and NTCA are capable of performing their respective function autonomously without any external support.

- Inferential capability: this enables agents to work on an exact goal until it is achieved. Agents can analyze the available data for decision making such as network traffic prediction and traffic class determination.

- Responsiveness: is defined as the ability of an agent to perceive the environment and perform on it with minimal latency, which is important in real-time processes.

Agents should be designed such that the task execution (prediction and classification in case of NTPA and NTCA) must be performed with strict time delay.

- Pro-activeness: means agents should take advantage of particular opportunities that aim to improve their action performance, to dynamically adapt to the changes in the surrounding environment of the network. By analyzing the historical data, it could predict the amount of traffic that comes in a given link. Using this knowledge, it pro-actively allocates the required resources such as bandwidth.

- Social behavior: the agent's decision should not be affected by any external interference either from human interactions or other agents in the system. NTPA and NTCA should be capable of performing independently but also could communicate and collaborate to perform a sequence of tasks such as classification followed by prediction or vice versa. For instance, NTCA could classify traffic into various classes and provide the output to the predictor. Based on the classes, the predictor could predict the amount of incoming traffic of a given time in the coming hours or days.

6.2.1 Proposed Network Traffic Predictor Agent Architectural Framework

A given NTPA is equipped with an "input unit" that enables it to receive a network traffic processing request and network state sampling/measuring points. Furthermore, NTPA is equipped with several facts that are model features that could be the amount of incoming network traffic, the available bandwidth in a given link, IP addresses, source and destination ports, and labels (protocols) [46]. The main target to focus on is the "cognitive/reasoning unit". It is the brain of NTPA enables it with the reasoning capability, which is typically an DL model. The "planning strategy" component of NTPA, is the steps for the action to be performed in accomplishing the required processing. Another component of NTPA is a validation unit. It is composed of validation rules for

the final decision; for example, knowing the amount of arriving network traffic of given traffic classes that helps make decisions. Finally, we have an "output or action" unit. This is the final outcome, that could be a prediction result to be sent to the other agents. Figure 6.5 (bottom) shows the building blocks of an NTPA.

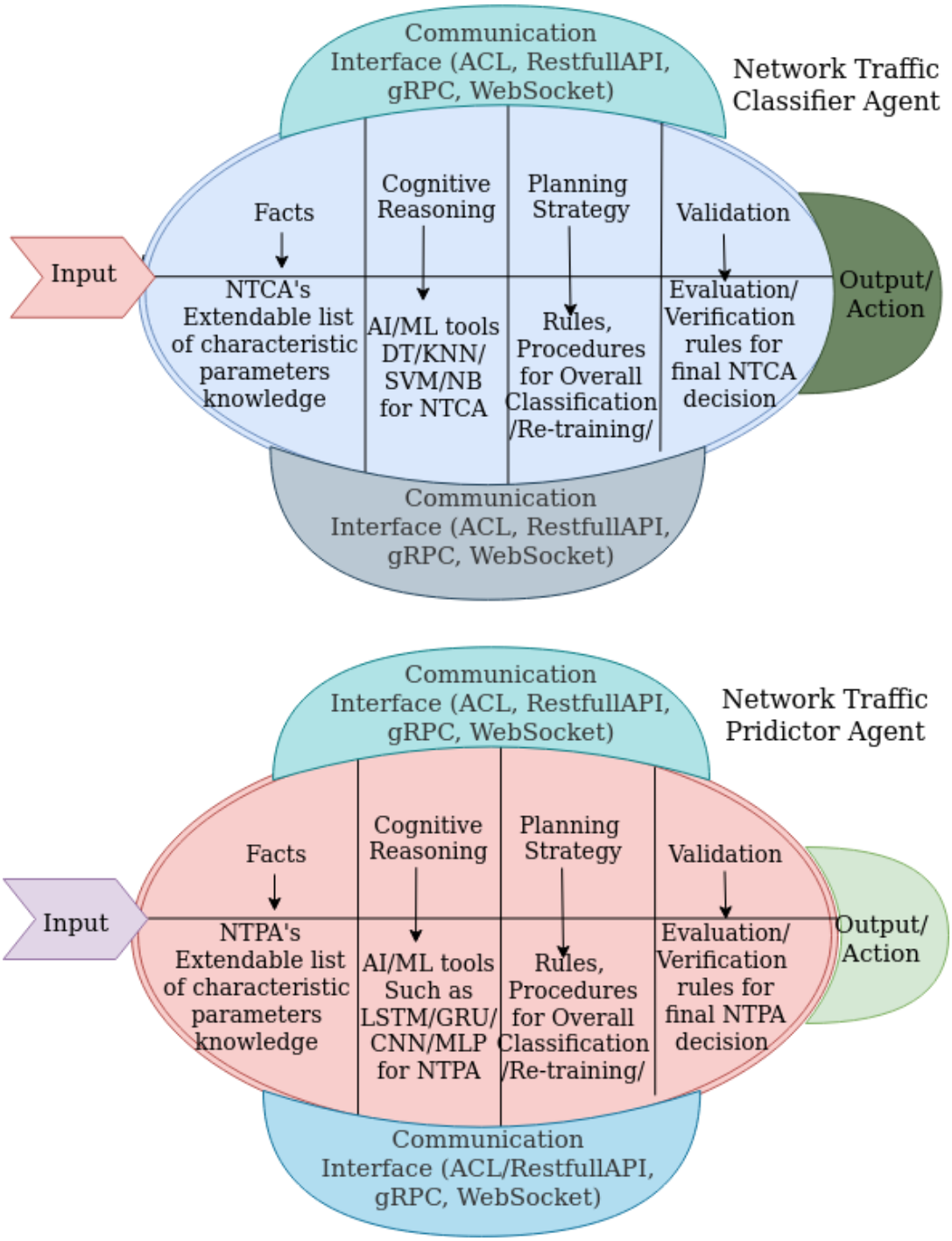


Figure 6.5: Network Traffic Classifier Agent (top) and Network Traffic Predictor (bottom) Agent Architecture.

6.2.2 Proposed Network Traffic Classifier Agent Architectural Framework

The general architecture of NTCA is similar to NTPA. However, the actual values, parameters, and algorithms used are different. That means input parameter, knowledge base, cognition algorithms, validation technique and the output for NTCA are different from NTPA. In general, taking network traffic as input, NTCA classifies them into the classes to which the traffic belongs. NTCA helps other network agents in making network decisions including QoS provisioning and monitoring based on SLA, resource provisioning, and network slicing. The NTCA's design and implementation framework is depicted in Figure 6.5 (top). Figure 6.17 clearly illustrates the statistical differences in the memory used by all the four traffic classifier agent designs.

In particular, simpler than NTPA, NTCA has an "input-unit" that can be used in receiving a service request and network state sampling/measuring points. In addition, NTCA has facts i.e., incoming traffic features, which could be the size of a packet, source and destination IP addresses, protocols, etc. The "cognition/reasoning unit" is the component that gives the central intelligence for NTCA. It uses it to identify and classify traffic. The "planning strategy unit" is the steps/procedures in providing the class of the traffic. Another component of NTCA is a validation unit. It is composed of validation rules for the final classification decision, i.e., knowing the amount and type of arriving network traffic of given traffic classes that helps make decisions. Finally, the remaining NTCA component is an "output unit" which could be a classification result to be sent to other agents.

6.2.3 Communication Interface Between Agents

The communication interface enables the creation of an agent chain for collaboration and cooperation to create a complete MANA-NMS. Agents could be deployed in a containerized and distributed environment that communicate based on open-source communication interfaces. The standard communication interface for agents is ACL [146]. ACL

enables a complex knowledge and information exchange mechanism which includes facts and knowledge base, the agent's goal, strategy, plans, and outputs. We used ACL as a communication interface in our simulation. However, other generic communication interfaces are tested in our work in [25]. These are REST, WebSocket, and gRPC. Each of them has its pros and cons in terms of latency for web-based communicating intelligent agents.

6.3 Implementation Scenario and Conceptual Framework

We used the osBrain [43] library for the design and implementation of NTPA and NTCA. osBrain is an agent simulation environment implemented using Python.

We provide the hardware configuration information shown in Table 6.1 as it affects the prediction results. Therefore, the same hardware was used to run the LSTM, GRU, CNN, and MLP models.

Agents within the MAS are running independently as system processes (uses multiprocessing). Since osBrain uses ZeroMQ [174], agents communicate using zeroMQ. zeroMQ is a messaging library for flexible and efficient communication between agents. It is an open-source universal messaging library that uses asynchronous communication between agents in the MAS. The basic communication patterns used by osBrain are *Push-Pull*, *Request-Reply* and *Publish-Subscribe*.

Table 6.1. Hardware Configuration

Hardware	**Capacity**
Memory	16 GB 2133 MHz LPDDR3
Processor	2.7 GHz Intel Core i7
Storage	2 Terabytes

6.3.1 Implementation Conceptual Framework for MANA-NMS using NTPA

NTPA has been implemented in a MANA-NMS framework as a proof of concept. The MANA-NMS architecture is depicted in Figure 6.6 showing the integration of NTPA along with the dispatcher, the audit log agents, and other service processing agents. NTPA mainly uses LSTM. However. we also used GRU, CNN, and MLP algorithms to compare the performance against LSTM-based NTPA. MANA-NMS is assumed to be deployed in an edge/cloud data center, where the incoming traffic is collected at the gateway. We assumed a Poisson arrival process for service arrival into the system. Poisson arrival process is considered with a 1^st degree approximation of the traffic arrival. However, other models could be used such as fractal process arrival.

We collected the amount of incoming traffic to observe the service request distribution in time. The data is collected for 1 year period. To store and manage the history of traffic data for future retraining of agents, a centralized database can be used. Alternatively, it could be possible to provide the agent with database management capability. In the agent interaction dynamics, agents will be responsible for pulling results from other agents and logging their actions. NTPA uses the collected dataset to train or retrain its' internal cognitive reasoning algorithms. Training of the DL component of NTPA could be done centrally (remotely) or locally (at the agent). The training process is done periodically.

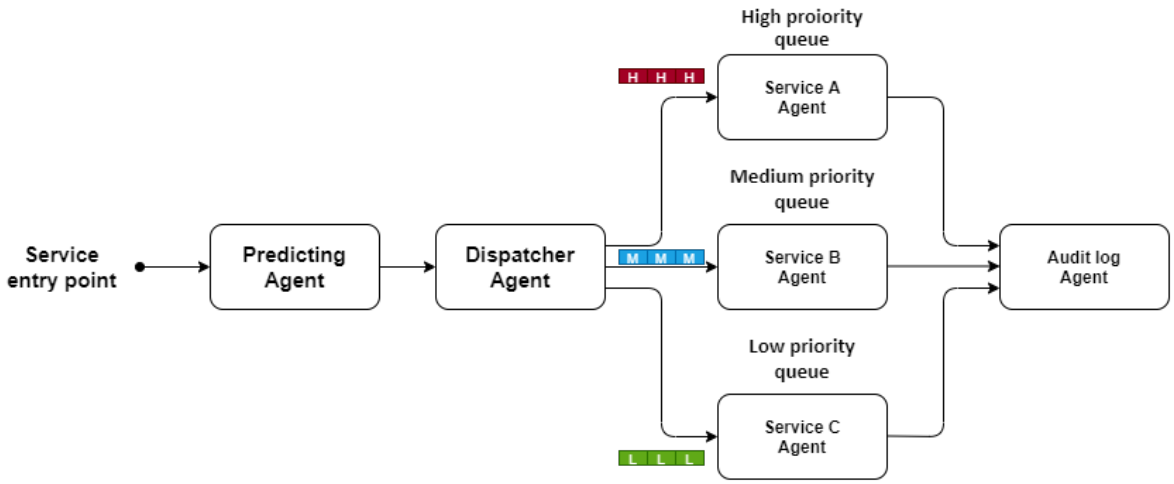


Figure 6.6: Simple Architecture for the MANA-NMS.

After training is done, NTPA predicts the incoming traffic for the next hours/days. The predicted data indicates the real incoming traffic demands at a given time.

By using the NTPA's prediction traffic values, the resources/agents are dynamically allocated, creating agent chain for that particular period. In other words, the predicted value will be used to instantiate the number of service agents by the orchestration agent, through a request communication protocol channel. Then the orchestration agent creates the required agent chain that is considered as the network resources to perform the required service execution. This can save a lot of unused resources at some time of the day. By doing so, we optimize the performance of the network, reduce the running cost, and increase the network efficiency.

Moreover, we have a dispatcher agent (event handler agent), whose goals are to: add the incoming daily traffic data/services in a FIFO queue, divide the traffic into three different queues consulting the NTCA and based on the task/service priority whether high, medium, low, and communicate through three different communication channels to the service agents (A, B, C).

Scheduling agent schedules the incoming tasks using First-In-First-Out (FIFO) queuing algorithm while using classifying agent to classify the tasks into high, medium, or

low priority classes. This represents three main application areas in 5G network Ultra-reliable Machine-Type Communication (uMTC), Massive Machine-Type Communication (mMTC) or Extreme Mobile Broadband (xMBB), respectively. The service agents execute tasks scheduled in three different queues depending on the task priority. They represent the complementary service agents in the MANA-NMS. Three type of service agents assumed to serve 5G/B5G/6G traffic, which are: a) service agent type A serving the high priority queue (uMTC), b) service agent type B serving the medium priority queue (mMTC), and service agent type C serving the low priority queue (xMBB). Finally, once the service agents finish their processes, the audit-log agent will pull the results through three separate communication channels and save them.

6.3.2 Implementation Conceptual Framework for MANA-NMS using NTCA

Similar to the NTPA, we used the NTCA design guideline presented above, as depicted in Figure 6.5, for the agent implementation. Using the input traffic, NTCA classifies the incoming network traffic into the appropriate classes. NTCA helps with important decisions in network management, for instance for QoS provisioning, resource provisioning, dynamic network slicing, etc. In performing autonomous service management and provisioning; such as resources provisioning, and task scheduling, it is necessary to classify all incoming network traffic as per the protocol to which it belongs to. The Protocols indicate the application or category the traffic/user or service belongs to. The NTCA uses an ML algorithm as an internal cognitive part, to perform the required traffic classification.

Supervised ML algorithms are employed for the NTCA's implementation. The supervised ML used a labeled (protocols or port numbers) network traffic dataset. The supervised learning models used are K-NN, DT, SVM, and NB and their performance is compared. We also assumed the NTCA can collect incoming network traffic and store

it as a historic dataset. This dataset will be used by the agent to label and utilize it in training and re-training the internal cognitive component. The assumption is that the dataset will be stored in a centralized/distributed database: it could be internally or using other database management agents. In our case, NTCA can handle the dataset and enable the dataset to retrieve it when needed.

In the NTCA operation, the traffic dataset is collected using Wireshark in real-time. About 100,000 entries are collected while surfing the internet for about an hour.

The data is saved in CSV format and used in the feature extraction and feature selection stages. The extracted features are the source and destination IP address, packet size, and protocol. The main goal is to demonstrate the implementation of an NTCA before using it to recompose a MANA-NMS system.

NTCA, is a building block in the MANA-NMS framework as depicted in Figure 6.7. The NTCA collects the incoming network traffic and classifies them into three different classes. Similar to the NTPA implementation, we assumed high priority (HP), for uMTC service types, medium priority (MP) for mMTC service types, and low priority (LP) for xMBB service types. After the network traffic is classified, it is sent to an event handler (task manager/dispatcher). The event handler then decides where to schedule the traffic relaying based on the scheduling policy. A FIFO policy is used in case of traffic belonging to the same classes. Then a task is scheduled onto the appropriate service agent to be executed.

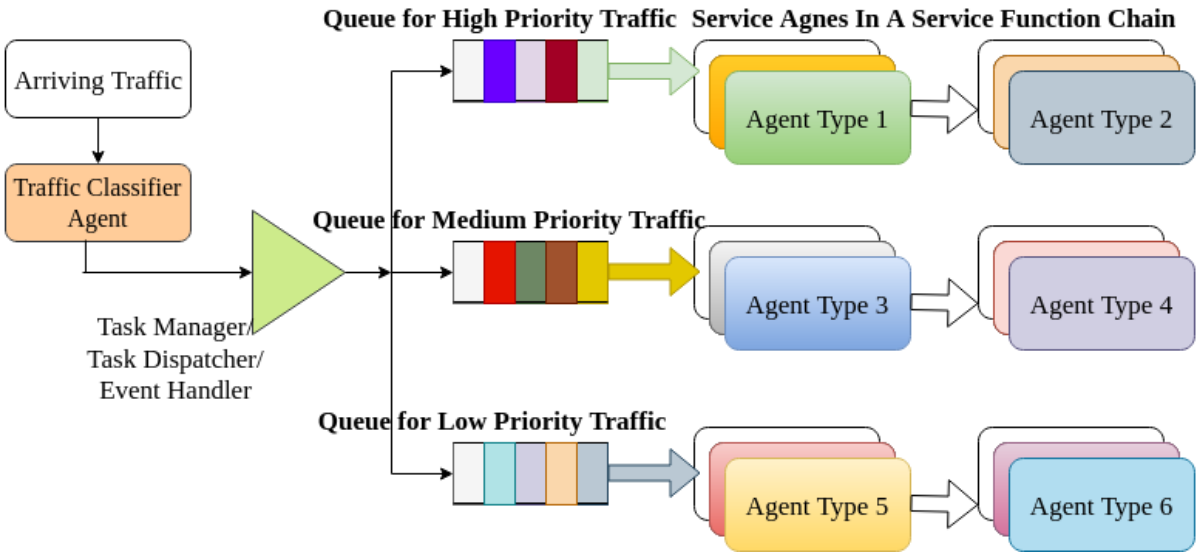


Figure 6.7: Implemented MANA-NMS Scenario.

Job/Task/event Handler and Queuing Scheduling

A task (event) manager agent is another type of agent in a given MANA-NMS architecture. It receives the classified network traffic from NTCA. We assumed every class of the network traffic as a task requiring execution in the service processing agent. The task manager decides which service processing agent should handle the service. This decision is performed using a scheduling policy or intelligent scheduling agent.

Scheduling Policy

"Join Class Related Queue" is used for scheduling. Scheduling is a policy that sends tasks to the queues of the appropriate service processing agent. The agent should be subscribed to that particular task execution. For instance, when an HP class job is sent by the event manager, it will be placed in the queue of a particular service agent that is allocated to process an HP task.

Queuing Model Employed

We assumed an M/M/1 Markovian model for the queue. The queuing policy is assumed to be FIFO similar to the NTPA scheduling policy. In general, M/M/1 means that the system has a Poisson arrival process with arrival rate λ, an exponential service time distribution with service time μ, and one server. Jobs arrive at the server's queue at independent exponentially distributed time intervals with mean $1/\lambda$, they join the queue, and once in the head of the queue, they get served by the server whose service time per job is independent and exponentially distributed with parameter $1/\mu$. A task will be scheduled into the appropriate queue when the task manager/event handler releases it. We assumed three different queues that are associated with a particular service agent chain which indicates the three classes of services to be scheduled in the appropriate agent chain.

6.4 Performance Evaluation

This section presents the performance evaluation results of NTPA and NTCA equipped with different DL and ML models, respectively. We used metrics such as prediction and classification accuracy, training and predicting latency, and memory and CPU utilization to evaluate the performance. Therefore, at the end of this section we will able to identify appropriate DL/ML-based NTPA and NTCA for different prediction/classification network applications with respect to their time and resource constraints.

Table 6.3 shows the DL models parameters used in the experiments.

6.4.1 Accuracy of NTPA using Different DL Models

We used sliding window technique to train our DL models and cross validation to avoid the overfitting with validation split of 0.1 as mentioned in Table 6.3 along with other training parameters. The validation split determines the fraction of training data used

for validation. The validation data is not used for training but evaluates the loss at the of each epoch.

Figure 6.8 illustrates the performance of the different Deep learning models. Where x-axis is day and y-axis is traffic(Tera bit per second (Tbps)). In our experiment we predict traffic over different days with different DL models (GRU, CNN, MLP and LSTM) and take the average of traffics and compare it with real traffic. Figure 8 shows the LSTM, GRU, MLP, and CNN based NTPA prediction results against the actual network traffic. As shown in the figure, there is no significant difference between their prediction results while MLP predicts the network traffic differently. Overall, LSTM outperforms other models.

To measure the accuracy for different models we used two methods: Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). RMSE is a measure that evaluates how good or bad is the prediction using Euclidean distance by calculating the difference between the true value and prediction. Also, MAPE is a different method that measures the prediction accuracy as a ratio. RMSE and MAPE are expressed as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \parallel p(i) - \hat{p}(i) \parallel^2}{N}} \tag{6.6}$$

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{V_i - P_i}{V_i} \right| \tag{6.7}$$

Where in Equation 6.6, N is the number of data points, $p(i)$ is the measured true value, and $\hat{p}(i)$ is the predicted value. In Equation 6.7, n is the number of fitted points, V_i is the actual value, and P_i is the predicted value.

Figures 6.9 and 6.10 demonstrate the accuracy for DL models with respect to different dataset size. We have calculated the average for one week, one month, three months, six months, and one year for the MSE and MAPE errors and compared different models. As shown in the figures, LSTM is the most accurate model to predict the network traffic.

Table 6.2. Configuration Parameters

Parameter Name	Deep Learning Models			
	LSTM	GRU	CNN	MLP
Learning Rate	0.1	0.3	0.2	0.01
Validation-split	0.1	0.1	0.1	0.1
Epochs	20	20	20	20
Dropout	0.2	0.1	0.2	0.2
Momentum	0.9	0.8	0.7	0.9
Hidden-layer	1	1	1	1

Additionally, as the training dataset size grows, the error value decreases. In other words, training the NTPA with larger datasets provides higher prediction accuracy. The RMSE error values for 1 year long dataset as shown in Figure 6.9, for LSTM, GRU, CNN, and MLP are 0.03, 0.06, 0.2, and 5.7, respectively. Moreover, the MAPE error values for 1 year long dataset as shown in Figure 6.10, for LSTM, GRU, CNN, and MLP are 0.24, 0.44, 0.77, and 7.17, respectively. Both figures indicate LSTM has the most accurate prediction performance.

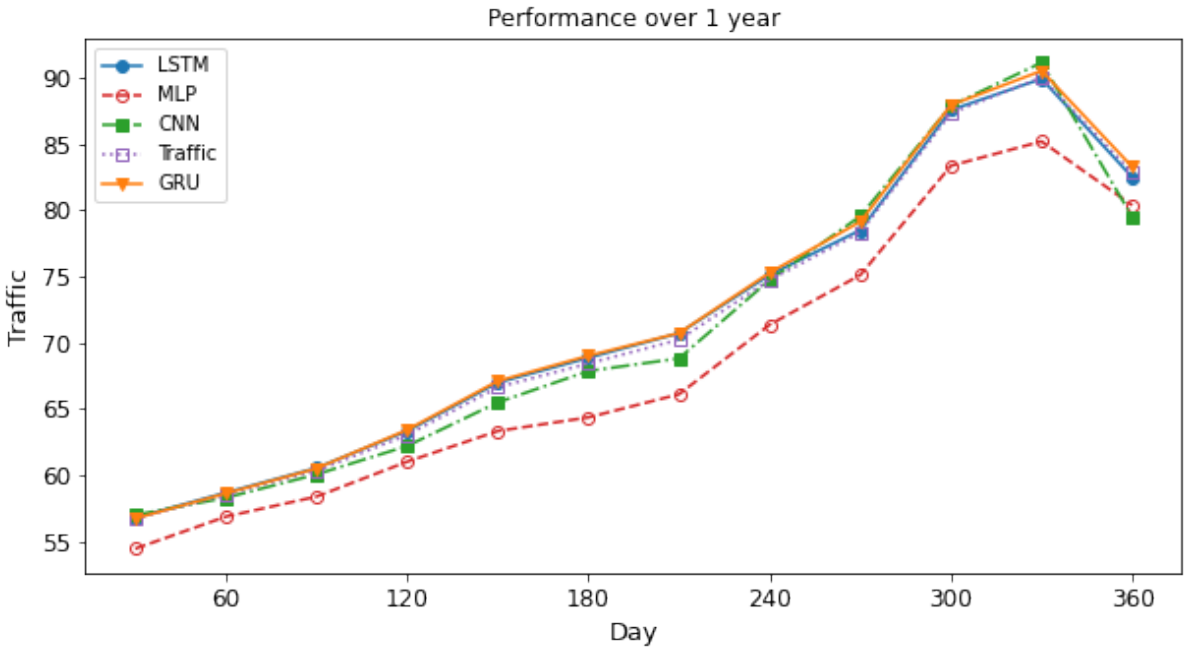


Figure 6.8: Different DL Models Performance Comparison.

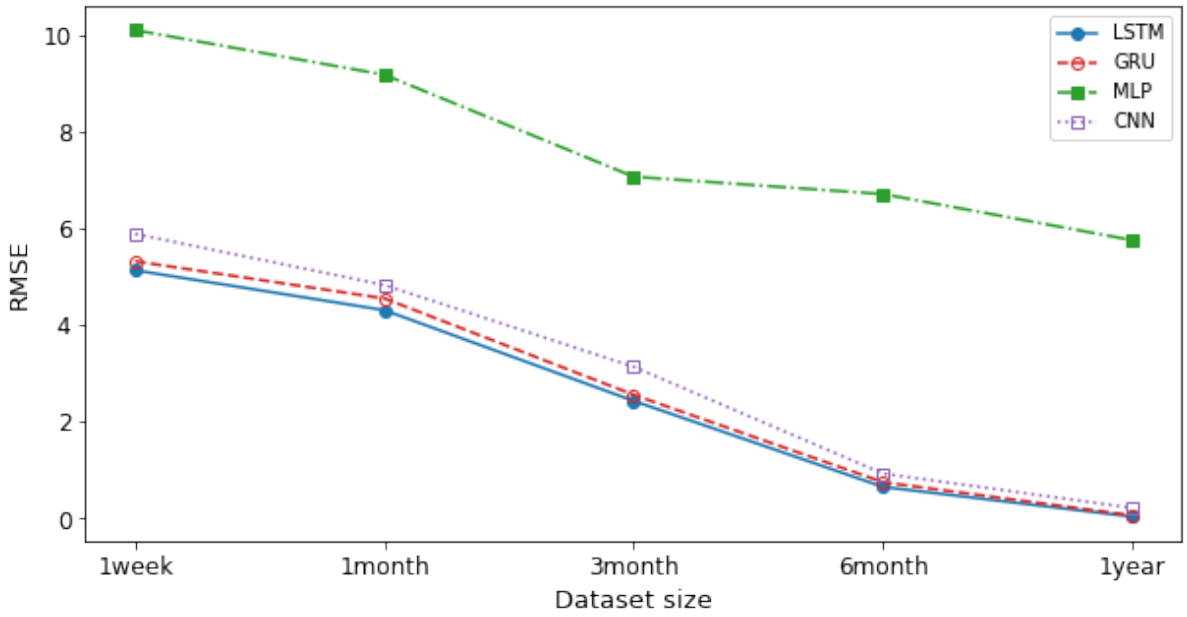


Figure 6.9: RMSE for Different Dataset Size.

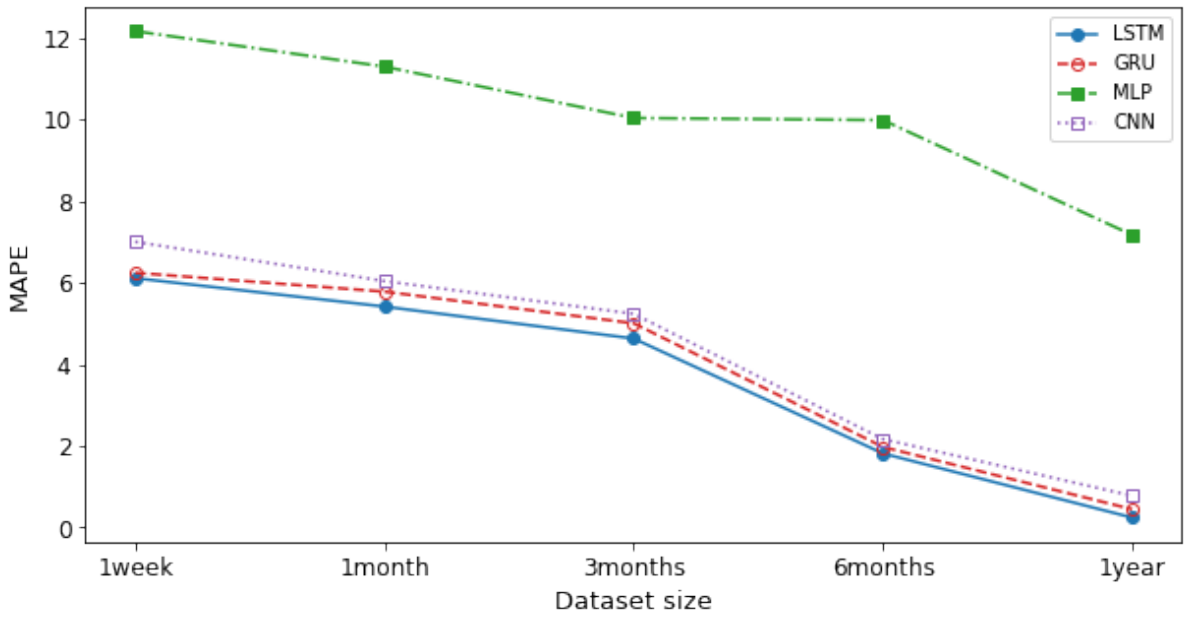


Figure 6.10: MAPE for Different Dataset Size.

6.4.2 Training Latency and Predicting Latency of NTPA using Different DL Models

Figure 6.11 shows the comparison of training latency for the LSTM, GRU, CNN, MLP models. Training latency is represented as the amount of time taken in training an NTPA. The training latency was measured over different dataset size. According to the figure, we see CNN has the lowest training latency compared to other models. However, GRU has the highest training latency. Average training latency values for 1 year long dataset over 20 training trials for the CNN, MLP, LSTM, and GRU models are 74.036 s, 120.873 s, 269.647 s, 291.274 s, respectively. In addition to the training latency measure, we assessed the predicting latency. Predicting latency is defined as the amount of time taken to predict the traffic at a previously unknown time slot. As shown in Figure 6.12, the predicting latency has been evaluated over a simulation period of one hour. On average, the LSTM, MLP, CNN, and GRU NTPA designs have predicting latency values of 0.769 μs, 0.834 μs, 1.025 μs, 1.523 μs, respectively.

The result shows NTPA takes longer to be trained, however, it takes shorter amount

of time to predict compared to NTCA. This confirms the advantage of the DL methods over ML methods, where a DL model is much quicker in testing while the an ML model's testing time will increase as the dataset size grows as mentioned in section 6.1.4.

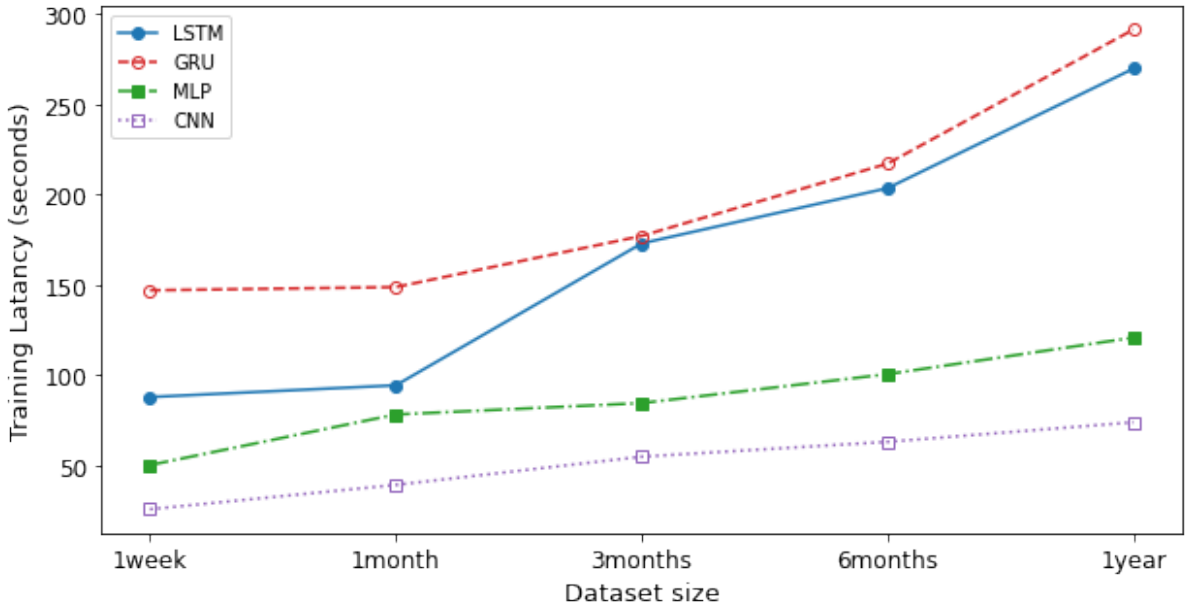


Figure 6.11: Comparison of NTPA's Training Latency.

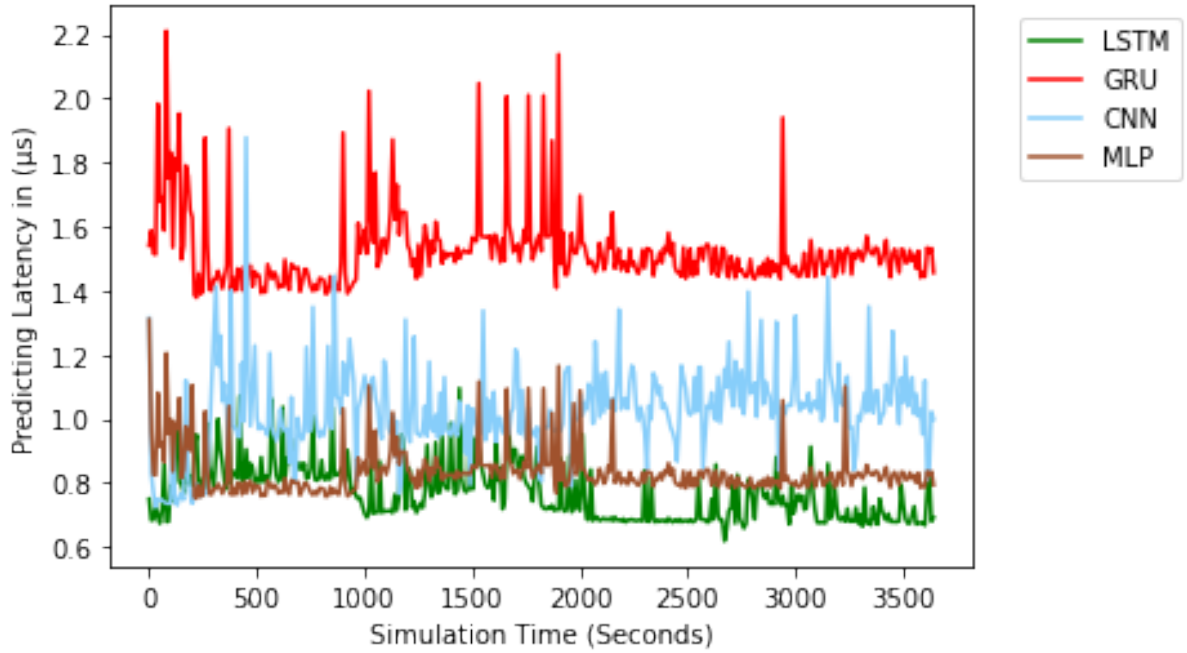


Figure 6.12: Comparison of NTPA's Predicting Latency.

6.4.3 Computing Resource Utilization of NTPA Using Different DL Models

Memory Usage

We examined the memory usage (in MB) of DL models. As the result is shown in Figure 6.13, We observe GRU-based NTPA has the highest and CNN-based NTPA has the lowest memory usage compared to other models.

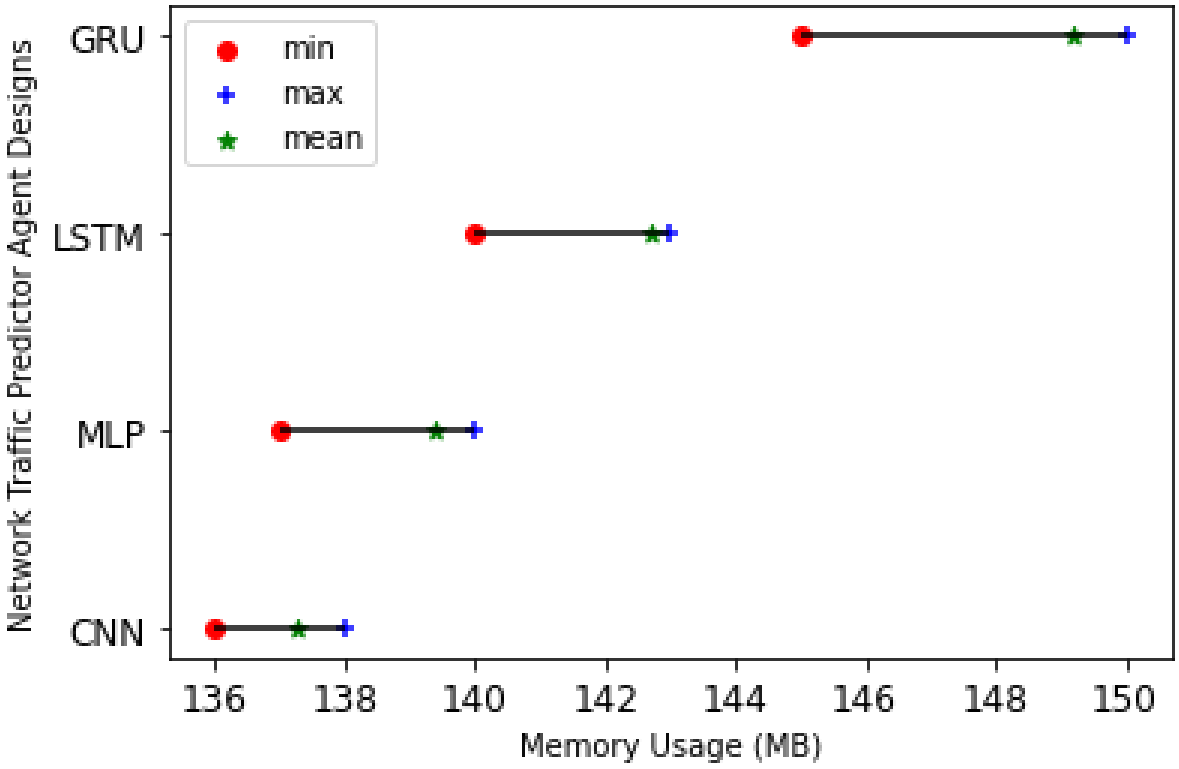


Figure 6.13: Comparison of NTPA's Memory Usage.

CPU Usage

We also examined the CPU usage of DL models. The result is shown in Figure 6.14. We observe that LSTM has the lowest CPU usage compared to other models. Additionally, as seen in the figure, the CPU usage of GRU is the highest.

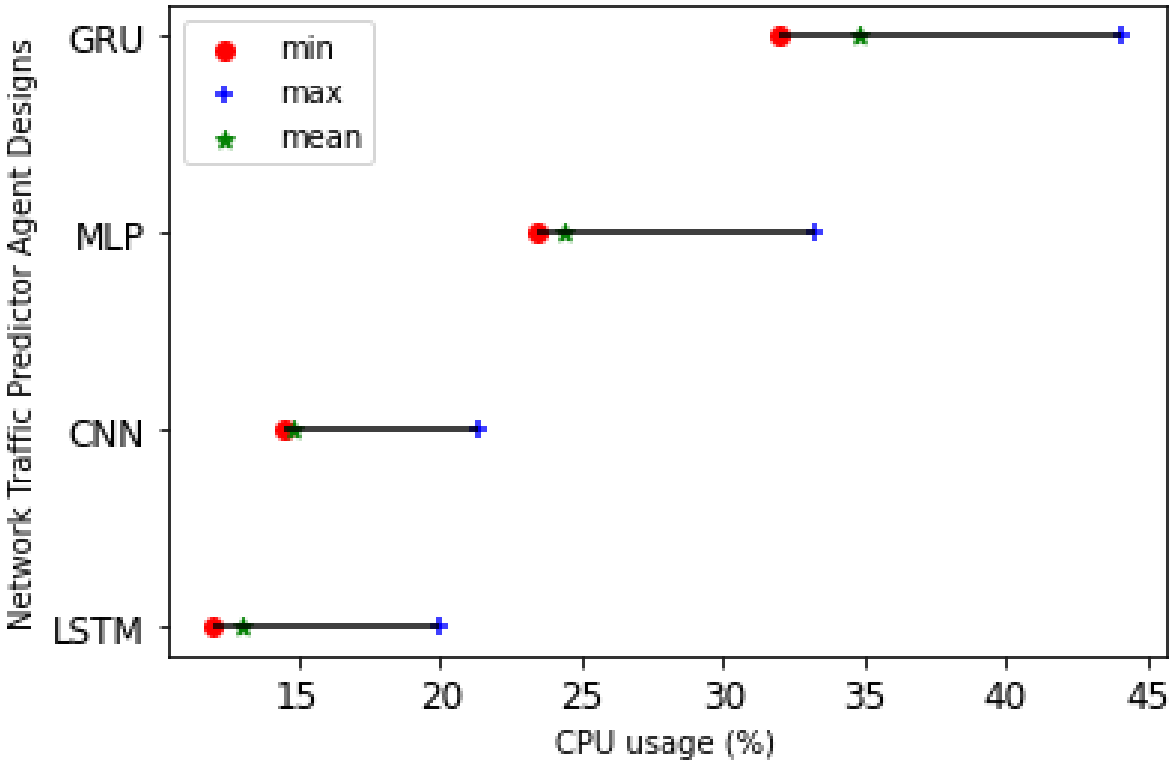


Figure 6.14: Comparison of NTPA's CPU Usage.

6.4.4 Accuracy of NTCA using Different ML Models

We have evaluated our ML algorithm for the NTCA designs. We split the collected data into a training (80%) and test (20%) set. The classification accuracy is considered by various factors such as the dataset size and the number of features.

For the sack of generating a generic result, that is consistent across time and multiple trials in a given environment for a given model, we have run the NTCA algorithms several times, running our models 10 times. We calculated the percentage of error made by the classifier agent for a number of classification trials. We evaluated the mean and variance in terms of accuracy for different NTCA using various ML models. The result is presented in Table 6.3. According to the table, we can understand DT-based NTCA is the most robust and accurate of all the classifier agents. This is because it has the smallest variance while the algorithm is run multiple times.

Table 6.3. NTCA's Classification Accuracy

k_{th} run of algorithm	Classifier Agent Designs			
	K-NN (%)	Decision Tree (%)	SVM (%)	Naive Bayes (%)
1	98.95	99.6	94.77	88.01
2	98.87	99.44	94.45	88.01
3	98.79	99.52	94.53	89.14
4	98.79	99.6	94.53	87.61
5	98.63	99.44	95.17	88.42
6	98.63	99.52	95.09	88.09
7	98.63	99.36	94.21	87.77
8	98.87	99.68	94.85	88.9
9	98.87	99.44	94.93	87.93
10	98.95	99.44	94.93	88.74
Mean	98.798	99.504	94.746	88.262
Variance	0.01628	0.00967	0.09394	0.26282

6.4.5 Classification and Training Latency of NTCA using Different ML Models

Training latency refers to the delay incurred in training a given ML model.

The mean training latency of 154.3, 154.9, 155.1, and 162.2 ms, are measured for the DT, K-NN, NB, and SVM based NTCA's, respectively. It is worth mentioning that the DT-based NTCA design exhibits the lowest average training latency with 154.3 ms, as shown in Figure 6.15. We observe that NB, DT, and K-NN models train faster than the SVM-based NTCA model.

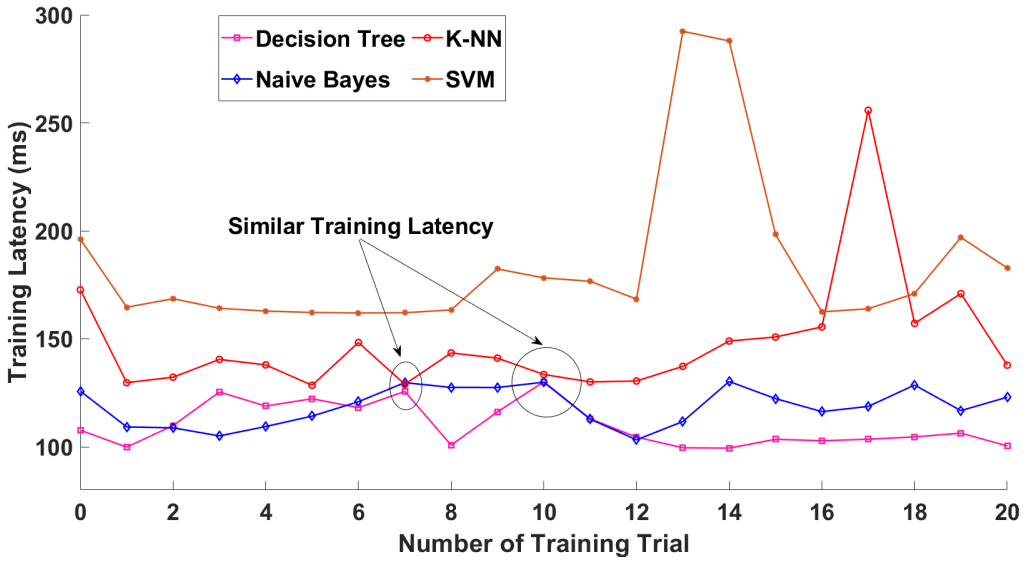


Figure 6.15: Comparison of NTCA's Training Latency

Classification latency is how long it takes to classify new data. In 6.16, we compared NTCA's classification accuracy simulating over a one-hour simulation period.

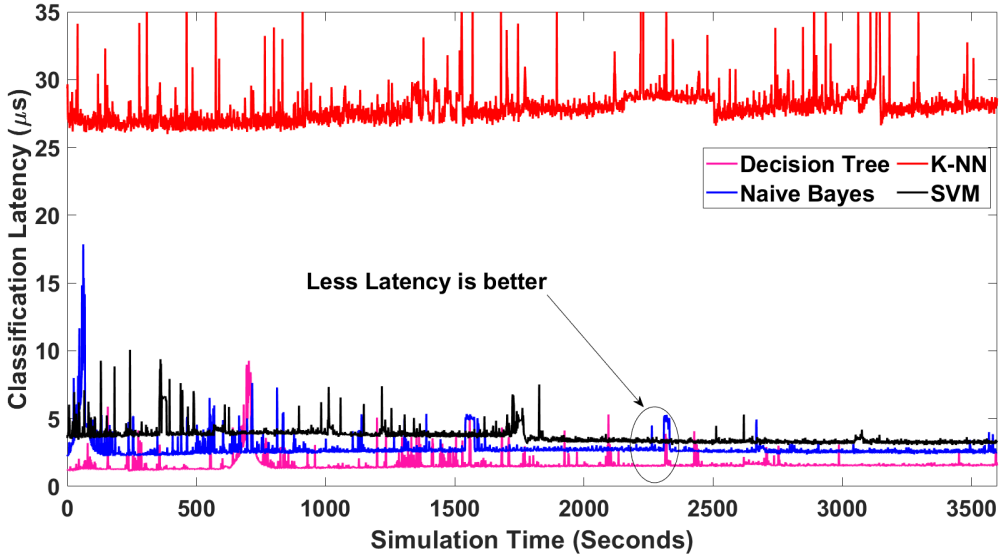


Figure 6.16: Comparison of NTCA's Classification Latency.

The average latency of K-NN, SVM, NB, and DT-based NTCA are 28.01 μs, 3.668 μs, 2.773 μs, and 1.593 μs, respectively. Relatively, SVM, NB, and DT NTCA impose the least mean latency which is less than 4 μs. And, the DT-based NTCA has the least average latency which is 1.593 μs.

6.4.6 Computing Resource Utilization of NTCA using Different ML Models

Memory Usage

Figure 6.17 shows that the NB traffic classifier agent design uses the least average memory of 141.1 megabytes (MB) to train the model and classify new data instances. Despite the fact that the DT-based NTCA design offered the best classification accuracy of 99.504%, it uses the most memory resources amounting to 150MB. This gives the system designer a chance to make a trade-off between classification accuracy and memory usage while trying to choose a traffic classifier agent design.

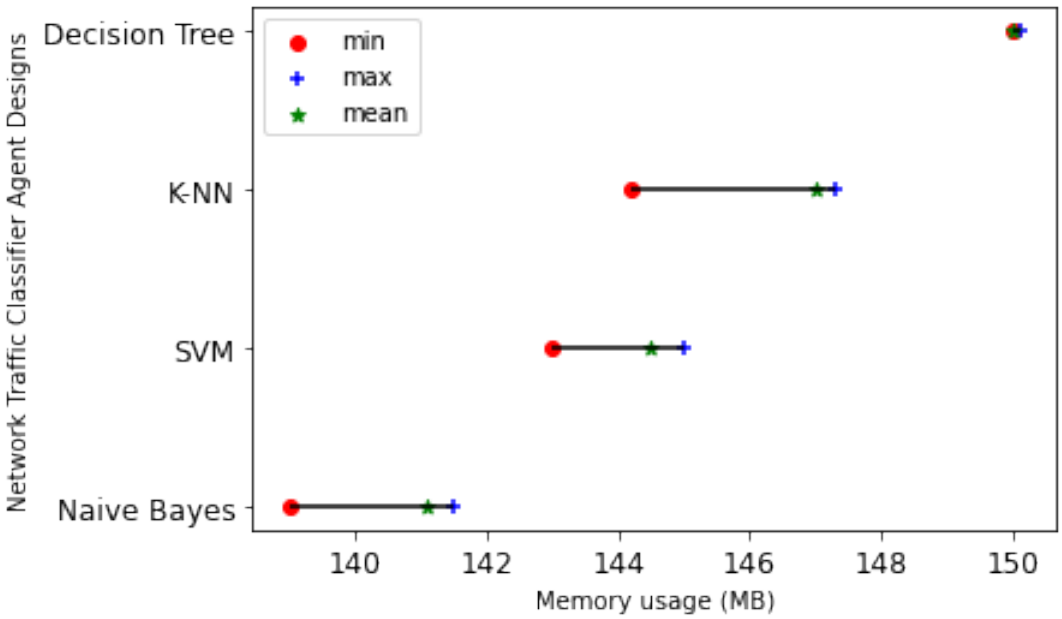


Figure 6.17: Comparison of NTCA's Memory Usage.

CPU usage

In Figure 6.18 represents the statistical differences of each traffic classifier performance including minimum, mean, and maximum CPU usage. The SVM traffic classifier has the least maximum CPU utilization of 31.2% while the DT-based NTCA has the largest maximum CPU utilization of 79.2%.

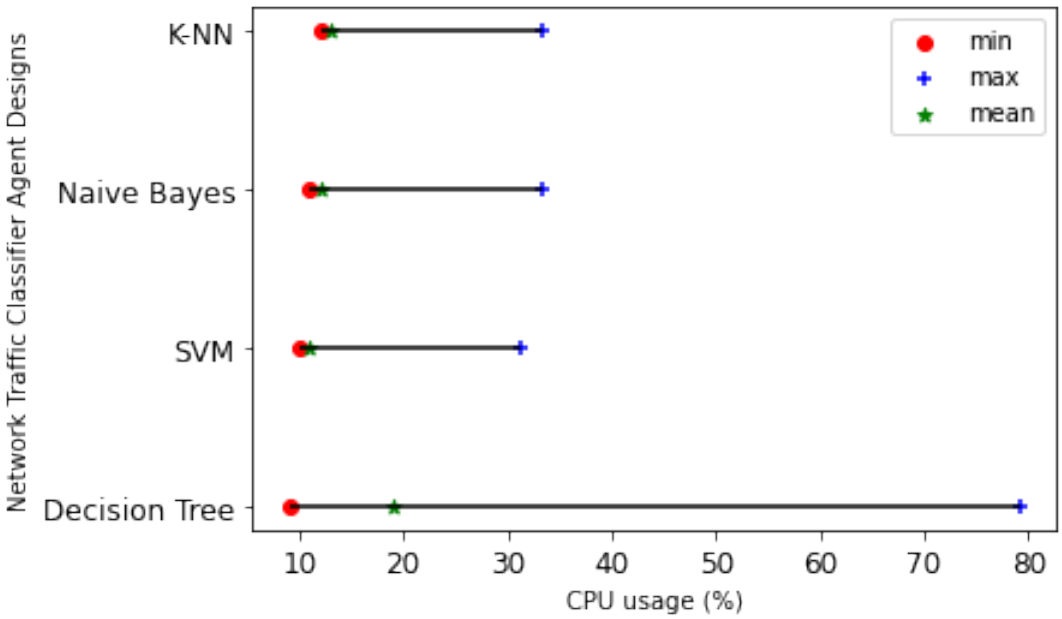


Figure 6.18: Comparison of NTCA's CPU Usage.

6.5 Conclusion and Future Work

Traffic classification and prediction are essential for network resource management, QoS provisioning and monitoring, security, and network failure detection. For example, proactive-based network management, resource allocation, and traffic scheduling can be performed using the prediction result. An organized introduction of intelligence in the network is required to have automated network control and management. Using multi-agent-based service-orientation, a MANA-NMS architecture was presented in our previous work.

In this work, we have designed and implemented traffic classifier and predictor agents. Both NTCA and NTPA leverage ML/DL as an internal cognitive component of the agents to use in their decision process. The agents are autonomous in performing their respective decision. Using different cognition elements in the design and implementation, the performance of both agents is evaluated in terms of classification and prediction accuracy, classification and prediction latency, training latency, and computation resource utiliza-

tion. The result suggests that LSTM based NTPA and DT-based NTCA performs better than the rest in terms of prediction and classification accuracy and latency, respectively.

As a future work a realistic testbed implementation is required. Moreover, a complete system using the autonomous and atomic agents as a building block is necessary. However, we have implemented only NTPA and NTCA in this work. Therefore, it is necessary to have other services, such as routing, QoS monitoring, 5G Core Access and Mobility Management Function (AMF), Session Management Function (SMF), Firewall, Deep Packet Inspection, and other novel functions to have a complete MANA-NMS model for network systems such as 5G wireless network, autonomic network management, etc. Thus, it is an open challenge to implement such functions as intelligent agents. Furthermore, the agent is expected to be deployed in a container or virtual environment. Experimental evaluation of such deployment is also another open challenge that we will be working on.

Acknowledgment

This work has been partially funded by NATO Science for Peace and Security (SPS) Programme in the framework of the project SPS G5428 "Dynamic Architecture based on UAVs Monitoring for Border Security and Safety", and partially by the US National Science Foundation under the New Mexico SMART Grid Center - EPSCoR cooperative agreement Grant OIA-1757207.

Chapter 7

DRL for Online Latency Aware Workload Offloading in MEC

The growing number of Internet of Things (IoT) devices, such as smart phones and smart watches, generate huge amount of data and tasks. Normally, some of these IoT devices are resource-constrained and do not have the capacity to process the tasks locally. The mobile cloud computing (MCC) technology has been proposed to allow these devices to offload their tasks to a remote data center. However, transmitting the tasks from IoT devices to a remote data center via the Internet is expensive, leading to high and uncontrollable latency [148], thus unable to meet many IoT applications' requirements. For example, augmented reality requires the network delay to be less than 20 ms, which cannot be satisfied by MCC [9].

To reduce the network latency, mobile edge computing (MEC) has been proposed to deploy many MEC servers at the network edge. Hence, instead of offloading tasks to a remote data center, IoT devices can offload their tasks to the nearby MEC servers offering low network delay, thus potentially reducing the response time. However, computing resources of MEC servers are limited, thus offloading a task to the nearest MEC server may not always be optimal because , while it minimizes the network delay to offload the task it may incur high computing latency of executing the task. Many studies have designed methods to decide whether to offload tasks at the IoT devices [151,175]. However, the focus of these papers is the offloading decision problem, i.e., if an IoT device has already determined to offload a task and transmitted the task to the base station (BS) , then which MEC server should be selected to execute the task. Note that offloading decision and resource management are coupled together, meaning that whether an MEC server is suitable to execute a task depends on how much computing resource in the MEC server is allocated to the task, which is determined by the amount of remaining com-

puting resource of the MEC server and the priority of the task. That is, if the task has low priority, i.e., low latency requirement, it is not necessary to assign all the remaining computing resource of the MEC server to the task.

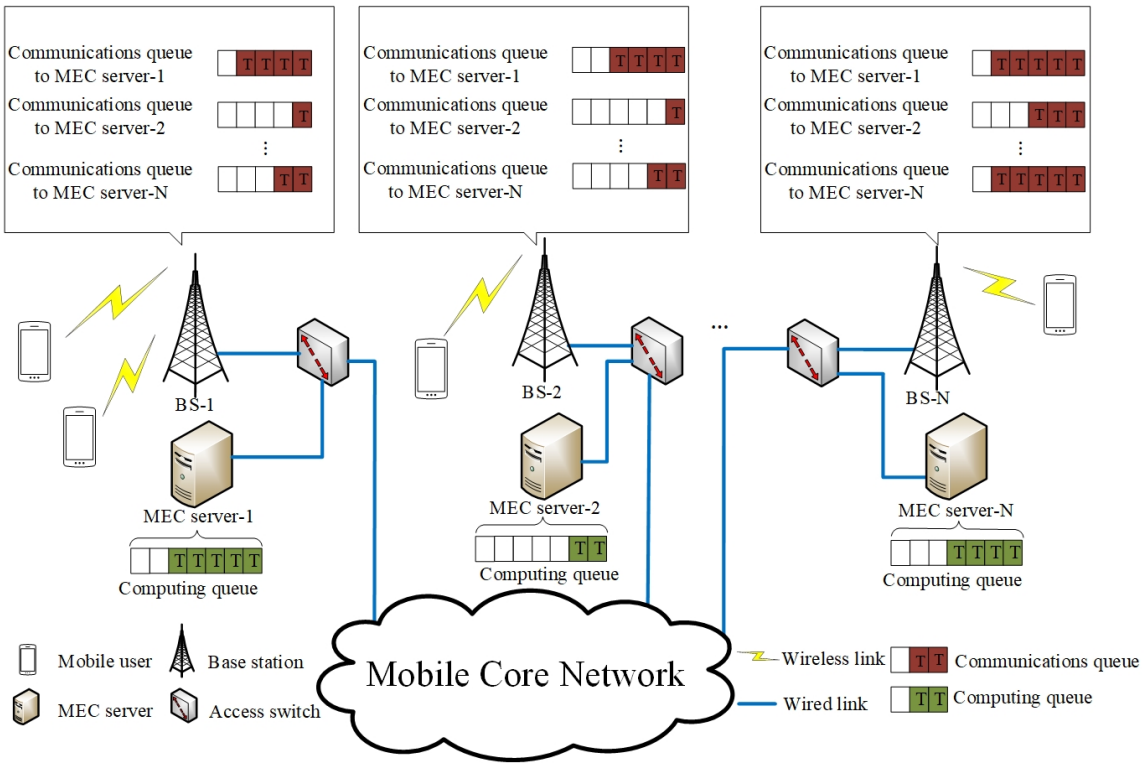


Figure 7.1: The MEC architecture.

To solve the joint offloading decision and resource management problem, machine learning and non-machine learning based solutions have been developed. Non-machine learning based solutions suggest a centralized controller to solve the optimization problem and determine the offloading and resource allocation of the incoming tasks at the BS [58, 149, 150]. However, these solutions have the following drawbacks: 1) they only minimize the latency of the current IoT tasks by optimizing the offloading decision and resource allocation and do not consider the performance of the future IoT tasks, which may lead to the insufficient computing resources for the future IoT tasks at an MEC server, thus increasing their response time, and 2) they cannot make real-time decisions, i.e., the offloading decision and resource allocation cannot be made upon the arrival of a task. The existing machine learning based solutions employ deep reinforcement learning (DRL)

to minimize the expected cumulative response time of all the tasks, which can resolve the second drawback of the non-machine learning based solutions [17, 73]. However, these solutions simplify the system by ignoring the waiting time of the tasks in the queues as well as the priorities of the tasks. Specifically, Fig. 7.1 shows the architecture of MEC, where each base station (BS) is attached to an MEC server via an access switch and maintains a number of communication queues, each of which buffers the arriving IoT tasks. Each MEC server executes the offloaded tasks and has a computing queue holding the tasks that are waiting for the computing resources to be released by the running tasks. The waiting time of an IoT task in the communication and computing queues would significantly affect the offloading decision and resource allocation, but yet to be considered in the existing solutions.

In this paper [13], we apply the advantage actor critic (A2C) method to solve the mentioned problem. Each BS observes the states of the system and determines the actions including the destination MEC server ID and the amount of the computing resources allocated to a task upon its arrival at the BS. The major contributions of the paper are listed as follows:

- We formulate the joint offloading decision and resource management problem by considering different priorities of tasks and the waiting time of the tasks in the communication and computing queues. We model this optimization problem as Markov Decision Process (MDP).

- We propose the Deep reinforcement lEarning based offloading deCision and rEsource managemeNT (DECENT) algorithm to solve the problem based on A2C.

- We demonstrate that DECENT outperforms the other two baseline approaches via extensive simulations.

The rest of this paper is organized as follows. Section 7.1 presents related work. Section 7.2 illustrates the system model to estimate the response time of offloading a task

and presents the problem formulation of joint offloading decision and resource management. Section 7.3 provides the detail of the DECENT algorithm. Section 7.4 discusses the simulation results, and Section 7.5 concludes the work.

7.1 Related Work

Many works focus on the strategy to determine a task should be offloaded to the nearby MEC server or executed locally such that the response time or the energy consumption of executing the task can be minimized [27,95,106]. For example, Elgazzar *et al.* [54] proposed a decision model to evaluate whether offloading a task to the nearby MEC server improves its performance or not. The system operates by selecting a suitable resource provider to perform a task based on contextual information. Sun and Ansari [149] proposed a solution to place private virtual machines (VMs) with fixed computing resources for different mobile users to optimize the tradeoff between the migration gain and the migration cost. Assigning static computing resources to different VMs may lead to low resource utilization and increase the response time. By classifying the tasks into different IoT applications, Fan *et al.* [58] converted the task offloading problem into the application VM allocation problem. They proposed a method to dynamically adjust computing resources of different applications in each MEC server based on their workloads, thus reducing the computing delay of all tasks in the MEC server. Sun and Ansari [150] proposed a Latency aware workload offloading algorithm to optimize the offloading decision such that the average response time of the tasks can be minimized. These two papers require a centralized server to obtain the tasks from different BSs and solve an optimization problem. Also, the waiting time of the tasks in the communications queue and the priorities of the tasks are not considered in these papers. Badnava *et al.* [27] employed a Deep Q-Network to choose the best communication channel for task offloading to maximize the lifetime of a swarm of unmanned aerial vehicles (UAVs). Jia *et al.* [75] aimed to balance the workload among different geo-distributed MEC servers such

that the computing latency could be minimized. Yet, the network delay of transmitting the workload among the MEC servers is ignored. Alfakih *et al.* [17], [26] applied the reinforcement learning approach to optimize the offloading decision and bandwidth allocation to minimize the system cost, which comprises energy consumption of a mobile device and computing delay of a task. However, the paper does not dynamically allocate computing resources to different tasks. Huang *et al.* [73] proposed performance-aware resource allocation to efficiently assign computing and communication resources to users. The objective of the work is to maximize the long term performance of the system by using deep deterministic policy gradient (DDPG) to achieve the best resource allocation.

7.2 System Model

Fig. 7.1 shows the architecture of the MEC. Each BS is attached to an MEC server and communicates with the IoT devices, which are trying to offload their tasks in the coverage area of the BS. Each BS maintains a number of communications queues, each of which holds the tasks waiting to be transmitted to the desired MEC server, and each MEC server maintains a computing queue that holds the tasks waiting for sufficient computing resources to be released on the MEC server. Each BS determines where to offload a task and how much computing resources is allocated to the task upon its arrival such that the average response time of the task is minimized. Let $\mathcal{I}$ and $\mathcal{K}$ be the sets of the tasks and the MEC servers in the system. Let i and k be the indices of the tasks and MEC servers, respectively. Note that each BS is attached to an MEC server and we will use the same index to represent a BS and its attached MEC server. In general, the response time of offloading task i to MEC sever k, denoted as T_{ik}, comprises the network delay T_{ik}^{net} and computing delay T_{ik}^{comp}, i.e., $T_{ik} = T_{ik}^{net} + T_{ik}^{comp}$. The network delay T_{ik}^{net} is the elapsed time from the arrival of task i at the BS until its delivery to an MEC server k. Furthermore, the computing delay T_{ik}^{comp} is the length of time from the arrival of task i at MEC server k to its completion. Without loss of generality, we do not consider the

delay of sending the result of a task back to the IoT device. [24]

Network Delay: The network delay of offloading task i to MEC server k comprises: 1) the transmission time T_{ik}^{trans} of task i from the BS to MEC server k via the network, i.e., $T_{ik}^{trans} = \frac{l_i}{w_k}$, where l_i is the size of task i in bits and w_k is the capacity of the path from the BS to MEC server k in bps. 2) the E2E delay between the BS and MEC server k, denoted as T_{ik}^{e2e}, and 3) the waiting time of task i in the communication queue from the BS to MEC server k, denoted as $T_{ik}^{wait_net}$. Here, the E2E delay T_{ik}^{e2e} can be measured and monitored by the network controller. For instance, the software defined networking (SDN) technology can be applied to the mobile core network and thus the SDN controller can periodically monitor and record the E2E delay between any two endpoints [152,172]. Also, the waiting time $T_{ik}^{wait_net}$ is the time duration between the arrival of task i at the BS and its transmission start time to MEC server k, which is approximately equal to the sum of the transmission time of all the tasks in the communications queue, i.e.,

$$T_{ik}^{wait_net} = \sum_{i' \in \mathcal{I}'_{ik}} \frac{l_{i'}}{w_k} \tag{7.1}$$

where $\mathcal{I}'_{ik}$ is the set of tasks in the communications queue for MEC server k when task i arrives at the BS. Hence, the network delay of offloading task i to MEC server k is

$$T_{ik}^{net} = T_{ik}^{trans} + T_{ik}^{e2e} + T_{ik}^{wait_net} = \frac{l_i}{w_k} + T_{ik}^{e2e} + \sum_{i' \in \mathcal{I}'_{ik}} \frac{l_{i'}}{w_k}. \tag{7.2}$$

Computing Delay: The computing delay of offloading task i to MEC server k comprises 1) the waiting time $T_{ik}^{wait_comp}$ of task i to be executed in the computing queue of MEC server k, and 2) the execution time $T_{ik}^{exe_comp}$ of task i at MEC server k. Normally, $T_{ik}^{exe_comp}$ depends on the complexity of task i (i.e., how many CPU cycles are required) and the amount of computing resources allocated to task i in MEC server k, denoted as r_{ik}. The complexity of task i is usually proportional to the size of task i [59], and thus we have $T_{ik}^{exe_comp} = \frac{\mu l_i}{r_{ik}}$, where μ is the coefficient to map the size of a task in bits to the

complexity of the task in CPU cycles. In addition, the waiting time $T_{ik}^{wait_comp}$ depends on the number of tasks and their complexities in the computing queue when task i arrives at MEC server k, i.e., $T_{ik}^{wait_comp} = \sum_{i'' \in \mathcal{I}''_{ik}} \frac{\mu l_{i''}}{r_{i''k}}$, where $\mathcal{I}''_{ik}$ is the set of tasks in the computing queue of MEC server k at the arrival of task i. Hence, the computing delay of task i to MEC server k is

$$T_{ik}^{comp} = T_{ik}^{exe_comp} + T_{ik}^{wait_comp} = \frac{\mu l_i}{r_{ik}} + \sum_{i'' \in \mathcal{I}''_{ik}} \frac{\mu l_{i''}}{r_{i''k}}. \tag{7.3}$$

We formulate the joint offloading decision and resource management problem as follows.

$$\textbf{P0:} \quad \arg\min_{r_{ik}, x_{ik}} \sum_{i \in \mathcal{I}} \eta_i \left(\sum_{k \in \mathcal{K}} x_{ik} \left(T_{ik}^{comp}(r_{ik}) + T_{ik}^{net} \right) \right), \tag{7.4}$$

$$s.t. \ \forall i \in \mathcal{I}, \sum_{k \in \mathcal{K}} x_{ik} = 1, \tag{7.5}$$

$$\forall i \in \mathcal{I}, \forall k \in \mathcal{K}, x_{ik} \in \{0, 1\}, \tag{7.6}$$

$$\forall i \in \mathcal{I}, \forall k \in \mathcal{K}, r_{ik} = \{0, R_1, R_2, R_3, \cdots R^{\max}\}, \tag{7.7}$$

where x_{ik} is a binary variable indicating whether task i is offloaded to MEC server k ($x_{ik} = 1$) or not ($x_{ik} = 0$) and η_i is the priority or weight of task i. A larger η_i indicates the system prefers to reduce the response time of the task and vice versa. The objective of **P0** is to minimize the overall weighted response time of all the tasks, where $\sum_{k \in \mathcal{K}} x_{ik} \left(T_{ik}^{comp} + T_{ik}^{net} \right)$ is the response time of task i. Constraint (7.5) indicates any task can only be offloaded to a specific MEC server. Constraint (7.6) means that x_{ik} is a binary variable. Constraint (7.7) defines the feasible values of r_{ik}, where $\{0, R_1, R_2, \cdots\}$ are the different computing resource blocks (e.g., the number of CPU cores) that can be allocated to task i in MEC server k and $R^{\max}$ is the maximum capacity of an MEC server.

P0 is nontrivial to be solved because 1) different tasks arrive at different time slots, and thus the BS cannot make the immediate decision for a task to minimize the delay if it is not aware of the future incoming tasks' information, and 2) **P0** is an NP hard problem even if the BS can predict the information of all the tasks (i.e., their arrival time, complexities, and weights). Hence, we propose to apply a DRL method to find the sub-optimal solution of **P0** in real-time.

7.3 Deep reinforcement learning based offloading decision and resource management

We apply MDP $(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{R}})$ to reformulate **P0**: 1) $\boldsymbol{\mathcal{S}}$ indicates the state space. A state at the arrival of task i, denoted as $\boldsymbol{s}_i \in \boldsymbol{\mathcal{S}}$, includes

- Weight and data size of a new task, i.e., η_i and l_i.
- Remaining computing resource of the MEC servers once new task i arrives, i.e., $\boldsymbol{C} = \{c_k \,|\forall k \in \boldsymbol{\mathcal{K}}\}$, where c_k is the remaining computing resource of MEC server k. Here, $c_k = c^{\max} - \sum_{j \in \boldsymbol{\mathcal{I}}_{ik}^{comp}} r_{jk}$, where $\boldsymbol{\mathcal{I}}_{ik}^{comp}$ is the set of tasks executed by MEC server k once task i arrives.
- Computing workload of MEC servers' computing queues once new task i arrives, i.e., $\boldsymbol{D} = \{d_k \,|\forall k \in \boldsymbol{\mathcal{K}}\}$, where d_k is the computing workload of MEC server k's computing queue, i.e., $d_k = \sum_{j \in \boldsymbol{\mathcal{I}}_{ik}^{comp_queue}} z_j$. $\boldsymbol{\mathcal{I}}_{ik}^{comp_queue}$ is the set of tasks in MEC server k's computing queue once new task i arrives.
- Waiting time of the communication queues once task i arrives, i.e., $\boldsymbol{B} = \{b_k \,|\forall k \in \boldsymbol{\mathcal{K}}\}$, where b_k is the waiting time of the communication queue for MEC server k, i.e., $b_k = \sum_{j \in \boldsymbol{\mathcal{I}}_{ik}^{comm_queue}} \frac{l_j}{w_k}$. $\boldsymbol{\mathcal{I}}_{ik}^{comm_queue}$ is the set of tasks in the communications queue once task i arrives.

2) $\boldsymbol{\mathcal{A}}$ is the set of actions for a BS to offload a task. The action set of task i, denoted as $\boldsymbol{a}_i \in \boldsymbol{\mathcal{A}}$, comprises $\boldsymbol{a}_i = \{r_{ik}, x_{ik}\}$; 3) $\boldsymbol{\mathcal{F}} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \rightarrow \boldsymbol{\mathcal{S}}$ defines the state transition

probability density function that maps the current states and actions into the next states; 4) $\boldsymbol{\mathcal{R}} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \rightarrow \mathbb{R}$ is the reward function. The reward function for task i can be defined as the negative value of task i's response time, i.e., $r_i = -\eta_i \left(\sum_{k \in \mathcal{K}} x_{ik} \left(T_{ik}^{comp} + T_{ik}^{net} \right) \right)$.

We then design the DECENT algorithm, which is based on Advantage Actor Critic (A2C) [80], to solve the MDP problem. A2C is a DRL method combining policy-based and value-based reinforcement learning. In A2C, there are two neural networks, i.e., the actor and critic networks. The actor network provides the stochastic policy $\pi_\theta(\boldsymbol{a}_i|\boldsymbol{s}_i)$ to choose the actions $\boldsymbol{a}_i$ such that the expected cumulative reward, denoted as $J(\theta)$, is maximized. Here,

$$J(\theta) = \mathbb{E}\left[\sum_{i'=i}^{|\mathcal{I}|} \gamma^{i'} r_{i'} \right], \tag{7.8}$$

where θ is the parameter of the actor network, $\gamma \in [0, 1]$ is the discount factor, and $|\mathcal{I}|$ is the total number of the tasks. According to [171], the gradient of $J(\theta)$ is:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(\boldsymbol{a}_i|\boldsymbol{s}_i) A(\boldsymbol{s}_i, \boldsymbol{a}_i)], \tag{7.9}$$

where $A(\boldsymbol{s}_i, \boldsymbol{a}_i)$ is the Advantage function defined as

$$A(\boldsymbol{s}_i, \boldsymbol{a}_i) = r_i + \gamma V_\upsilon(\boldsymbol{s}_{i+1}) - V_\upsilon(\boldsymbol{s}_i). \tag{7.10}$$

Here, $V_\upsilon(\boldsymbol{s}_i)$ and $V_\upsilon(\boldsymbol{s}_{i+1})$ are the state-values with respect to task i and $i+1$ estimated by the critic network and υ is the parameter of the critic network. Hence, the actor's parameter θ is updated by the gradient descend, i.e.,

$$\theta := \theta - \beta_a \nabla_\theta J(\theta), \tag{7.11}$$

where β_a is the learning rate of the actor network.

The critic network in A2C is used to evaluate the actions taken by the actor network

and provides the advantage value to the actor network to improve the policy. The objective of the critic network is to minimize the loss function $J(\upsilon)$, which is defined as the mean square error between the estimated state-value and the expected cumulative reward, i.e.,

$$J\left(\upsilon\right) = \left(r_i + \gamma V_\upsilon(\boldsymbol{s}_{i+1}) - V_\upsilon(\boldsymbol{s}_i)\right)^2 \tag{7.12}$$

Denote $\nabla_\upsilon J\left(\upsilon\right)$ as the gradient of $J\left(\upsilon\right)$ with respect to the parameter υ. Then, υ is updated based on

$$\upsilon := \upsilon - \beta_c \nabla_\upsilon J\left(\upsilon\right), \tag{7.13}$$

where β_c is the learning rate of the critic network.

The structures of the actor and critic networks are as follows. The actor network comprises an input layer taking in the input state $\boldsymbol{s}_i = \{\eta_i, l_i, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{B}\}$, a hidden layer with 128 neurons and a relu activation function, and an output layer generating a probability distribution over actions with a softmax activation function. Likewise, critic network has an input layer taking in the state and action pair $\langle \boldsymbol{s}_i, \boldsymbol{a}_i \rangle$, a hidden layer with 128 neurons and a relu activation function, and an output layer generating the state value.

Algorithm 1 summarizes the DECENT algorithm, which is the process of training the actor and critic networks. Specifically, upon an arrival of task i at the BS, the actor network applies the current policy to generate the action $\boldsymbol{a}_i = \{r_{ik}, x_{ik}\}$ based on the current state $\boldsymbol{s}_i$. Note that we apply the ϵ-greedy policy where it selects the random actions with the probability of 10% and the greedy actions (that maximize the expected cumulative reward) with the probability of 90%. Based on the actions $\boldsymbol{a}_i$, we calculate the corresponding reward r_i. This process is repeated until the actor and critic networks are converged. The well-trained actor and critic networks are used to determine the actions of incoming tasks in real-time.

Algorithm 1: DECENT algorithm

Initialize discount factor γ, learning rates β_a and β_c, and exploration rate ϵ.
for *each training episode* **do**
  for *each task arrival* **do**
    Obtain the current state $\boldsymbol{s}_i$;
    Input $\boldsymbol{s}_i$ to the actor network to calculate the actions $\boldsymbol{a}_i = [x_{ik}, r_{ik}]$ using ϵ-greedy;
    Calculate reward r_i;
    Store transition (s_i, a_i, r_i, s_{i+1}) in the replay buffer;
  end
  Sample transitions from the replay buffer;
  Update the actor neural network based on Eq. (7.11);
  Update the critic neural network based on Eq. (7.13);
end

7.4 Simulation Results

In this section, we will conduct extensive simulations to validate the performance of the DECENT algorithm. Assume that there are one BS and 4 MEC servers located in different distances from the BS in km. The distances d_k from the BS to 4 MEC servers are 0, 1, 2, and 3 km, respectively, where the distance is used to calculate the E2E delay T_{ik}^{e2e} between the BS and an MEC server, i.e., $T_{ik}^{e2e} = \alpha \times d_k + \zeta$. Here, α and ζ are the coefficients, which are initially set to be 0.03 s/km and 0.03 s, respectively. The computing capacity of all the MEC servers are the same, i.e., $R^{max} = 200$ Mcycles/s. The capacity of the links from the BS to the MEC servers are the same, i.e., $w_k = 2 \times 10^9$ bits/s. The arrival of tasks follows a Poisson distribution with the average arrival rate equaling to 50 tasks. The data size of an arrival task l_i is randomly generated from a normal distribution, i.e., $l_i \sim N(3 \times 10^7\ bits, 3 \times 10^5)$. The computation intensity of a task $\mu =$ 0.15 CPU cycles/bit. In addition, the number of computing resource blocks that can be assigned to an incoming task $r_{ik} \in \{10, 20, 40, 60, 80, 100, 120, 140, 160, 200\}$ Mcycles/s, and the weight of an arrival task η_i is uniformly selected, i.e., $\eta_i \sim U\{10, 20, 50, 100\}$, where a higher weight of a task implies the task has to be executed in a lower delay, and vice versa. Table 7.1 shows other simulation parameters.

The two baseline algorithms, i.e., nearest server and largest server, are used to compare the performance with DECENT. Here, the nearest server algorithm selects the closest MEC server (i.e., the lowest E2E delay) from the BS, and the largest server algorithm picks the MEC server with the largest remaining computing resource to offload a task.

Parameter	**Value**
Learning rate of the actor network β_a	0.0001
Learning rate of the critic network β_c	0.0002
Exploration rate ϵ	0.1
Link capacity w_k	2×10^9 bits/s
MEC server capacity R^{max}	200 Mcycle/s

Table 7.1. Simulation Parameters.

Fig. 7.2 illustrates the learning curve for the DECENT algorithm, where DECENT can train the actor and critic networks to generate a better actions to maximize the average weighted reward in terms of minimizing the average weighted response time. The learning curves can quickly be converged around 500 episodes. Fig. 7.3 compares the weighted response time of different tasks for different algorithms where the average task arrival rate λ is 50 tasks/s. DECENT outperforms the baseline algorithms as the weighted response time of all the tasks can be lower than 1 second. The nearest MEC server and largest capacity MEC server algorithms, on the other hand, incurs a longer computing delay and communications delay, thus leading to a longer weighted response time.

We also examine the impact of the average task arrival rate λ on the average weighted response time among 6,400 tasks. As shown in Fig. 7.4, as λ increases, the average weighted response time incurred by DECENT slightly increases, while still maintaining a low level, i.e., < 500 ms. The average weighted response time increment of the other two baseline algorithms over λ is similar but much larger than DECENT, which demonstrates

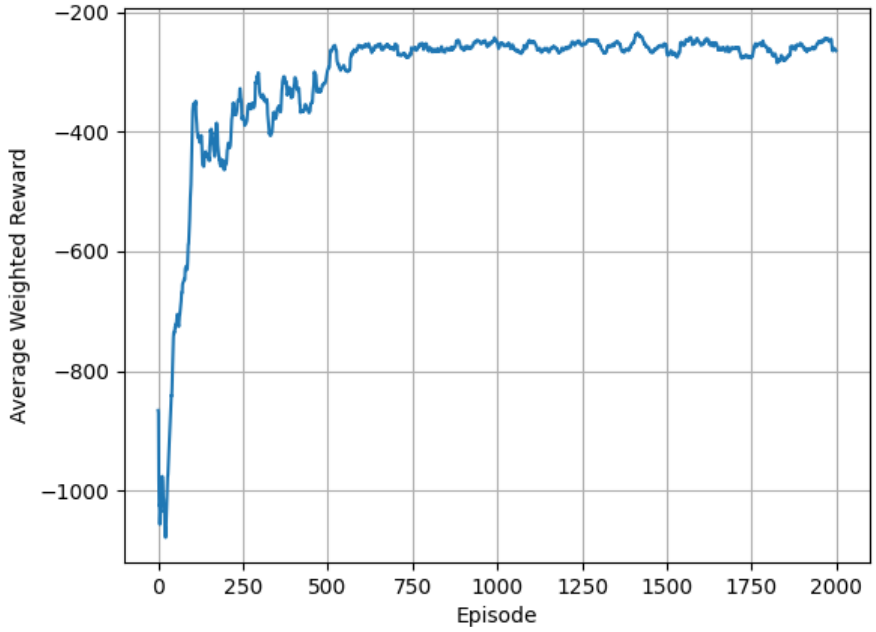


Figure 7.2: Learning curve of DECENT.

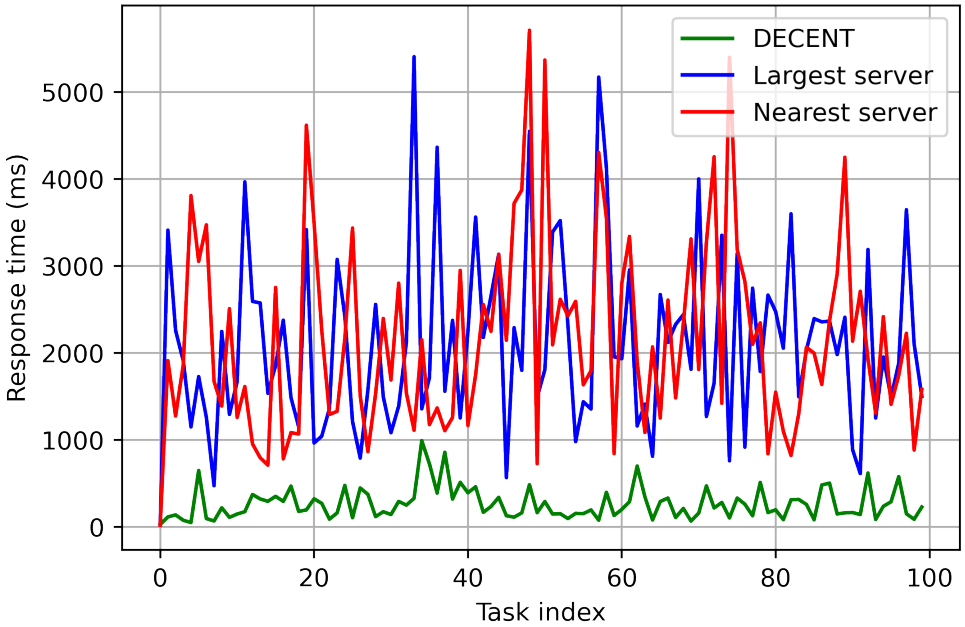


Figure 7.3: Performance comparison.

that DECENT achieves better offloading decision and resource management in both light and heavy workload scenarios. All the tasks can be clustered into four classes, and each class contains the tasks with the same weight. Fig. 7.5 shows the average network and computing delay of the tasks from different classes for DECENT. We can

find that the tasks with higher weight incur lower execution time than the tasks with lower weight, which demonstrates that DECENT can adjust the offloading decision and resource allocation according to the weight of incoming tasks.

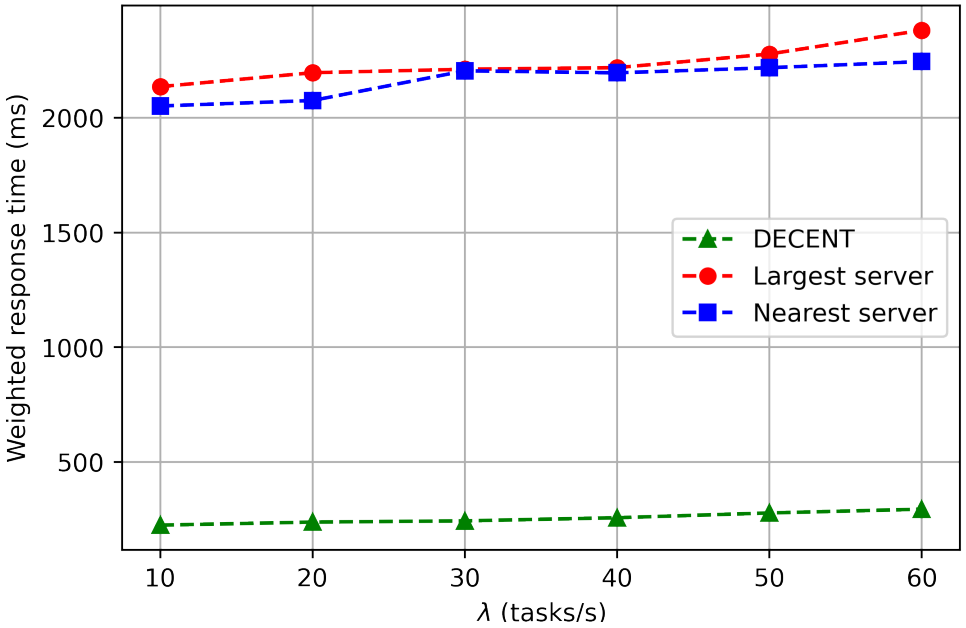


Figure 7.4: Average weighted response time over average task arrival rate for different algorithms.

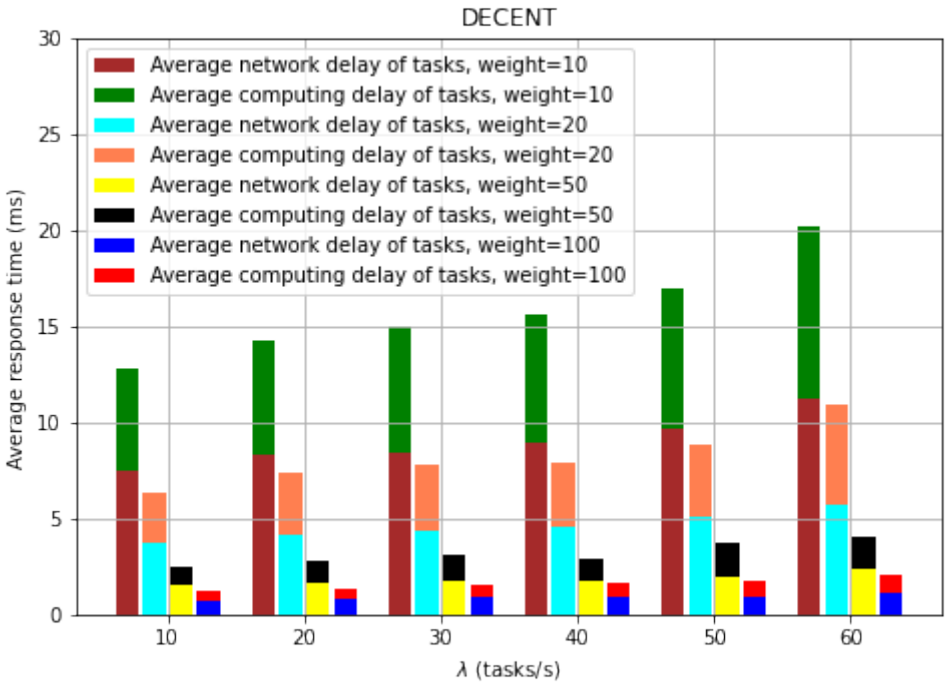


Figure 7.5: Average response time w.r.t λ and weights of the tasks.

Figs. 7.6 and 7.7 show the average weighted response time among 6,400 tasks by varying the computation intensity μ and E2E delay coefficient α, respectively. Note that increasing μ and α would directly increase the E2E delay T_{ik}^{e2e} and the execution time $T_{ik}^{exe_comp}$, respectively. DECENT always incurs the lowest average weighted response time than the two baseline algorithms in different scenarios. It is interesting to see that the performance of the largest server algorithm in Fig. 7.7 is significantly degraded as α increases. This is because increasing α increases T_{ik}^{e2e}, which may dominate the response time, and so offloading tasks to the nearby servers is preferred to reduce the weighted response time. Yet, the largest server algorithm does not consider the network delay, thus leading to significant performance degradation.

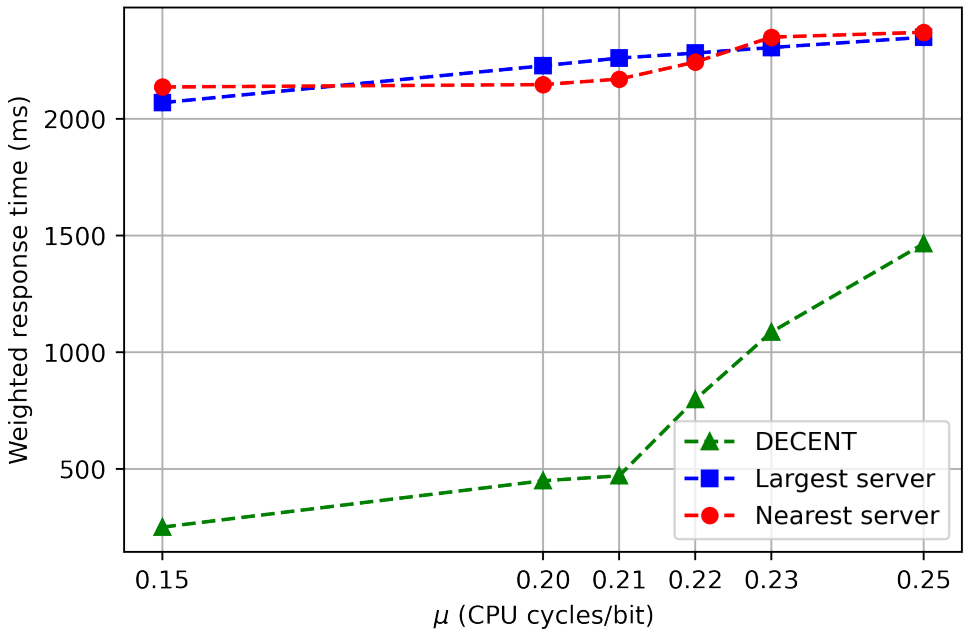


Figure 7.6: Average weighted response time over μ.

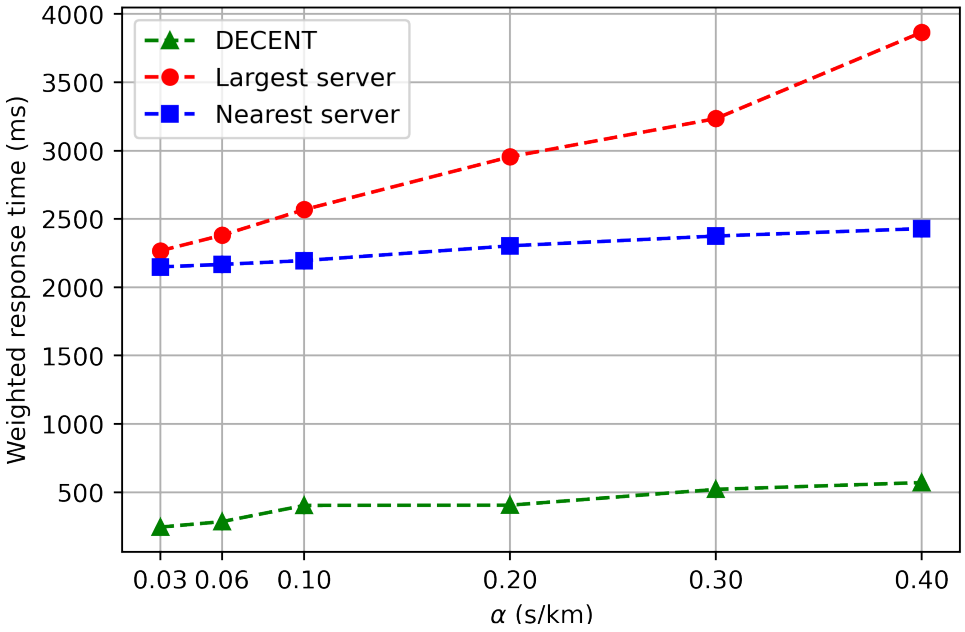


Figure 7.7: Average weighted response time over α.

7.5 Conclusion

By considering the waiting time of a task in the communication and computing queues as well as different priorities of the tasks, this paper propose the DECENT algorithm allowing each BS to determine the offloading decision and computing resource allocation for each arrival task in real-time such that the cumulative weighted response time is minimized. As compared to the two baseline algorithms, DECENT has been demonstrated to have lower response time in different scenarios via extensive simulations. Also, DECENT is capable of adjusting the offloading decision and computing resource allocation based on the weight of the incoming tasks to minimize the cumulative weighted response time.

Chapter 8

Hospital Asset Usage Prediction using Deep Learning

IoT has been used for various applications and is becoming ubiquitous. Ranges of industrial, human, medical, and research applications employ IoT-based systems. Hospital asset management is one of the applications of IoT. In particular, IoT and Deep Learning (DL) can be used for inventory management in hospitals. Managing inventory in a hospital has complexity and high operational costs. This is due to expensive medical equipment being used and stored in multiple storage rooms within the hospital. And there is usage unpredictability of the medical devices [44]. In general, the healthcare sector has been facing various challenges in effectively managing these supplies for a long time. Lacking supplies and materials may postpone treatment and endanger patients' health whereas unused items or overstocking of supplies increases costs. Enhancing inventory management in hospitals such as supplies acquisition and movement across different locations while controlling costs requires coordination and integration between different fields such as manufacturing and operations research management to provide analytical methodologies, considering the limited human resources and supplies [104].

The goal of inventory management in hospitals is to assist in keeping the accurate recording of the inventory on hand to lower the out-of-stock rate of medical assets and eliminate overstocking and product expirations. In other words, balancing costs with the right amount of inventory considering complex and last-minute changes happening within the hospitals to ensure customer satisfaction is the biggest challenge that has caused poor inventory and distribution management in the hospital. Adopting forecasting techniques can help optimize stocking levels and thus sustain quality and timely patient care while efficiently utilizing expensive medical equipment.

Numerous strategies for managing inventory have been suggested, such as the two-bin

Kanban system, reorder point/reorder quantity policy, reorder point/order-up-to policy, and periodic and continuous policies, [131]. The Periodic Automatic Replenishment (PAR) level system is frequently used to regulate inventory in aligning stock levels with the demand. PAR Levels refer to the minimum and maximum quantities set for a specific item, where the minimum level indicates the need for reordering when the quantity gets close to it. On the other hand, maximum levels indicate an excessive stock of an item, which may lead to wastage or expiration. A drawback of offline periodic solutions is that they only utilize the current inventory information disregarding historical information. This can lead to overstocking of some equipment or stock-outs. Furthermore, a continuous inventory system allows for continually updating the inventory on hand according to utilization data and capturing details that can improve patient care. For continuous replenishment systems, barcode and Radio Frequency Identification (RFID) tag-based strategies are commonly used due to no energy, inexpensive, and help automate the retrieval of information such as asset location and quantity [40, 90, 169]. This along with IoT and DL-based intelligent, real-time, and accurate decision-making systems is possible to improve productivity, patient safety, and lower costs.

In this paper [56], we design and develop an IoT and DL-based automated inventory management system for hospitals to ensure supply availability for scheduled medical procedures, up-to-date inventory information, and minimal clinical personnel involvement. In medical units, conventional inventory replenishment models establish both the timing of order (i.e., the reorder point) and the quantity to be ordered (i.e., the reorder quantity). These models often rely on experience-based inventory levels instead of data-driven actual inventory levels, resulting in an excess of units while others are in short supply. These approaches determine the reorder point and reorder quantity based on the current state of the inventory (i.e., available stock, used stock, and item's expiration dates). However, employing forecasting approaches helps predict the future inventory demand by considering demand variability and cost of ordering and thus improves inventory levels

and logistics operations costs.

Our system design is based on a real-world hospital case where there are multiple inventories located at different departments within a hospital, holding different types of medical equipment. Each item is equipped with an ID and a BLE-based tag to manage its movement and type. Additionally, it records the time when an item is added to the inventory and when it exits the inventory. The major contributions of the paper are listed as follows:

- We design a framework that automates the asset tracking problem in a hospital environment using DL techniques.
- We combine DL models with Variational Mode Decomposition (VMD) method to accurately predict the medical asset availability in an inventory for different time horizons in the future.
- We evaluate our design through a real-world case study involving 29 clean storage zones (medical supply inventories) within a hospital. We selected the Central Supply Front (EDC) clean storage and focus our experiments on the four most frequently used categories.

The rest of this paper is organized as follows: Section 8.1 presents a literature review. Section 8.2 presents the asset tracking problem. Section 8.3 outlines the system model along with the data preparation process and the prediction algorithms. Section 8.4 discusses the system's performance evaluation and Section 8.5 concludes this work.

8.1 Literature Review

Duan *et al.* [50] proposed a metaheuristic-based simulation optimization framework to minimize the expected outdate rate under a predetermined fill rate constraint for supply chain inventory management of highly perishable products. They employed an age-based

replenishment policy using an old inventory ratio that measures the freshness of the perishable products. They tested their approach for one blood center with three and six hospitals. Their inventory management policy helped reduce the system's expected outdate rate from 19.6% to 1.04% on average such that the demand and the supply are balanced. Rappold *et al.* [126] developed an analytic optimization framework to reduce inventory carrying and materials management costs in hospital environments. Their approach modeled the non-stationary stochastic demand and a stochastic bill of materials in surgical departments. Their results demonstrated great savings for hospital investment in inventory. Zwaida *et al.* [10] applied a deep reinforcement learning-based algorithm to address the drug inventory management problem in a hospital and minimize the drug refilling cost and shortage rate in real-time. Lee *et al.* [89] integrated RFID technique with neural network technology to automate and improve the efficiency of inventory management in hospitals. They deployed their proposed system integrated for tracking the movement and the usage patterns of the infusion pumps at a hospital. Their demand forecasting model for medical assets demonstrated that employing neural network technologies has several benefits such as reducing time spent on searching for medical devices, periodic maintenance, stock-take or locating of the lost devices, and releasing the medical staff from doing the paperwork. The work in [19] presented a comparative study of machine learning and classical statistical methods for demand forecast in a hospital setting. The objective of this study is to understand whether the application of statistical methods suffices to facilitate demand forecasting for an inventory management method in hospitals or whether employing machine learning algorithms can be invaluable. The study indicated that in an inventory management context, machine learning algorithms outperformed statistical approaches in terms of accuracy. Additionally, machine learning results can be influenced by time series characteristics as the results achieved by this work show machine learning has a high accuracy only on smooth and erratic time series. On the other hand, machine learning algorithms have larger execution times than

statistical methods. Therefore, despite promising results from machine learning algorithms [60] to address medical shortages and inventory optimization we need to select the appropriate method with respect to the data characteristics. Nguyen *et al.* [117] reviewed and evaluated the scientific literature on the main benefits, opportunities, and challenges of data analytics-based pharmaceutical supply chains to enhance operational performance. Data analytics has proved to be a promising solution to improve pharmaceutical supply chain management. In particular, predictive techniques using machine learning algorithms provide accurate forecasts, thus contributing to tackling shortages and high inventory levels in a hospital environment. However, a few data resources have been extensively used so far. Additionally, there are various data types that need to be combined to ensure integrity and transparency in inventory management. Therefore, data analytics and artificial intelligence capabilities need to be investigated to address medical equipment shipping, distribution, and dispensing in inventory management as well as providing support in times of crisis such as the pandemic.

Although there have been research works that focus on designing and optimizing inventory and replenishment systems for hospitals, most of them either rely on statistical approaches or only a few utilize DL techniques to address drug shortages in hospitals. As far as our knowledge goes, no research has been conducted to specifically optimize the PAR inventory management policy while incorporating all types of medical devices and drugs available in a hospital inventory using DL approaches.

8.2 Problem Description

We aim to design an online solution that dynamically determines the future availability of medical equipment over a time span W while benefiting healthcare facilities using DL. Given a time series $T = \{t_1, \cdots, t_n\}$, each time step is associated with a number of available medical devices in the inventory. The time series consists of regularly spaced window time w in minutes. Our goal is to provide predictions within the time intervals

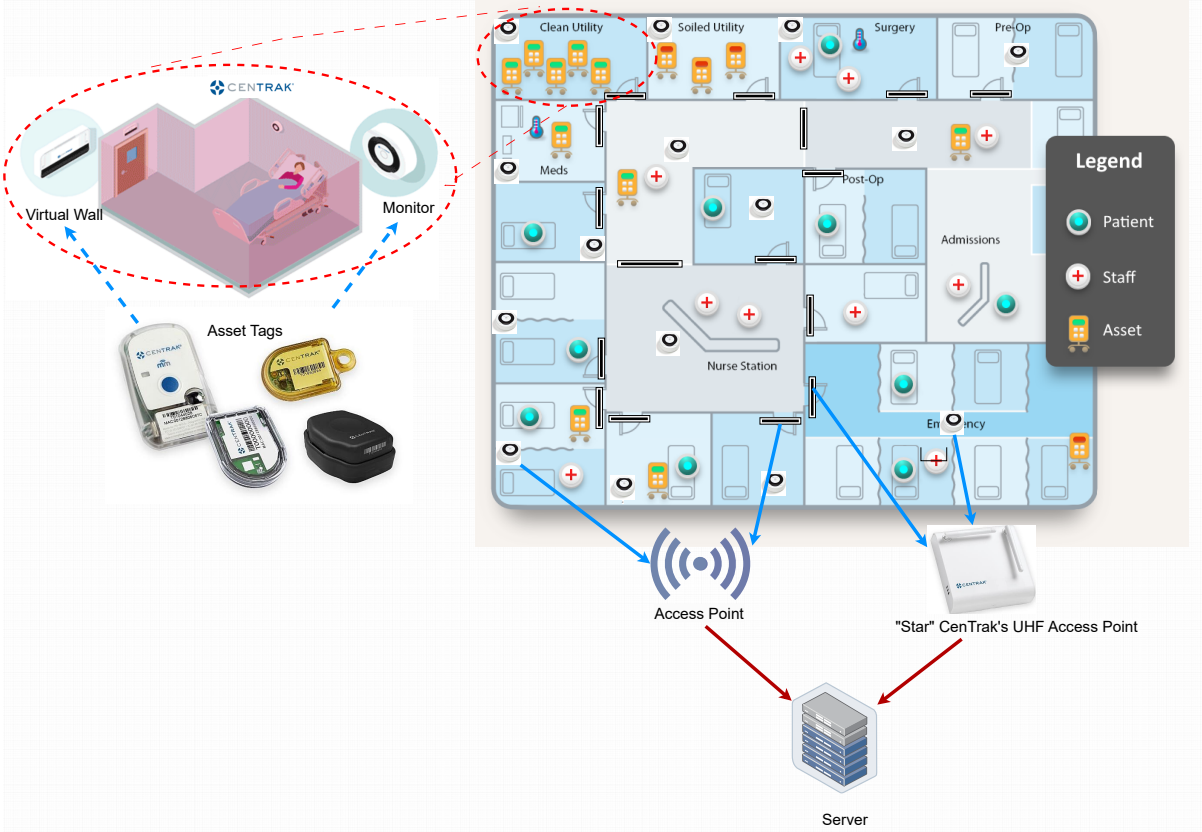


Figure 8.1: IoT Infrastructure [2].

as well as forecasting the minimum and maximum number of devices available for four different time horizons h in the future.

8.3 System Model

8.3.1 Data Modeling-Time Series Analysis

Time series analysis is a fundamental component of data modeling. It involves leveraging insights from past observations to construct precise models capable of capturing the underlying structure of a series. These models, in turn, enable us to make predictions and categorize future events accurately. Researchers have dedicated considerable effort to developing more reliable and precise algorithms and models for time series data analytics. While the field of time series modeling has made remarkable strides, most of these models are designed to work with conventional time series data [162]. However, the challenge lies in the fact that raw data seldom conforms to the ideal scenario of having all input variables sampled at a consistent rate with synchronized timestamps across various variables. This irregularity arises due to the diverse range of sensing devices and recording techniques used to collect data. Transforming uneven data into an even dataset with fixed time intervals plays a pivotal role in enhancing the data's usability, consistency, and interpretability (e.g Fixed time intervals allow you to precisely associate predictions with the corresponding moments in the dataset). This transformation renders the data more suitable for a wide array of analytical tasks, including prediction and statistical analysis. Unevenly spaced time series are usually examined after first converting data into evenly spaced observations through interpolation (generally linear). Yet, doing so can result in various important and difficult-to-quantify biases, particularly if the time between observations is excessively unpredictable. There are currently fewer methods specifically made for analyzing time series data with irregular spacing. To meet strict performance requirements, the least-squares spectrum analysis methods are frequently used. Yet, we will continue to use the re-sampling method due to its simplicity and low computational resource requirements.

8.3.2 IoT Infrastructure-Data Collection and Processing

In this section, we discuss the data collection and processing procedures for our IoT infrastructure in smart hospitals. The dataset we utilized comprises wireless data frames transmitted by CenTrak devices manufactured by CenTrak, as shown in Fig. 8.1 . These devices are strategically placed throughout the hospital environment, including rooms, corridors, and bays, to capture location information.

The CenTrak devices transmit a unique space identifier (e.g., room), which is received by any tag present in that space. The tags then communicate the space ID and their unique ID over existing Wi-Fi networks and/or CenTrak's specialized UHF network. This information is transmitted to a location server in real time, where it is delivered to the software application and other connected healthcare IT solutions.

We have two types of devices in our IoT infrastructure: virtual walls and monitors. Both types emit flashes of IR light that are read by the tags. Monitors are placed centrally on the ceiling of a room or area and flashlights are in all directions. On the other hand, virtual walls are designed with blinders, directing the flashlight downward from the device on one side of the vertical plane. Virtual wall devices are typically positioned near doors or other thresholds, capturing data about a tag's entry/exit without the need for extensive information beyond that.

In the specific context of measuring par levels, which include Clean and Soiled rooms, either type of device can be utilized. Approximately two-thirds of the rooms use monitors, while one-third use virtual walls. Monitors are centrally located on the ceiling, while virtual walls are placed near the door threshold on the interior of the room.

Our dataset was collected from several inventories across different departments within the hospital. It contains valuable information about the inventory, such as the type of medical device, the device ID/tag, the enter timestamp indicating when the device entered the inventory, the exit timestamp representing when the device left the inventory, the zone ID where the inventory is located, and the name of the inventory's location.

However, it is important to note that noise can be introduced during the dataset collection process, resulting in inaccuracies in the proposed approach's performance.

One form of noise we identified in the collected data is the presence of short-time differences between the device's exit and entry timestamps in the inventory (e.g., 0, 1, and 5 seconds). Such occurrences imply that the device has been in use for only a very brief period, which is not logical in a hospital setting. To address this, we established an in-use minimum threshold (th) for all devices and removed all data points that fell within a time range of 0-th minutes. This process helps ensure the reliability of the data.

Next, we computed the number of remaining devices at the enter timestamp in the inventory for each category, taking into account their enter and exit times. This enabled us to construct another dataset with additional information, including the enter timestamp, device category, inventory location, and the number of remaining devices at the enter timestamp.

Furthermore, we did time series analysis as discussed in 8.3 to transform the irregular time intervals (resulting from the unpredictability of entry and exit recording) . To facilitate predictions within fixed-sized windows (e.g., hourly, daily, weekly, monthly, and annually), we transformed the data through resampling. Consequently, we created a new dataset with fixed observation intervals (w), providing a structured framework for measurements and facilitating the learning process.

8.3.3 Prediction Algorithms

We used multiple DL techniques for our forecasting framework such as Long short-term memory (LSTM), Gated recurrent unit (GRU), Convolutional Neural Networks (CNN), and Variational Mode Decomposition (VMD). Next, we briefly discuss them.

VMD

was proposed by Dragomiretskiy and Zosso [48] in 2014. VMD is an entirely non-recursive variational model [87] that concurrently decomposes a time series into K intrinsic mode functions (IMFs). Each IMF has a specific bandwidth in the spectral domain [94] and can be computed by solving the following convex optimization problem:

$$\begin{aligned} \underset{\{u_k\},\{\omega_k\}}{\text{minimize}} \quad & \sum_{k=1}^{K} \left\| \partial_t \left[\left(\delta(t) + \tfrac{j}{\pi t}\right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \\ \text{subject to} \quad & \sum_{k=1}^{K} u_k = f \end{aligned} \tag{8.1}$$

where f is the input time series, t represents the time step, and $\delta(\cdot)$ denotes the Dirac delta function [112].

VMD-LSTM

refers to a combined approach that utilizes both VMD and LSTM techniques for analyzing and processing sequential data. The combination of VMD and LSTM, known as VMD-LSTM, aims to leverage the strengths of both techniques. VMD can be used as a preprocessing step to decompose the input data into its constituent IMFs. Each IMF can then be treated as a separate input sequence, and LSTM models can be trained on these individual components to capture their respective dynamics and patterns. By combining the decomposition capability of VMD and the sequential modeling capability of LSTM, VMD-LSTM can potentially provide improved analysis and prediction performance for complex sequential data.

VMD-GRU

is an integrated approach that combines VMD and GRU techniques to analyze and process sequential data effectively. VMD-GRU takes advantage of the strengths of both

VMD and GRU techniques. VMD is first utilized as a preprocessing step to decompose the input data into its constituent IMFs. Each IMF is then treated as an independent input sequence, allowing GRU models to be trained on these individual components. This approach enables the capture of distinctive dynamics and patterns within each IMF. By combining the decomposition capabilities of VMD with the sequential modeling abilities of GRU, VMD-GRU has the potential to significantly enhance the analysis and prediction performance for complex sequential data.

VMD-CNN

The VMD-CNN framework is a combination of two techniques: VMD and CNN. By combining VMD and CNN, the VMD-CNN framework likely leverages the strengths of both techniques. It may involve using VMD to decompose a time series signal into its constituent modes and then employing a CNN to process and analyze these modes further. CNNs have a good nonlinear mapping capability [34], and theoretically, they have been proven to be universal approximators that can estimate any nonlinear function to any desired accuracy [111], [108]. The CNN could potentially learn discriminative features from the decomposed modes for tasks such as classification, prediction, or anomaly detection.

8.4 Experiment and Numerical Results

The proposed framework is evaluated using our real-world trial-generated dataset of medical equipment-IoT device interaction. First, we describe the combination of DL models with the VMD method and the way it works in our experimental setup. As shown in Fig. 8.2, VMD decomposes the input time series into multiple IMFs. Next, each IMF is given to the CNN model for training. Then, each model generates its predicted signal. Finally, the predicted signals are summed to reconstruct the final prediction.

Table 8.1. Case-Study Parameters

Parameter	Value
Device categories	Rental Bed, Asset Tag, Pulse Oximeter, Supply Cart
Window size (w)	$10, 20, 30, 60$ *minutes*
Range prediction horizon (h)	1 *hour*, 1 *week*, 1 *day*, 1 *month*
Device in-use minimum threshold (th)	5 *minutes*

8.4.1 Case Study

Our case study environment is a hospital with 29 clean storage zones (medical supply inventories), each of which stores various types and quantities of medical equipment. Each clean zone is equipped with a sensing infrastructure of IoT devices that provide real-time location information about tracked medical equipment. The IoT network collects data as each piece of equipment enters or leaves the clean zone. Table 8.1 presents the specific parameter values utilized to assess our framework in this actual scenario.

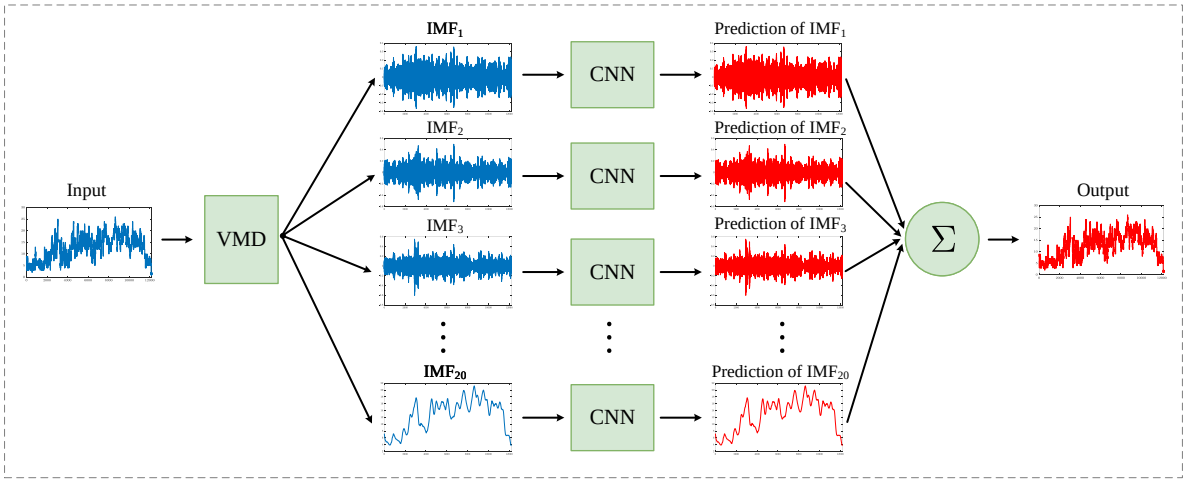


Figure 8.2: Structure of VMD-CNN [112].

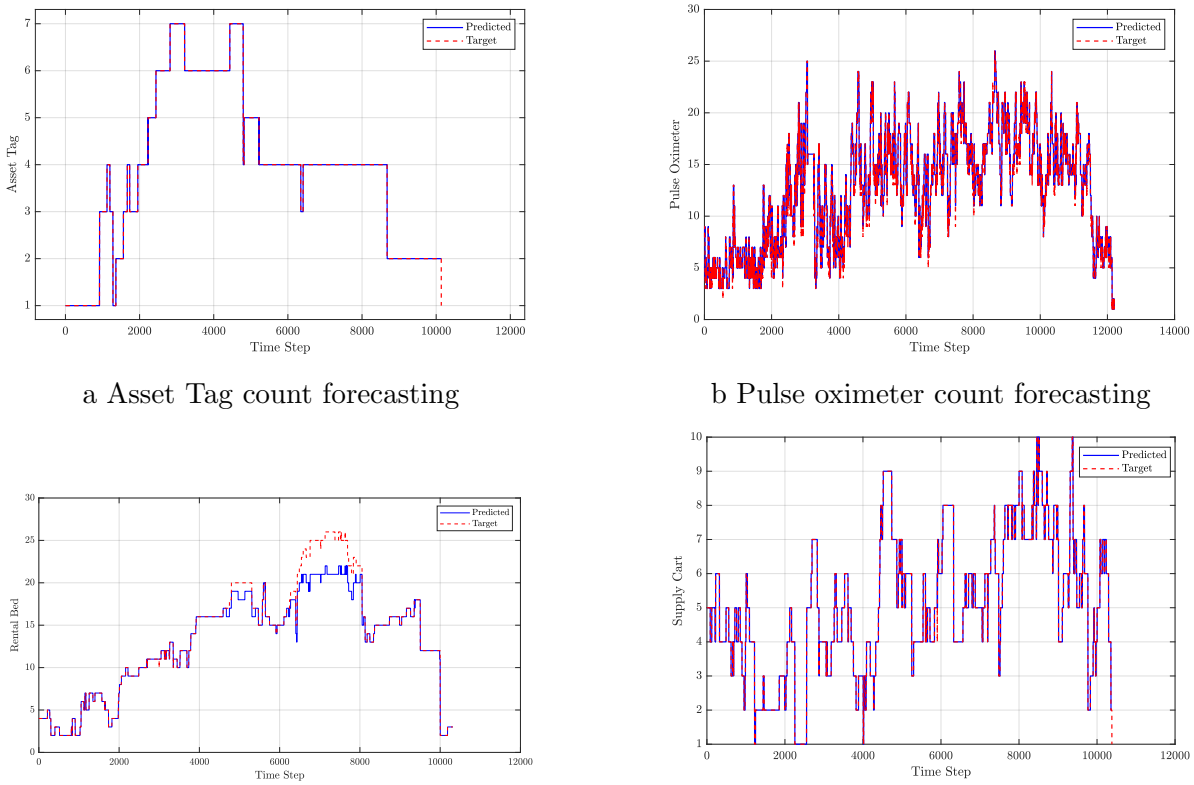
8.4.2 System Evaluation and Overall Accuracy

In this study, the system's ability to predict the number of ready-to-use medical devices in the inventory was evaluated using a real-world-generated dataset. To assess this capability, four types of devices with the highest availability variation were chosen and

Table 8.2. Evaluation of combined DL models with VMD method forecasting performance.

Device	Model	Error Analysis							
		Root Mean Square Deviation (RMSE)				Mean Absolute Percentage Error (MAPE)			
	Window Size	10	20	30	60	10	20	30	60
Asset-Tag	**VMD-CNN**	**0.0180**	**0.1006**	**0.0489**	**0.0537**	**0.2213**	**2.9624**	**1.0367**	**0.5202**
	VMD-GRU	0.2191	0.2587	0.2304	0.2549	4.8672	6.0714	4.7389	0.5283
	VMD-LSTM	0.2587	0.3618	0.3598	0.3950	6.0714	7.5021	7.2833	7.3841
Supply-Cart	**VMD-CNN**	**0.0562**	**0.0915**	**0.0903**	**0.1269**	**0.6753**	**1.5888**	**0.8037**	**1.1456**
	VMD-GRU	0.4914	0.4735	0.4699	0.5726	13.8513	13.0267	12.6215	15.1739
	VMD-LSTM	0.7976	0.7753	0.7592	0.8524	22.4840	21.5400	20.6915	22.6279
Pulse-Oximeter	**VMD-CNN**	**0.2213**	**0.3097**	**0.3692**	**0.4856**	**1.1862**	**1.7820**	**2.2640**	**2.8395**
	VMD-GRU	0.6937	0.8395	0.9618	1.1896	6.7061	7.5971	8.3315	10.8461
	VMD-LSTM	1.2650	1.3414	1.4560	1.7724	13.9395	14.2070	15.0476	18.1823
Rental-Bed	VMD-CNN	3.3939	3.5612	3.6968	1.5346	7.5626	7.8923	8.4118	4.2624
	VMD-GRU	**0.1863**	**0.2507**	**0.3034**	**0.4111**	**1.5511**	**1.8789**	**2.3796**	**3.5270**
	VMD-LSTM	0.6416	0.6996	0.7390	0.8006	9.7438	10.0180	10.2710	10.7511

analyzed using VMD-LSTM, VMD-GRU, and VMD-CNN models. An error analysis was performed using root mean square error (RMSE) and mean absolute percentage error (MAPE) metrics, and the results were compared across the models. The VMD-CNN model outperformed the VMD-LSTM and VMD-GRU models for all devices, except for the Rental Bed category, where the VMD-GRU model had an edge over the others. Based on the overall better performance of the VMD-CNN model, it was selected, and the predicted values were visualized in relation to the target values for each device type. Figures 8.3a, 8.3b, 8.3c, and 8.3d show the plots for each device type, where the count of the device is plotted against the time step, with each time step referring to a 30-minute interval. The VMD-CNN model was able to closely match the target values for all de-



a Asset Tag count forecasting

b Pulse oximeter count forecasting

c Rental bed count forecasting

d Supply cart count forecasting

Figure 8.3: Device count evolution over time step using VMD-CNN: predicted vs target values for different devices.

vices.

The system was designed to perform several horizon lookaheads, including a one-step lookahead to mean 30 minutes in the future, 2-step, 48-step, 336-step, and 1440-step to show a 1-hour, 1-day, week, and month look ahead, respectively. For the plots in Figure 8.3a, a one-step lookahead was used to show how closely the model matches the target values. Additionally, the behavior and statistics of each device type were observed, such as the Asset tag device, which had a constant count or small step changes over time, with occasional variations. This tricky data distribution could have been missed in LSTM models due to the bias created for longer durations, but the CNN model was able to capture it.

In Figure 8.3b, it is evident that the data distribution and its nature fluctuate signif-

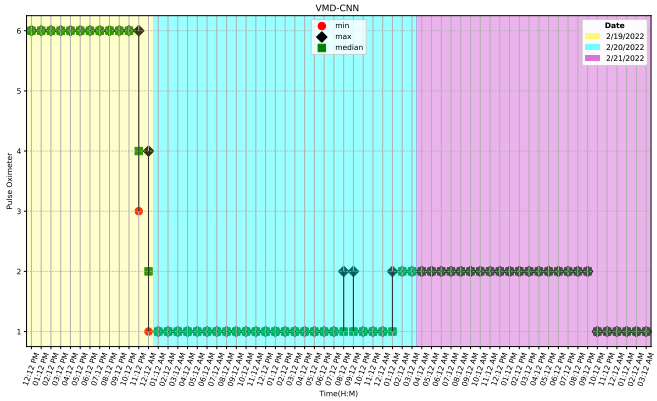
icantly, and this device type typically has poor error performance. However, the VMD-CNN model showed reasonable performance overall and performed better than the other models. The other two plots in Figures 8.3c and 8.3d had a data distribution that was intermediate between the two extreme distributions, and the system was able to perform well, closely matching the target values. These plots were also generated for a one-step horizon or one-step lookahead.

8.4.3 Range Prediction for Different Device Categories

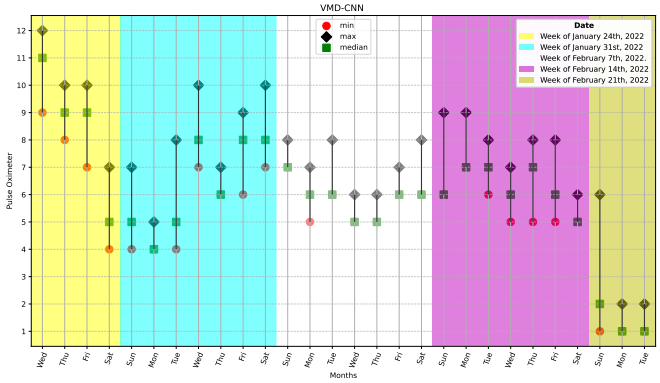
In hospital asset management, it is important to accurately predict the availability of devices in the inventory. In a recent set of experiments, we used VMD-CNN to predict the minimum, maximum, and median number of devices available for different device categories and time horizons. As shown in Fig. 8.4, we were able to accurately predict the availability of the Pulse Oximeter device hourly, daily, and monthly (see Table 8.2). When summarizing the predictions, it is useful to provide information about the spread of the data, which can be achieved by using the minimum and maximum values. The monthly availability range for the Pulse Oximeter device, as shown in Fig. 8.4c, is an example of this. Additionally, when describing the central tendency of the data, the median is often a better measure to use when the data is not normally distributed or if there are outliers present. This is particularly relevant in hospital asset management, where outliers can heavily influence the data. To obtain minimum, maximum, and median values, we perform multiple iterations of the prediction process. After running multiple iterations, we aggregate the results to compute the minimum, maximum, and median number of available devices. The minimum represents the lowest availability observed among all iterations, while the maximum represents the highest availability. The median is calculated by listing and arranging the results of all the iterations in orderly manner and finding the middle value . Proceeding with the discussion, We omitted other devices' simulation results to conserve space. Fig. 8.4a illustrates the hourly prediction for three

consecutive days in February 2022. The first day is Saturday starting from noon to 11 PM showing Pulse Oximeter will be available at its maximum at noon and then the availability decreases to a single device at night. On the second day of the weekend, Sunday, there is a larger time window of data points where the availability is the same as the previous day meaning that the hospital staff are not going to use the device. On the third day, Monday, which is Presidents' Day, when federal employees get the day off, the model forecasts the availability of the device will increase and remain the same all day except for the evening when it will be required perhaps due to an emergency situation (e.g., it could be a COVID patient who needs to measure their blood oxygen level).

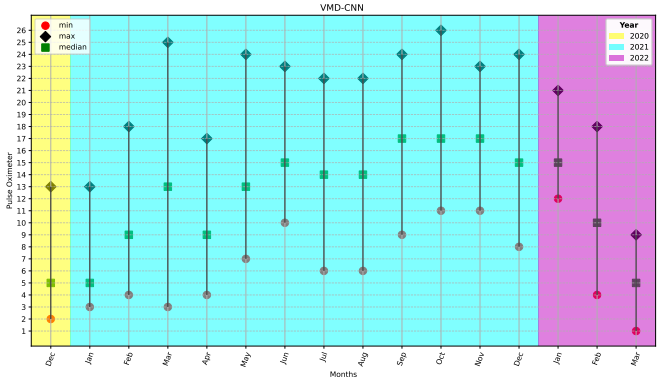
Fig. 8.4b demonstrates the daily predictions. As shown in the figure, each stick is associated with a day forecasting the minimum and maximum number of available devices within the inventory throughout that day. Fig. 8.4c shows the device availability flow monthly where in the last month of the year 2020 the device availability range is at the lowest amount implying that the demand for Pulse Oximeter in the hospital is high, whereas, throughout the year 2021, the availability fluctuates and reaches its peak showing the demand for this device is not as high. In early 2022, the availability decreases and reaches its minimum.



a Hourly range prediction.



b Daily range prediction.



c Monthly range prediction.

Figure 8.4: Pulse Oximeter range prediction for different time horizons using VMD-CNN.

8.4.4 Impact of Combining DL models with VMD

Finally, we examined the impact of incorporating VMD into the DL algorithms performance. The outcome is illustrated in Fig. 8.5. From the top figure, we can observe that the majority of device categories have profited from the combined approach and achieved higher performance except for Supply Cart. Additionally, the combined approach in the bottom figure shows better performance than the regular DL algorithm for all categories. This behavior is attested to the fact that the combined approach can help achieve better accuracy for the system.

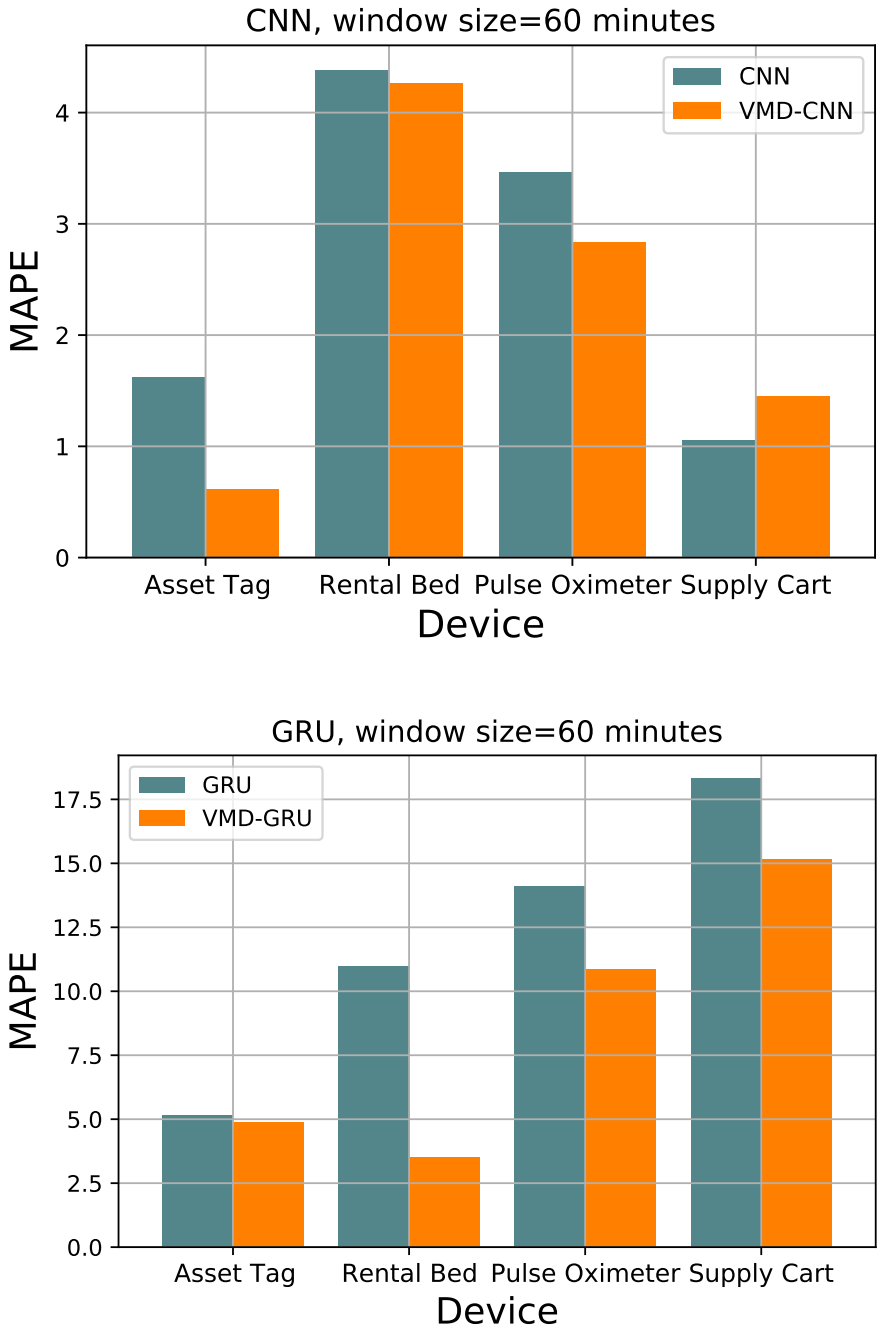


Figure 8.5: Accuracy comparison for combined CNN (top) and GRU (bottom) approaches.

8.5 Conclusions

Unfortunately, most of the inventory management tasks are carried by healthcare workers manually requiring training process and collaboration with vendors, align with hospital staff such as physicians and nurses imposing a burden on the workers. Automating the current hospital inventory management systems possesses multiple advantages such as releasing staff from logistics tasks, reducing the risks to improve the patient safety, eliminating the need for designing complex work flow systems, and processing a large amount of data to come up with an efficient state of inventory management.

In this paper, we proposed a DL-based framework that automates the tracking process of supply consumption and notifies the personnel to adjust the inventory levels. We used three DL models and their combination with VMD method to enhance the prediction accuracy using a real-world dataset. The VMD-CNN approach achieved the highest accuracy in comparison to other approaches. Our experimental results indicate that our design can substantially help monitor the medical asset usage accurately, whether used as a standalone system or in conjunction with other asset tracking tools. This study is replicable because it does not include sensitive data covered by HIPAA. While customers of this solution can securely store and communicate Protected Health Information in a limited manner, that data was explicitly excluded from this study and can easily be parsed out in future studies if it is present.

Acknowledgment

This paper is based upon work supported in part by the National Science Foundation under the New Mexico SMART Grid Center - EPSCoR cooperative agreement Grant OIA-1757207.

Chapter 9

Conclusion

9.1 Summary of Contributions

This thesis has explored the integration of machine learning (ML) and deep learning (DL) with EMC analysis, edge computing, and healthcare applications to develop intelligent EMI-aware systems. Key contributions include:

- **Advancing EMC with Waveform-Aware EMI Analysis:** Demonstrated that modulated EMI signals can increase coupling efficiency by up to 27% compared to CW signals, challenging traditional EMC assumptions.
- **Developing a GAN-Based EMI Classification Framework:** Introduced a deep learning model for EMI waveform classification, leveraging GAN-generated signal patterns for enhanced accuracy.
- **Applying EMI Mitigation to Medical Technology:** Proposed predictive ML models for hospital asset tracking, demonstrating improved equipment availability and reliability.
- **Optimizing Mobile Edge Computing for EMI-Aware Systems:** Implemented DRL-based workload offloading strategies, enhancing resource efficiency and reducing latency in MEC environments.
- **Integrating EMI Classification into Smart City Applications:** Established a unified ML-driven EMI mitigation framework for smart infrastructure, healthcare, and communication networks.

By combining EMC principles with AI-driven optimization, this research lays a foundation for next-generation EMI-aware digital systems, fostering resilience and adaptabil-

ity in real-world applications.

9.2 Future Research Directions

While this research provides significant contributions, several open challenges remain. Future work can explore:

- **Real-Time EMI Suppression:** Developing adaptive ML models for real-time EMI suppression using reinforcement learning (RL) to dynamically adjust system parameters.
- **Multi-Agent Reinforcement Learning for IoT Optimization:** Investigating multi-agent reinforcement learning (MARL) techniques to improve decision-making in large-scale IoT environments. MARL can enable decentralized decision-making, where each IoT device optimizes its own performance while cooperating with other agents to achieve system-wide efficiency.
- **Energy-Efficient AI for Smart Cities and IoT:** Exploring AI-driven optimization strategies to balance power consumption and latency in smart cities. Edge-based AI models can be further optimized for energy-efficient operations, reducing computation overhead while maintaining high predictive accuracy.
- **Security and Privacy in IoT-Based EMI Mitigation:** Addressing security concerns in EMI mitigation systems by integrating blockchain-based federated learning approaches. A decentralized learning framework can enhance data privacy while enabling collaborative model training across multiple IoT devices and medical institutions.
- **Automotive and Aerospace EMI Risk Assessment:** Extending GAN-based EMI classification techniques to automotive and aerospace systems, where EMI

effects can disrupt autonomous vehicle communication, avionics, and satellite networks.

- **Scalable Edge Computing for Healthcare Applications:** Enhancing real-time medical asset tracking and predictive maintenance using scalable ML models deployed in edge computing environments. Ensuring high reliability and fault tolerance in EMI-prone hospital settings remains a key challenge.

By addressing these future directions, upcoming research can further enhance EMI-aware system design, intelligent workload allocation, and data privacy in cloud-edge computing infrastructures. This work paves the way for a new generation of AI-powered EMI mitigation techniques, advancing smart cities, healthcare, and industrial applications.

References

[1] 1.9. naive bayes — scikit-learn 0.24.0 documentation. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed on January 5, 2021.

[2] CenTrak | Accurate Hospital Real-Time Location System Tracking Systems. "https://centrak.com/". Accessed on 2023-07-10.

[3] Support vector machine - mitosis technologies. [Online]. Available: https://www.mitosistech.com/support-vector-machine/. Accessed on January 5, 2021.

[4] Hamagen - israeli health ministry. https://govextra.gov.il/ministry-of-health/hamagen-app/download-en/, 2020.

[5] IoT enabled Contact Tracing, accessed 22 March 2021. https://github.com/zakhavan/IoT-enabled_Contact_Tracing.

[6] Cell Phone Trilateration Algorithm, accessed 29 December 2020. https://www.101computing.net/cell-phone-trilateration-algorithm/.

[7] Simple exponential smoothing, accessed 29 December 2020. https://otexts.com/fpp2/ses.html.

[8] Alaa A. Abd-alrazaq, Mohannad Alajlani, Dari Alhuwail, Aiman Erbad, Anna Giannicchi, Zubair Shah, Mounir Hamdi, and Mowafa Househ. Blockchain technologies to mitigate covid-19 challenges: A scoping review. *Computer Methods and Programs in Biomedicine Update*, page 100001, 2020.

[9] Michael Abrash. Latency–the sine qua non of ar and vr. *Blog post, Dec*, 2012.

[10] Tarek Abu Zwaida, Chuan Pham, and Yvan Beauregard. Optimization of inventory management to prevent drug shortages in the hospital supply chain. *Applied Sciences*, 11(6):2726, 2021.

[11] Yousif Abuidris, Rajesh Kumar, and Wang Wenyong. A survey of blockchain based on e-voting systems. In *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*, ICBTA 2019, page 99–104, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. A survey of covid-19 contact tracing apps. *IEEE Access*, 8:134577–134601, 2020.

[13] Zeinab Akhavan, Mona Esmaeili, Babak Badnava, Mohammad Yousefi, Xiang Sun, Michael Devetsikiotis, and Payman Zarkesh-Ha. Deep reinforcement learning for online latency aware workload offloading in mobile edge computing. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 2218–2223, 2022.

[14] Zeinab Akhavan, Mona Esmaeili, Dimitrios Sikeridis, and Michael Devetsikiotis. Internet of things-enabled passive contact tracing in smart cities. *Internet of Things*, 18:100397, 2022.

[15] Andrei R Akhmetzhanov, Kenji Mizumoto, Sung-mok Jung, Natalie M Linton, Ryosuke Omori, and Hiroshi Nishiura. Epidemiological characteristics of novel coronavirus infection: A statistical analysis of publicly available case data. *medRxiv*, 2020.

[16] William Dolan Alan Ritter, Colin Cherry. Data-driven response generation in social media. In *https://www.aclweb.org/anthology/D11-1054*, 2011.

[17] Taha Alfakih, Mohammad Mehedi Hassan, Abdu Gumaei, Claudio Savaglio, and Giancarlo Fortino. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa. *IEEE Access*, 8:54074–54084, 2020.

[18] Saeed H Alsamhi, Ou Ma, Mohammad Samar Ansari, and Faris A Almalki. Survey on collaborative smart drones and internet of things for improving smartness of smart cities. *IEEE Access*, 7:128125–128152, 2019.

[19] Carlos Miguel Ferreira Alves. Demand forecasting in a multi-specialty hospital setting: a comparative study of machine learning and classical statistical methods. 2018.

[20] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[21] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. P. Fitzek. Multi-agent based autonomic network management architecture. *IEEE Transactions on Network and Service Management*, pages 1–1, 2021.

[22] Sisay Tadesse Arzo, Zeinab Akhavan, Mona Esmaeili, Michael Devetsikiotis, and Fabrizio Granelli. Multi-agent-based traffic prediction and traffic classification for autonomic network management systems for future networks. *Future Internet*, 14(8):230, 2022.

[23] Sisay Tadesse Arzo, Claire Naiga, Fabrizio Granelli, Riccardo Bassoli, Michael Devetsikiotis, and Frank H. P. Fitzek. A theoretical discussion and survey of network automation for iot: Challenges and opportunity. *IEEE Internet of Things Journal*, 8(15):12021–12045, 2021.

[24] Sisay Tadesse Arzo, Dimitrios Sikeridis, Michael Devetsikiotis, Fabrizio Granelli, Rafael Fierro, Mona Esmaeili, and Zeinab Akhavan. Essential technologies and concepts for massive space exploration: Challenges and opportunities. *IEEE Transactions on Aerospace and Electronic Systems*, 59(1):3–29, 2023.

[25] Arzo, Sisay Tadesse, Scotece, Domenico, Bassoli, Riccardo, Barattini, Daniel, Granelli, Fabrizio, Foschini, Luca, and Fitzek, Frank H. P. Msn: A playground framework for design and evaluation of microservices-based sdn controller. *Journal of Network and Systems Management*, 30(1):1573–7705, 2021.

[26] Babak Badnava, Mona Esmaeili, Nasser Mozayani, and Payman Zarkesh-Ha. A new potential-based reward shaping for reinforcement learning agent. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 01–06, 2023.

[27] Babak Badnava, Taejoon Kim, Kenny Cheung, Zaheer Ali, and Morteza Hashemi. Spectrum-aware mobile edge computing for uavs using reinforcement learning. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 376–380, 2021.

[28] Georgios D Barmparis and GP Tsironis. Estimating the infection horizon of covid-19 in eight countries with a data-driven approach. *Chaos, Solitons & Fractals*, page 109842, 2020.

[29] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, and Tang Anh Quy. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, 2020.

[30] M. Behringer, A. Dutta, Y. Hertoghs, and S. Bjarnason. Autonomic networking - from theory to practice. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–17, Krakow, Poland, May 2014.

[31] Wanyu Bian, Panfeng Li, Mengyao Zheng, Chihang Wang, Anying Li, Ying Li, Haowei Ni, and Zixuan Zeng. A review of electromagnetic elimination methods for low-field portable mri scanner. In *2024 5th International Conference on Machine Learning and Computer Application (ICMLCA)*, pages 614–618. IEEE, 2024.

[32] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[33] Isobel Braithwaite, Thomas Callender, Miriam Bullock, and Robert W Aldridge. Automated and partly automated contact tracing: a systematic review to inform the control of covid-19. *The Lancet Digital Health*, 2020.

[34] Saeed Chehreh Chelgani, Hamid Nasiri, Arash Tohry, and Hassan Reza Heidari. Modeling industrial hydrocyclone operational variables by SHAP-CatBoost - A "conscious lab" approach. *Powder Technology*, 420:118416, 2023.

[35] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[36] Hechang Chen, Bo Yang, Hongbin Pei, and Jiming Liu. Next generation technology for epidemic prevention and control: Data-driven contact tracking. *Ieee Access*, 7:2633–2642, 2018.

[37] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[38] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[39] Konstantinos Christidis, Dimitrios Sikeridis, Yun Wang, and Michael Devetsikiotis. A framework for designing and evaluating realistic blockchain-based local energy markets. *Applied Energy*, 281:115963, 2021.

[40] Alberto Coustasse, Shane Tomblin, and Chelsea Slack. Impact of radio-frequency identification (rfid) technologies on the hospital supply chain: a literature review. *Perspectives in Health Information Management*, 10(Fall), 2013.

[41] Paolo Crovetti and Francesco Musolino. Digital suppression of emi-induced errors in a baseband acquisition front-end including off-the-shelf, emi-sensitive operational amplifiers. *Electronics*, 10(17):2096, 2021.

[42] Paolo Stefano Crovetti and Francesco Musolino. Interference of spread-spectrum emi and digital data links under narrowband resonant coupling. *Electronics*, 9(1):60, 2020.

[43] Open Sistemas de Información Internet S.L. Revision d0b241de. osbrain - 0.6.5. [Online]. Available: `https://osbrain.readthedocs.io/en/stable/`. Accessed on October 31, 2020.

[44] Jan De Vries. The shaping of inventory systems in health services: A stakeholder analysis. *International Journal of Production Economics*, 133(1):60–69, 2011.

[45] Wen Dong, Tong Guan, Bruno Lepri, and Chunming Qiao. Pocketcare: Tracking the flu with mobile phones using partial observations of proximity and symptoms. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–23, 2019.

[46] A. Dorri, S. S. Kanhere, and R. Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, Apr. 2018.

[47] M. L. Dowell and R. D. Bonnell. Learning for distributed artificial intelligence systems. In *[1991 Proceedings] The Twenty-Third Southeastern Symposium on System Theory*, pages 218–221, Columbia, SC, USA, Mar. 1991.

[48] Konstantin Dragomiretskiy and Dominique Zosso. Variational Mode Decomposition. *IEEE Transactions on Signal Processing*, 62(3):531–544, 2014.

[49] Xue Du, Guanghui Wei, Hongze Zhao, and Xiaodong Pan. Research on continuous wave electromagnetic effect in swept frequency radar. *Mathematical Problems in Engineering*, 2021:1–10, 2021.

[50] Qinglin Duan and T Warren Liao. A new age-based replenishment policy for supply chain inventory optimization of highly perishable products. *International journal of production economics*, 145(2):658–671, 2013.

[51] Ken TD Eames and Matt J Keeling. Contact tracing and disease control. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, 2003.

[52] Roman Egger. Topic modelling: modelling hidden semantic structures in textual data. In *Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications*, pages 375–403. Springer, 2022.

[53] Christopher Ifeanyi Eke, Azah Norman, Liyana Shuib, Faith B Fatokun, and Isaiah Omame. The significance of global vectors representation in sarcasm analysis. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pages 1–7. IEEE, 2020.

[54] Khalid Elgazzar, Patrick Martin, and Hossam S Hassanein. Cloud-assisted computation offloading to support mobile services. *IEEE Transactions on Cloud Computing*, 4(3):279–292, 2014.

[55] Mona Esmaeili, Mohammad Ahmadi, Mohammad David Ismaeil, Sharareh Mirzaei, and Jorge Canales Verdial. Advancements in ai-driven customer service. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 1–5, 2024.

[56] Mona Esmaeili, Zeinab Akhavan, Hamid Nasiri, Yonatan Melese Worku, Sisay Tadesse Arzo, Andreas Stavropoulos, Michael Devetsikiotis, and Payman Zarkesh-Ha. Medical asset management: Deep learning based asset usage prediction in a hospital setting using real data. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 944–951, 2023.

[57] Mona Esmaeili, Sameer D. Hemmady, Oameed Noakoasteen, Christos Christodoulou, Edl Schamiloglu, and Payman Zarkesh-Ha. Impact of waveform modulation from electromagnetic interference sources on coupling to digital electronic interconnects. In *2024 International Symposium on Electromagnetic Compatibility – EMC Europe*, pages 662–665, 2024.

[58] Qiang Fan and Nirwan Ansari. Application aware workload allocation for edge computing-based iot. *IEEE Internet of Things Journal*, 5(3):2146–2153, 2018.

[59] Qiang Fan, Jianan Bai, Hongxia Zhang, Yang Yi, and Lingjia Liu. Delay-aware resource allocation in fog-assisted iot networks through reinforcement learning. *IEEE Internet of Things Journal*, 2021.

[60] Rasoul Fatahi, Hamid Nasiri, Arman Homafar, Rasoul Khosravi, Hossein Siavoshi, and Saeed Chehreh Chelgani. Modeling operational cement rotary kiln variables with explainable artificial intelligence methods – a "conscious lab" development. *Particulate Science and Technology*, 41(5):715–724, 2023.

[61] Neil Ferguson, Daniel Laydon, Gemma Nedjati-Gilani, Natsuko Imai, Kylie Ainslie, Marc Baguelin, Sangeeta Bhatia, Adhiratha Boonyasiri, Zulma Cucunubá, Gina Cuomo-Dannenburg, et al. Report 9: Impact of non-pharmaceutical interventions

(npis) to reduce covid19 mortality and healthcare demand. *Imperial College London*, 10:77482, 2020.

[62] Agostino Forestiero and Giuseppe Papuzzo. Agents-based algorithm for a distributed information system in internet of things. *IEEE Internet of Things Journal*, 8(22):16548–16558, 2021.

[63] Francesca Fossati, Stefano Moretti, Patrice Perny, and Stefano Secci. Multi-resource allocation for network slicing. *IEEE/ACM Transactions on Networking*, 28, 2020.

[64] Open Networking Foundation. Open network operating system (onos).

[65] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[66] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[67] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13, 2016.

[68] Hexa-X. D1.2 – Expanded 6G vision, use cases and societal values — including aspects of sustainability, security and spectrum, Apr. 2021.

[69] Hexa-X. D5.1 – A flagship for B5G/6G vision and intelligent fabric of technology enablers connecting human, physical, and digital worlds, Dec. 2021.

[70] Hexa-X. D5.1 – AI-driven communication computation co-design: Gap analysis and blueprint, Aug. 2021.

[71] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[72] Peng Hu. Iot-based contact tracing systems for infectious diseases: Architecture and analysis. *arXiv preprint arXiv:2009.01902*, 2020.

[73] Binbin Huang, Zhongjin Li, Yunqiu Xu, Linxuan Pan, Shangguang Wang, Haiyang Hu, and Victor Chang. Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing. *Wireless Communications and Mobile Computing*, 2020, 2020.

[74] Mahdi Inaya, Michael Meli, Dimitrios Sikeridis, and Michael Devetsikiotis. A real-subject evaluation trial for location-aware smart buildings. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 301–306. IEEE, 2017.

[75] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. Cloudlet load balancing in wireless metropolitan area networks. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[76] Longquan Jiang, Bo Zhang, Qin Ni, Xuan Sun, and Pingping Dong. Prediction of snp sequences via gini impurity based gradient boosting method. *IEEE Access*, 7:12647–12657, 2019.

[77] Dong Jin, Christopher Hannon, Zhiyi Li, Pablo Cortes, Srinivasan Ramaraju, Patrick Burgess, Nathan Buch, and Mohammad Shahidehpour. Smart street lighting system: A platform for innovative smart city applications and a new frontier for cyber-security. *The Electricity Journal*, 29(10):28–35, 2016.

[78] Anshuman Kalla, Tharaka Hewa, Raaj Anand Mishra, Mika Ylianttila, and Madhusanka Liyanage. The role of blockchain to fight against covid-19. *IEEE Engineering Management Review*, 48(3):85–96, 2020.

[79] Nam Sung Kim, Todd Austin, David Baauw, Trevor Mudge, Krisztián Flautner, Jie S Hu, Mary Jane Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *computer*, 36(12):68–75, 2003.

[80] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[81] Ondrej Krejcar, Petra Maresova, Ali Selamat, Francisco José Melero, Sabina Barakovic, Jasmina Barakovic Husic, Enrique Herrera-Viedma, Robert Frischer, and Kamil Kuca. Smart furniture as a component of a smart city—definition based on key technologies specification. *IEEE Access*, 7:94822–94839, 2019.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[83] Mayura Kulkarni and Shubhangi Kale. Information retrieval based improvising search using automatic query expansion. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1226–1230. IEEE, 2021.

[84] Praveen Kumar, Lohith Reddy, and Shirshu Varma. Distance measurement and error estimation scheme for rssi based localization in wireless sensor networks. In *2009 Fifth international conference on wireless communication and sensor networks (WCSN)*, pages 1–4. IEEE, 2009.

[85] Rajesh Kumar, Abdullah Aman Khan, Sinmin Zhang, WenYong Wang, Yousif Abuidris, Waqas Amin, and Jay Kumar. Blockchain-federated-learning and

deep learning models for covid-19 detection using ct imaging. *arXiv preprint arXiv:2007.06537*, 2020.

[86] Rajesh Kumar, WenYong Wang, Jay Kumar, Ting Yang, Abdullah Khan, Wazir Ali, and Ikram Ali. An integration of blockchain and ai for secure data sharing and detection of ct images for the hospitals. *Computerized Medical Imaging and Graphics*, 87:101812, 2021.

[87] Salim Lahmiri. Comparing variational and empirical mode decomposition in forecasting day-ahead energy prices. *IEEE Systems Journal*, 11(3):1907–1910, 2015.

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[89] CKM Lee and Sathishwaran Palaniappan. Effective asset management for hospitals with rfid. In *2014 IEEE International Technology Management Conference*, pages 1–4. IEEE, 2014.

[90] Mark O Lewis, S Balaji, and Arun Rai. Rfid-enabled capabilities and their impact on healthcare process performance. 2010.

[91] Bingheng Li, Ling Zhang, Da Li, Zheming Gu, Bo Pu, Jun Fan, and Er-Ping Li. A novel deep reinforcement learning methodology for emi mitigation in multiscenario pinmap design. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 14(7):1273–1280, 2024.

[92] Hang Li. *Learning to rank for information retrieval and natural language processing*. Springer Nature, 2022.

[93] Xin Li, Piji Li, Wei Bi, Xiaojiang Liu, and Wai Lam. Relevance-promoting language model for short-text conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8253–8260, 2020.

[94] Tengteng Liu, Xiang Ma, Shuo Li, Xuemei Li, and Caiming Zhang. A stock price prediction method based on meta-learning and variational mode decomposition. *Knowledge-Based Systems*, 252:109324, sep 2022.

[95] Yanchen Liu, Myung J Lee, and Yanyan Zheng. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. *IEEE Transactions on Mobile Computing*, 15(10):2398–2410, 2015.

[96] X. Long, X. Gong, X. Que, W. Wang, B. Liu, S. Jiang, and N. Kong. Autonomic networking: Architecture design and standardization. *IEEE Internet Computing*, 21(5):48–53, Sept. 2017.

[97] Wenzhe Lv, Sheng Wu, Chunxiao Jiang, Yuanhao Cui, Xuesong Qiu, and Yan Zhang. Decentralized blockchain for privacy-preserving large-scale contact tracing. *arXiv preprint arXiv:2007.00894*, 2020.

[98] "Tayeb Ben Meriem, Ranganai Chaparadza, Benoît Radier, Said Soulhi, José-Antonio LozanoLópez, and Arun Prakash". GANA - Generic Autonomic Networking Architecture. 2016.

[99] Joseph Meyer, Reza Moghimi, and Noah Sturcken. Package voltage regulators: The answer for power management challenges. In *International Symposium on Microelectronics*, volume 2019, pages 000438–000443. International Microelectronics Assembly and Packaging Society, 2019.

[100] Katina Michael and Roba Abbas. Behind covid-19 contact trace apps: the google–apple partnership. *IEEE Consumer Electronics Magazine*, 9(5):71–76, 2020.

[101] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[102] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[103] Saraju P Mohanty, Uma Choppali, and Elias Kougianos. Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, 5(3):60–70, 2016.

[104] Karen Moons, Geert Waeyenbergh, and Liliane Pintelon. Measuring the logistics performance of internal hospital supply chains–a literature study. *Omega*, 82:205–217, 2019.

[105] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. A survey of autonomic network architectures and evaluation criteria. *IEEE Communications Surveys Tutorials*, 14(2):464–490, 2012.

[106] Anwesha Mukherjee, Debashis De, and Deepsubhra Guha Roy. A power and latency aware cloudlet selection strategy for multi-cloudlet environment. *IEEE Transactions on Cloud Computing*, 7(1):141–154, 2016.

[107] Md Shirajum Munir, DH Kim, SF Abedin, and CS Hong. A risk-sensitive social distance recommendation system via bluetooth towards the covid-19 private safety. In *Korean Computer Congress (KCC)*, 2020.

[108] Seyedsina Nabavirazavi, SS Iyengar, and Naveen Kumar Chaudhary. Lightweight malicious packet classifier for iot networks. In *International Conference on Information Security, Privacy and Digital Forensics*, pages 139–150. Springer, 2022.

[109] Amir Nafchi, Mona Esmaeili, Balu Santhanam, and Ramiro Jordan. 1 a new affine fractional fourier transform with circular convolution property. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pages 111–114, 2023.

[110] Amir Raeisi Nafchi, Mona Esmaeili, Alireza Ghasempour, Eric Hamke, Balu Santhanam, and Ramiro Jordan. Mitigating the time-varying doppler shift in high-mobility wireless communications using multi-angle centered discrete fractional fourier transform. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0607–0612, 2021.

[111] Hamid Nasiri and Mohammad Mehdi Ebadzadeh. MFRFNN: Multi-Functional Recurrent Fuzzy Neural Network for Chaotic Time Series Prediction. *Neurocomputing*, 507:292–310, 2022.

[112] Hamid Nasiri and Mohammad Mehdi Ebadzadeh. Multi-step-ahead Stock Price Prediction Using Recurrent Fuzzy Neural Network and Variational Mode Decomposition. *arXiv preprint arXiv:2212.14687*, 2022.

[113] Arundhati Navada, Aamir Nizam Ansari, Siddharth Patil, and Balwant A Sonkamble. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE control and system graduate research colloquium*, pages 37–42. IEEE, 2011.

[114] Victor-Emil Neagoe, Adrian-Dumitru Ciotec, and George-Sorin Cucu. Deep convolutional neural networks versus multilayer perceptron for financial prediction. In *2018 International Conference on Communications (COMM)*, pages 201–206, 2018.

[115] Pai Chet Ng, Petros Spachos, Stefano Gregori, and Konstantinos Plataniotis. Epidemic exposure notification with smartwatch: A proximity-based privacy-preserving approach. *arXiv preprint arXiv:2007.04399*, 2020.

[116] Pai Chet Ng, Petros Spachos, and Konstantinos Plataniotis. Covid-19 and your smartphone: Ble-based smart contact tracing. *arXiv preprint arXiv:2005.13754*, 2020.

[117] Angie Nguyen, Samir Lamouri, Robert Pellerin, Simon Tamayo, and Béranger Lekens. Data analytics in pharmaceutical supply chains: state of the art, opportunities, and challenges. *International Journal of Production Research*, pages 1–20, 2021.

[118] Van Ha Nguyen, Hai Au Huynh, SoYoung Kim, and Hanjung Song. Active emi reduction using chaotic modulation in a buck converter with relaxed output lc filter. *Electronics*, 7(10):254, 2018.

[119] Garima Nishad. Kalman Filters: A step by step implementation guide in python, accessed 29 December 2020. `https://towardsdatascience.com/kalman-filters-a-step-by-step-implementation-guide-in-python-91e7e123b968`.

[120] Oameed Noakoasteen, Jayakrishnan Vijayamohanan, Arjun Gupta, and Christos Christodoulou. Antenna design using a gan-based synthetic data generation approach. *IEEE Open Journal of Antennas and Propagation*, 3:488–494, 2022.

[121] Hadab Khalid Obayes, Firas Sabah Al-Turaihi, and Khaldoon H Alhussayni. Sentiment classification of user's reviews on drugs based on global vectors for word representation and bidirectional long short-term memory recurrent neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(1):345–353, 2021.

[122] Diego Ordóñez-Camacho and Edwin Cabrera-Goyes. An adaptive-bounds bandpass moving-average filter to increase precision on distance estimation from bluetooth rssi. In *International Conference on Information Theoretic Security*, pages 823–832. Springer, 2018.

[123] Young Joon Park, Young June Choe, Ok Park, Shin Young Park, Young-Man Kim, Jieun Kim, Sanghui Kweon, Yeonhee Woo, Jin Gwack, Seong Sun Kim, et al. Contact tracing during coronavirus disease outbreak, south korea, 2020. *Emerging infectious diseases*, 26(10):2465–2468, 2020.

[124] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[125] Nipun Ramakrishnan and Tarun Soni. Network traffic prediction using recurrent neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 187–193, 2018.

[126] James Rappold, Ben Van Roo, Christine Di Martinelly, and Fouad Riane. An inventory optimization model to support operating room schedules. In *Supply Chain Forum: An International Journal*, volume 12, pages 56–69. Taylor & Francis, 2011.

[127] S. Ray. A quick review of machine learning algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 35–39, Faridabad, India, 2019.

[128] Alan Ritter, Colin Cherry, and Bill Dolan. Data-driven response generation in social media. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[129] Ronald L Rivest, DJ Weitzner, LC Ivers, I Soibelman, and MA Zissman. Pact: Private automated contact tracing, 2020.

[130] Yara Rizk, Mariette Awad, and Edward W Tunstel. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529, 2018.

[131] Claudia R Rosales, Michael Magazine, and Uday Rao. The 2bin system for controlling medical supplies at point-of-use. *European Journal of Operational Research*, 243(1):271–280, 2015.

[132] Sebastian Sadowski and Petros Spachos. Rssi-based indoor localization with the internet of things. *IEEE Access*, 6:30149–30161, 2018.

[133] N. Samaan and A. Karmouch. Towards autonomic network management: an analysis of current and future research directions. *IEEE Communications Surveys Tutorials*, 11(3):22–36, 2009.

[134] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[135] Michael J Scott, Wim JC Verhagen, Marie T Bieber, and Pier Marzocca. A systematic literature review of predictive maintenance for defence fixed-wing aircraft sustainment and operations. *Sensors*, 22(18):7070, 2022.

[136] Claire Naiga Serugunda, Sisay Tadesse Arzo, Fabrizio Granelli, Riccardo Bassoli, Michael Devetsikiotis, and Frank H.P. Fitzek. Autonomous network traffic classifier agent for autonomic network management system. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

[137] MJ Shaw, B Harrow, and S Herman. Distributed artificial intelligence for multi-agent problem solving and group learning. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume 4, pages 13–26. IEEE, 1991.

[138] Yuriy Shlepnev. Decompositional electromagnetic analysis of digital interconnects. In *2013 IEEE International Symposium on Electromagnetic Compatibility*, pages 563–568. IEEE, 2013.

[139] Dimitrios Sikeridis, Ali Bidram, Michael Devetsikiotis, and Matthew J Reno. A blockchain-based mechanism for secure data exchange in smart grid protection systems. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2020.

[140] Dimitrios Sikeridis, Michael Devetsikiotis, and Ioannis Papapanagiotou. Occupant tracking in smart facilities: An experimental study. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 818–822. IEEE, 2017.

[141] Dimitrios Sikeridis, Ioannis Papapanagiotou, and Michael Devetsikiotis. Blebeacon: A real-subject trial dataset from mobile bluetooth low energy beacons. *arXiv preprint arXiv:1802.08782*, 2018.

[142] Dimitrios Sikeridis, Ioannis Papapanagiotou, and Michael Devetsikiotis. Crawdad dataset unm/blebeacon (v. 2019-03-12), 2019.

[143] Dimitrios Sikeridis, Ioannis Papapanagiotou, Bhaskar Prasad Rimal, and Michael Devetsikiotis. A comparative taxonomy and survey of public cloud infrastructure vendors. *arXiv preprint arXiv:1710.01476*, 2017.

[144] Dimitrios Sikeridis, Bhaskar Prasad Rimal, Ioannis Papapanagiotou, and Michael Devetsikiotis. Unsupervised crowd-assisted learning enabling location-aware facilities. *IEEE Internet of Things Journal*, 5(6):4699–4713, 2018.

[145] Fikret Sivrikaya, Nizar Ben-Sassi, Xuan-Thuy Dang, Orhan Can Görür, and Christian Kuster. Internet of smart city objects: A distributed framework for service discovery and composition. *IEEE Access*, 7:14434–14454, 2019.

[146] Gan Kim Soon, Chin Kim On, Patricia Anthony, and Abdul Razak Hamdan. A review on agent communication language. In Rayner Alfred, Yuto Lim, Ag Asri Ag

Ibrahim, and Patricia Anthony, editors, *Computational Science and Technology*, pages 481–491, Singapore, 2019. Springer Singapore.

[147] Pablo Sotres, Juan Ramón Santana, Luis Sánchez, Jorge Lanza, and Luis Munoz. Practical lessons from the deployment and management of a smart city internet-of-things infrastructure: The smartsantander testbed case. *IEEE Access*, 5:14309–14322, 2017.

[148] Xiang Sun and Nirwan Ansari. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016.

[149] Xiang Sun and Nirwan Ansari. Primal: Profit maximization avatar placement for mobile edge computing. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[150] Xiang Sun and Nirwan Ansari. Latency aware workload offloading in the cloudlet network. *IEEE Communications Letters*, 21(7):1481–1484, 2017.

[151] Xiang Sun and Nirwan Ansari. Adaptive avatar handoff in the cloudlet network. *IEEE Transactions on Cloud Computing*, 7(3):664–676, 2019.

[152] Xiang Sun and Nirwan Ansari. Green cloudlet network: A sustainable platform for mobile cloud computing. *IEEE Transactions on Cloud Computing*, 8(1):180–192, 2020.

[153] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[154] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

[155] PEPP-PT Team. Pan-european privacy-preserving proximity tracing, 2020.

[156] Pietro Tedeschi, Spiridon Bakiras, and Roberto Di Pietro. Iotrace: A flexible, efficient, and privacy-preserving iot-enabled architecture for contact tracing. *arXiv preprint arXiv:2007.11928*, 2020.

[157] Aakash Thapa, Teerayut Horanont, Bipul Neupane, and Jagannath Aryal. Deep learning for remote sensing image scene classification: A review and meta-analysis. *Remote Sensing*, 15(19):4804, 2023.

[158] K. Tsagkaris, M. Logothetis, V. Foteinos, G. Poulios, M. Michaloliakos, and P. Demestichas. Customizable autonomic network management: Integrating autonomic network management and software-defined networking. *IEEE Vehicular Technology Magazine*, 10(1):61–68, 2015.

[159] G Krishna Vamsi, Akhtar Rasool, and Gaurav Hajela. Chatbot: A deep neural network based human to machine conversation model. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2020.

[160] Lifeng Wang, Qingyu Li, Haoyi Yang, Ziyi Lin, and Junguo Zhang. A lightweight residual network based on pixel-level hyperspectral image for rice variety classification. *Available at SSRN 4866836*.

[161] Weitao Wang, Yuebin Bai, Chao Yu, Yuhao Gu, Peng Feng, Xiaojing Wang, and Rui Wang. A network traffic flow prediction with deep learning approach for large-scale metropolitan area network. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2018.

[162] Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, and Wendell Ela. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441:161–178, 2021.

[163] Kilian Q Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18:1473–1480, 2005.

[164] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*, 2017.

[165] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[166] H. Xu, L. Zhang, O. Onireti, Y. Fang, W. J. Buchanan, and M. A. Imran. Beeptrace: Blockchain-enabled privacy-preserving contact tracing for covid-19 pandemic and beyond. *IEEE Internet of Things Journal*, pages 1–1, 2020.

[167] Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the Conference on Information and Knowledge Management*, 2022.

[168] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pages 98–101, 2020.

[169] Wen Yao, Chao-Hsien Chu, and Zang Li. The use of rfid in healthcare: Benefits and barriers. In *2010 IEEE International Conference on RFID-Technology and Applications*, pages 128–134. IEEE, 2010.

[170] Mu Ye, Ni Ruiwen, Zhang Chang, Gong He, Hu Tianli, Li Shijun, Sun Yu, Zhang Tong, and Guo Ying. A lightweight model of vgg-16 for remote sensing image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:6916–6922, 2021.

[171] Chris Yoon. Understanding actor critic methods and a2c. [Online]. Available: `https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f`.

[172] Curtis Yu, Cristian Lumezanu, Abhishek Sharma, Qiang Xu, Guofei Jiang, and Harsha V Madhyastha. Software-defined latency monitoring in data center networks. In *International Conference on Passive and Active Network Measurement*, pages 360–372. Springer, 2015.

[173] Hu Yun, Zhuoran Sun, Jun Wu, Aiguo Tang, Min Hu, and Zhongyuan Xiang. Laboratory data analysis of novel coronavirus (covid-19) screening in 2510 patients. *Clinica Chimica Acta*, 2020.

[174] ZeroMQ. An open-source universal messaging library.

[175] Weiwen Zhang, Yonggang Wen, Kyle Guan, Dan Kilper, Haiyun Luo, and Dapeng Oliver Wu. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9):4569–4581, 2013.

[176] Yujiao Zhao, Linfang Xiao, Vick Lau, Yilong Liu, Alex T Leong, and Ed X Wu. Robust electromagnetic interference (emi) elimination via simultaneous sensing and

deep learning prediction for rf shielding-free mri. *arXiv preprint arXiv:2210.06730*, 2022.

[177] Xuan Zhou, Zhifeng Zhao, Rongpeng Li, Yifan Zhou, Jacques Palicot, and Honggang Zhang. Human mobility patterns in cellular networks. *IEEE communications letters*, 17(10):1877–1880, 2013.

[178] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*, 2021.

[179] Hang Li Zongcheng Ji, Zhengdong Lu. An information retrieval approach to short text conversation. In *https://arxiv.org/pdf/1408.6988.pdf*, 2014.