DISSERTATION

MACHINE LEARNING FOR OPTIMIZATION AND DECISION SUPPORT IN COMPLEX SYSTEM-OF-SYSTEMS: APPLICATIONS IN MICROELECTRONICS, HEALTHCARE, SMART CITIES, AND THE CIRCULAR ECONOMY

Submitted by

Arifuzzaman (Arif) Sheikh

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2025

Doctoral Committee:

Advisor: Edwin K. P. Chong

Steven J. Simske
Thomas H. Bradley
Xinran Wang

ABSTRACT

MACHINE LEARNING FOR OPTIMIZATION AND DECISION SUPPORT IN COMPLEX SYSTEM-OF-SYSTEMS: APPLICATIONS IN MICROELECTRONICS, HEALTHCARE, SMART CITIES, AND THE CIRCULAR ECONOMY

Machine Learning (ML)offers powerful tools for optimizing decision-making across complex System of Systems (SoS) environments characterized by decentralization, dynamic interactions, and resource constraints. This work develops AI-driven frameworks tailored to five critical domains: semiconductor manufacturing, electronics quality control, healthcare coordination, smart city infrastructure, and circular economy marketplaces. Each framework integrates domain-specific modeling with ML techniques to enable scalable, adaptive, and cooperative decision-making.

In Three-Dimensional Integrated Circuit (3D-IC) manufacturing, a multi-stage ML system combines Random Forests, Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) models to align defect detection and process optimization with global yield objectives. Dual annealing and adaptive weighting enhance Through-Silicon Via (TSV) formation reliability, supporting system-wide performance gains.

For electronics quality control, Natural Language Processing (NLP) and Naïve Bayes classification are applied to technician reports from Printed Circuit Board (PCB) testing, transforming unstructured narratives into structured insights. This approach improves component-level fault detection and demonstrates the role of human-in-the-loop analytics in manufacturing SoS.

In healthcare, a Multi-Agent Reinforcement Learning (MARL)framework models coordination among hospitals, telemedicine platforms, and wearable devices. Simulated SoS configurations—Directed, Acknowledged, Collaborative, and Virtual—reveal that decentralized

architectures offer superior adaptability, cooperation, and efficiency in Artificial Intelligence of Medical Things (AIoMT)-enabled environments.

Smart city infrastructure is addressed through a MARL-based SoS framework that optimizes transportation, energy, and public safety systems via policy learning and augmented reward design. Simulations demonstrate improvements in coordination, efficiency, and system responsiveness.

In circular economy digital marketplaces, time-series forecasting (e.g., AutoRegressive Integrated Moving Average (ARIMA), LSTM), Gradient Boosting Regressor (GBR), and reinforcement learning are used for resource prediction and price stabilization. The models enable dynamic supply-demand matching, supporting sustainable resource flows and demonstrating the potential of Artificial Intelligence (AI) for market-based Circular Economy (CE) implementation.

Together, these case studies demonstrate how ML techniques can improve system-wide performance, autonomy, and cooperation across interdependent domains. The contributions span algorithm design, SoS coordination strategies, and domain integration, offering a foundation for deploying ML in critical infrastructure and complex engineered systems.

ACKNOWLEDGEMENTS

All praise and thanks are due to God, the Most Gracious and Most Merciful, whose guidance and mercy have sustained me throughout this academic journey.

I would like to express my deepest gratitude to my advisor, Dr. Edwin K. P. Chong, for his exceptional guidance, steadfast support, and inspiring mentorship. His depth of knowledge, clarity of thought, and unwavering commitment to excellence have profoundly shaped both this dissertation and my growth as a researcher.

I am also sincerely thankful to the members of my dissertation committee — Dr. Steven S. Simske, Dr. Thomas H. Bradley, and Dr. Xinran Wang — for their invaluable feedback, thoughtful critique, and scholarly insight. Their contributions have enriched the intellectual depth and rigor of this work.

I wish to acknowledge my colleagues and friends for their encouragement, camaraderie, and moral support throughout this journey. Their presence has brought both balance and joy to the challenges of academic life.

Above all, I am profoundly grateful to my family — my parents, children, and my beloved wife — for their unconditional love, patience, and unwavering belief in me. Their sacrifices and encouragement have been the foundation of my perseverance, and I dedicate this achievement to them.

DEDICATION

In the name of God, the Most Gracious, the Most Merciful. All praise and thanks are due to God alone, who granted me the strength, patience, and clarity to complete this journey.

This dissertation is also dedicated to my parents, children, and my beloved wife — whose unwavering love, encouragement, and belief in me have been the foundation of my academic journey. Your constant support and countless sacrifices have inspired me to pursue my goals with passion, resilience, and purpose. This achievement is not mine alone, but ours, and I am forever grateful for your presence in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ACRONYMS

3D-IC Three-Dimensional Integrated Circuit *on page(s):* ii, viii, xiii, 4, 9, 17, 19, 20, 22, 24, 25, 33, 34, 54, 86–90, 93, 99, 101, 103, 124, 128, 129, 176, 343, 345, 348, 350, 360, 363

A2C Advantage Actor-Critic *on page(s):* 78

AAA Agent Adaptability Assessment *on page(s):* 264, 268–270, 278, 284

AI Artificial Intelligence *on page(s):* ii, iii, 5, 9–11, 13, 20, 31, 32, 88, 89, 93, 94, 99, 103, 123–128, 131, 156, 157, 164, 165, 202, 204, 205, 209, 222, 248, 288–292, 294, 299, 307, 309–311, 336–340, 342, 345, 347, 353, 367, 368, 370

AIoMT Artificial Intelligence of Medical Things *on page(s):* iii, ix, xv, 5, 8, 10, 14, 18, 22, 23, 33, 35, 54, 203–206, 208, 210, 211, 219, 244, 248, 344, 349, 355, 360, 363, 364

ANFIS Adaptive Neuro-Fuzzy Inference System *on page(s):* 29

ANOVA Analysis of Variance *on page(s):* 157, 158, 167, 172, 176

AOI Automated Optical Inspection *on page(s):* 178

API Application Programming Interface *on page(s):* 366, 368

ARIMA AutoRegressive Integrated Moving Average *on page(s):* iii, xi, xviii, 5, 20, 23, 59, 61, 62, 289, 300–302, 311, 312, 320–323, 336, 337, 339

AUC Area Under the Curve *on page(s):* xiii–xv, 106–108, 115, 117, 118, 124, 125, 127, 192, 193

BEOL Back End of the Line *on page(s):* 6, 60

CCC Coordination Consistency Coefficient *on page(s):* 264, 268–270, 274, 280, 284

CE Circular Economy *on page(s):* iii, 5, 10, 31, 32, 35, 288–294, 298–310, 312–320, 336–340, 344, 345, 347, 349, 353, 361, 363, 364, 367

CNN Convolutional Neural Network *on page(s):* xiii, 9, 13, 17, 19, 23, 24, 26, 33, 34, 36, 61, 62, 88, 92, 96–98, 109, 126, 127, 129, 176, 264, 266, 267, 269, 350, 360, 362, 363

CNNs Convolutional Neural Networks *on page(s):* ii, 4

CONOPS Concept of Operations *on page(s):* xvii, 260, 261

CPU Central Processing Unit *on page(s):* 76, 126, 350

CTGAN Conditional Tabular Generative Adversarial Network *on page(s):* 293

DDPG Deep Deterministic Policy Gradient *on page(s):* 78

DQL Deep Q-Learning *on page(s):* 289

DQN Deep Q-Network *on page(s):* xix, 317, 332–334

EDA Electronic Design Automation *on page(s):* 293

EEE Efficiency Evaluation Equation *on page(s):* 264, 268–270, 274, 278, 280, 284

EV Electric Vehicle *on page(s):* 253, 254, 260, 265, 268, 270, 274, 278, 280, 282, 284

FPGA Field-Programmable Gate Array *on page(s):* xv, 179, 189–192, 199, 200

GAN Generative Adversarial Network *on page(s):* 24, 93

GBR Gradient Boosting Regressor *on page(s):* iii, xii, xix, 5, 23, 61, 62, 289, 305, 306, 315, 318, 319, 329–332, 334–340

GPU Graphics Processing Unit *on page(s):* 76, 78, 83, 127, 350

HITL Human-in-the-Loop *on page(s):* 351

HRL Hierarchical Reinforcement Learning *on page(s):* 34, 54, 131, 132, 177

IPWFNN-DRLA Intelligent Probabilistic Wavelet Fuzzy Neural Network–Deep Reinforcement Learning Algorithm *on page(s):* 31

KNN K-Nearest Neighbors *on page(s):* 170

LCA Latent Class Analysis *on page(s):* 184, 185, 187, 196

LDA Latent Dirichlet Allocation *on page(s):* 187

LIME Local Interpretable Model-agnostic Explanations *on page(s):* 338, 347, 352

LP Linear Programming *on page(s):* 61, 62, 161

LR Linear Regression *on page(s):* xviii, xix, 300, 301, 322, 335, 336

LSA Latent Semantic Analysis *on page(s):* 185–187

LSTM Long Short-Term Memory *on page(s):* ii, iii, viii, xii, xiii, xviii, 4, 5, 9, 13, 17, 18, 20, 23, 25, 26, 33, 34, 36, 59, 61, 62, 86, 88, 92, 99–101, 103, 114, 125–127, 130, 176,

187, 263, 289, 299, 301, 302, 311, 313, 323–326, 328, 336, 339, 345, 347, 349, 350, 352, 360, 362, 363

MA2C Multi-Agent Advantage Actor-Critic with Communication *on page(s):* 22, 23, 30

MAE Mean Absolute Error *on page(s):* 290, 299, 307–309, 311–318, 335, 338

MAML Multi-Agent Machine Learning *on page(s):* 256, 268

MARL Multi-Agent Reinforcement Learning *on page(s):* ii, iii, vi–viii, xi, xiii, xv, xvii, 1, 4–10, 13, 15, 17, 18, 21–23, 30–33, 35–38, 41, 43, 48, 50, 52, 53, 55, 56, 58, 59, 61–65, 67, 68, 70, 71, 73, 75–77, 79, 80, 83–85, 133, 135, 205, 206, 209, 211, 213, 219, 222, 243, 245–252, 254, 256, 257, 284–286, 343–345, 347, 348, 350, 360, 362–366

MAS Multi-Agent Systems *on page(s):* xi, 63, 131

MBSE Model-Based Systems Engineering *on page(s):* 16, 17, 22, 250

MDP Markov Decision Process *on page(s):* 215, 223

MES Manufacturing Execution Systems *on page(s):* 174

ML Machine Learning *on page(s):* ii, iii, vi, viii, x, 1–4, 6–14, 17, 23, 24, 26, 27, 30, 32–37, 57, 58, 86, 130, 176–180, 248, 341, 342, 344–346, 348, 351–370

MSE Mean Squared Error *on page(s):* xxix, 171, 299, 308, 309, 311, 313, 315, 317, 318, 335

NLP Natural Language Processing *on page(s):* ii, vi, ix, 2, 5, 8, 10, 11, 13, 21, 27, 29, 30, 54, 59, 61, 62, 177–179, 360

NN Neural Networks *on page(s):* xix, 201, 306, 315, 316, 329, 331, 336

PCB Printed Circuit Board *on page(s):* ii, ix, 5, 10, 18, 21, 27, 33, 35, 54, 59, 62, 177–179, 202, 346, 351, 360, 366

PPO Proximal Policy Optimization *on page(s):* xvii, 61, 62, 78, 256, 257, 267, 269, 278, 284, 286, 346, 347, 349, 350

PuLP Python Linear Programming *on page(s):* 154, 160, 161

QC Quality Control *on page(s):* xiv, xv, 166–169

ReLU Rectified Linear Unit *on page(s):* 101, 307, 337

RF Random Forest *on page(s):* xviii, xix, 61, 305, 326, 336

RFC Random Forest Classifier *on page(s):* 19

RL Reinforcement Learning *on page(s):* 1, 18, 21–23, 30, 35, 62, 63, 78, 83, 203, 250, 256, 289, 304

RMSE Root Mean Square Error *on page(s):* 308, 309, 311, 313, 315, 317, 318, 335

RNN Recurrent Neural Network *on page(s):* 20, 263, 264, 266, 267, 269

ROC Receiver Operating Characteristic *on page(s):* xiii, xiv, 106–108, 115, 117, 118, 124, 125, 127

ROPE Role-Oriented Programming Environment *on page(s):* vi, 13, 16, 17, 285

RQ Research Question *on page(s):* 4, 5, 88, 132, 147, 180, 206, 252, 290

RRR Resilience and Reliability Rating *on page(s):* 265, 268–270

SAM Scanning Acoustic Microscopy *on page(s):* 24

SE Systems Engineering *on page(s):* x, 250, 251, 361

SHAP SHapley Additive exPlanations *on page(s):* 338, 347, 352, 366

SME Small and Medium-sized Enterprises *on page(s):* ix, 4, 6, 8, 10, 25, 26, 34, 35, 54, 59, 61, 62, 85, 86, 145–147, 149–153, 155, 158, 166, 170–177, 341, 351, 360

SMOTE Synthetic Minority Over-sampling Technique *on page(s):* xiii, xiv, 19, 92–94, 99–101, 104, 114–116, 125, 126

SoS System of Systems *on page(s):* ii, iii, vi–viii, x, xi, xiii–xviii, 1–19, 21–24, 32–44, 48–59, 63–65, 67–69, 73, 75, 77, 79, 80, 83–91, 101, 103, 119–121, 123, 124, 128–132, 137, 144–146, 176, 177, 179, 180, 202, 204–208, 210, 211, 214–221, 223–261, 263–271, 273–275, 279–282, 284–287, 289, 290, 340–346, 348–370

SPC Statistical Process Control *on page(s):* 63, 64

SSS Scalability Sensitivity Score *on page(s):* 265, 268–270

SVM Support Vector Machines *on page(s):* 61, 91, 92, 170, 182, 201

SVR Support Vector Regression *on page(s):* 171

t-SNE t-Distributed Stochastic Neighbor Embedding *on page(s):* xv, 198, 199, 201, 366

TCAD Technology Computer-Aided Design *on page(s):* 26

TD Temporal Difference *on page(s):* 65, 79

TF-IDF Term Frequency–Inverse Document Frequency *on page(s):* 182, 186–188

TPU Tensor Processing Unit *on page(s):* 127

TSV Through-Silicon Via *on page(s):* ii, xiii, 4, 8, 9, 20, 24, 26, 34, 54, 60, 61, 86–92, 94, 96–98, 100, 101, 103, 109, 110, 112, 113, 119, 123–127, 129, 343, 345, 348, 350, 360, 363

WAVE Workflow, Agent, View, and Environment *on page(s):* vi, 13, 16, 285

X-ray X-ray Inspection *on page(s):* 19, 24, 178

X-SEM Extended System Engineering Methodology *on page(s):* vi, 13, 16, 17

XAI Explainable Artificial Intelligence *on page(s):* 338

XGBoost Extreme Gradient Boosting *on page(s):* 61, 91, 126, 127, 347

Chapter 1

Introduction

1.1 Background and Context

Modern critical infrastructure systems—ranging from semiconductor fabrication and healthcare coordination to smart cities and digital marketplaces—are increasingly characterized by complexity, decentralization, and interdependence. These environments often function as *System of Systems (SoS)*, composed of independently managed, goal-oriented subsystems that must coordinate dynamically to achieve shared objectives. In such contexts, traditional rule-based or centrally planned optimization techniques struggle to keep pace with the demands of scale, adaptability, and responsiveness.

Machine Learning (ML) provides a transformative foundation for addressing these challenges. By learning from data, adapting to new scenarios, and optimizing policies over time, Machine Learning (ML) systems enable more scalable and autonomous decision-making. In particular, reinforcement learning (RL) and its multi-agent extension (MARL) offer mechanisms for agents to interact within complex environments, refine their behavior through trial and error, and cooperate—or compete—based on localized or global feedback.

In manufacturing environments, Machine Learning (ML) has shown promise in tasks such as defect detection, predictive maintenance, scheduling, and process control. When applied within a SoS framework, ML allows for subsystem-level intelligence that aligns with global performance goals, bridging the gap between local optimization and holistic efficiency. Similarly, domains like healthcare, smart cities, and circular economy logistics benefit from distributed ML architectures that accommodate data heterogeneity, asynchronous decision cycles, and evolving coordination needs.

This dissertation builds upon these trends by developing and evaluating ML-enabled frameworks across six application domains. By combining SoS engineering principles with

supervised learning, reinforcement learning, optimization, and Natural Language Processing (NLP), the research demonstrates how intelligent subsystems can contribute to more adaptive, efficient, and resilient complex systems. These contributions address both domain-specific challenges (e.g., semiconductor yield, defect classification, healthcare coordination) and cross-cutting issues in decision support, scalability, and real-time responsiveness.

1.2 Problem Statement

Across sectors such as semiconductor manufacturing, healthcare, smart cities, and industrial quality control, organizations face increasing pressure to manage large-scale, decentralized systems with limited resources, dynamic environments, and diverse performance goals. These settings operate as *Systems of Systems* SoS, where constituent subsystems are independently managed yet expected to coordinate and contribute to shared outcomes. Despite progress in automation and digital transformation, existing approaches remain limited in their ability to scale coordination, respond to real-time events, or align local decisions with global objectives.

Conventional optimization and control techniques often treat subsystems in isolation, failing to model the interactions, tradeoffs, and emergent behaviors that arise in real-world SoS environments. Static rule sets, monolithic planning architectures, or hardcoded policies break down in the presence of uncertainty, evolving system goals, or competing stakeholder incentives. This is particularly evident in high-stakes domains such as semiconductor fabrication, where yield optimization, defect mitigation, and process coordination must occur simultaneously across multiple interdependent units.

ML offers promising alternatives, but its integration into SoS environments is uneven. Supervised learning models may improve local accuracy (e.g., defect classification or predictive maintenance), yet remain disconnected from broader system-level priorities. Reinforcement learning, while inherently suited to sequential decision-making, often lacks mechanisms for aligning agent-level rewards with global system outcomes. These gaps result in fragmented

implementations, redundant computation, suboptimal performance, and missed opportunities for cross-system synergy.

The core problem addressed in this research is the absence of a unified framework that can:

- Integrate diverse ML techniques with SoS coordination structures.
- Align subsystem behavior with system-wide optimization goals.
- Adapt to evolving conditions through data-driven learning and feedback.
- Generalize across domains with varying levels of structure, scale, and complexity.

This dissertation responds to this gap by proposing and evaluating domain-specific and cross-domain ML-SoS frameworks. These frameworks operationalize learning agents, coordination mechanisms, and reward alignment strategies within realistic simulation environments, enabling scalable and adaptive decision-making. The research focuses on evaluating how such frameworks perform under diverse use cases and operational constraints, while offering guidance for their broader application in critical infrastructure systems.

1.3 Research Objectives and Questions

This dissertation seeks to develop and evaluate machine learning (ML) frameworks that enable scalable, adaptive, and cooperative decision-making in complex System-of-Systems (SoS) environments. The overarching objective is to integrate domain-specific models with ML-driven coordination mechanisms that align local subsystem behavior with global optimization goals.

1.3.1 Primary Research Objectives

The research is guided by the following primary objectives:

1. To design ML-based architectures that support coordination, autonomy, and reward alignment in heterogeneous SoS environments.

2. To operationalize and simulate decision-making agents across multiple domains using reinforcement learning, supervised learning, and hybrid optimization techniques.
3. To evaluate the effectiveness of proposed ML-SoS frameworks across six real-world-inspired case studies, with a focus on yield, efficiency, adaptability, and conflict resolution.
4. To generalize the theoretical and methodological contributions across sectors characterized by decentralization, dynamic interactions, and resource constraints.

1.3.2 Case-Driven Research Questions

The study is structured around six domain-specific case studies, each addressing a focused set of research questions:

Case Study 1: 3D-IC Semiconductor Manufacturing

- **RQ1.1:** How can ML models (e.g., Random Forests, CNNs, LSTMs) improve subsystem-level outcomes such as defect detection, TSV optimization, and failure prediction?
- **RQ1.2:** Can a System-of-Systems (SoS) integration of these models improve global yield beyond isolated performance gains?

Case Study 2: Hierarchical MARL for SoS Coordination

- **RQ2.1:** How do hierarchical reward structures affect agent cooperation and system-level convergence in semiconductor SoS simulations?
- **RQ2.2:** What role does coordination logic play in reconciling conflicts between local policies and global objectives?

Case Study 3: SME Manufacturing Optimization

- **RQ3.1:** How can system dynamics modeling and predictive analytics improve decision-making in resource-constrained SME settings?

- **RQ3.2:** What trade-offs emerge between cost, inventory, and lead time under different optimization formulations?

Case Study 4: NLP for Quality Control in Electronics

- **RQ4.1:** Can text mining and Naïve Bayes models extract actionable insights from unstructured PCB defect logs?
- **RQ4.2:** How do these insights enhance traditional quality control systems in electronics manufacturing?

Case Study 5: AIoMT-Enabled Healthcare SoS

- **RQ5.1:** What coordination strategies emerge when wearable devices, hospitals, and remote services interact in a MARL-based healthcare SoS?
- **RQ5.2:** How do agent behaviors evolve in response to dynamic health events and resource constraints?

Case Study 6: Smart Cities

- **RQ6.1:** How can MARL-based coordination improve efficiency and responsiveness across traffic, energy, and public safety systems?
- **RQ6.2:** What challenges arise in balancing agent-level performance and system-wide goals in real-time smart city scenarios?

Case Study 7: Circular Economy Digital Marketplaces

- **RQ7.1:** How effective are AI models (e.g., ARIMA, LSTM, GBR) in forecasting supply-demand patterns and optimizing resource flows in CE marketplaces?
- **RQ7.2:** What role can reinforcement learning play in stabilizing prices and improving market efficiency under volatile conditions?

Together, these questions frame the scope of inquiry across theoretical, methodological, and application-specific dimensions. The answers are pursued through a combination of simulation, data-driven modeling, and cross-case synthesis.

1.4 Scope and Delimitations

This dissertation investigates the application of Machine Learning (ML) methods within complex System of Systems (SoS) environments, focusing on domains where decentralized coordination, resource constraints, and dynamic interactions play a critical role. The scope spans seven distinct case studies that reflect a diversity of real-world applications, including semiconductor manufacturing, healthcare systems, electronics quality control, smart cities, Small and Medium-sized Enterprises (SME) optimization, and circular economy marketplaces.

The core methodology emphasizes reinforcement learning, supervised learning, and predictive modeling frameworks tailored to specific domains. In particular, Multi-Agent Reinforcement Learning (MARL)and hierarchical coordination strategies are explored in detail for semiconductor, smart city, and healthcare case studies, where agent-based decision-making is both viable and necessary. Other case studies—such as SME optimization and text-based quality control—leverage predictive analytics, system dynamics modeling, or natural language processing.

Key inclusions in this research:

- Design and simulation of intelligent agents in hierarchical and cooperative settings.
- Evaluation of supervised, time-series, and reinforcement learning models across domains.
- Cross-domain analysis of yield, efficiency, adaptability, and coordination performance.
- System-of-Systems framing of multi-subsystem environments using theoretical constructs.

Key exclusions and delimitations:

- Physical-level modeling of semiconductor processes (e.g., photolithography, BEOL fabrication) is excluded.
- Human-centric behaviors such as technician decision-making and labor scheduling are not explicitly modeled.

- Circular economy case studies do not include agent-based simulation; instead, they evaluate forecasting and price optimization models using synthetic data.
- The scope is limited to simulated environments; real-time industrial deployment and human-in-the-loop feedback are left for future work.

The detailed methodology and simulation scope are presented in Chapter 4, while each case study provides further domain-specific boundaries and assumptions. This chapter serves to establish the high-level contours of the research and clarify its intended domain of contribution.

1.5 Contributions

This dissertation contributes to the advancement of intelligent system coordination in complex, data-driven environments by integrating ML methods with System of Systems (SoS) engineering principles. Contributions span theoretical formulation, methodological innovation, and practical applications across multiple domains.

Theoretical Contributions

- Development of a unified conceptual framework that integrates Multi-Agent Reinforcement Learning (MARL) with System of Systems (SoS) structures to address decentralized coordination and reward alignment challenges.
- Formalization of hierarchical reward structures that reconcile local agent objectives with global system performance, enabling scalable cooperation.
- Extension of SoS theory into ML-intensive domains such as healthcare and smart cities, offering novel perspectives on autonomy, emergent behavior, and inter-agent conflict resolution.

Methodological Contributions

- Design of a simulation-based experimental framework that supports multi-level analysis of policy convergence, reward dynamics, and system-level performance.
- Operationalization of agent behavior through reinforcement learning, predictive modeling, and supervised classification tailored to domain-specific needs.
- Introduction of a cross-domain performance evaluation protocol that integrates metrics such as coordination scores, global yield, and sensitivity to reward structure design.
- Use of synthetic yet realistic simulation environments to validate ML frameworks in contexts where real-world data is scarce or proprietary.

Domain-Specific Contributions

- In **semiconductor manufacturing**, a MARL-based architecture is shown to improve global yield by aligning defect detection, TSV process optimization, and electrical failure prediction across subsystems.
- In **electronics quality control**, NLP and Naïve Bayes models are applied to technician reports, improving insight extraction from unstructured data.
- In **SME manufacturing**, a hybrid framework combining predictive analytics and optimization logic supports planning and cost-efficiency in low-volume environments.
- In **healthcare systems**, the integration of AIoMT devices and hospital services within a MARL framework demonstrates enhanced cooperation and real-time adaptability.
- In **smart cities**, the use of MARL agents for infrastructure coordination improves transportation and energy efficiency under dynamic reward structures.
- In **circular economy marketplaces**, time-series forecasting and Q-learning models support demand prediction and price stabilization for sustainable resource exchange.

Collectively, these contributions provide a robust foundation for applying ML techniques to complex, real-world SoS challenges, advancing both theory and practice across multiple critical infrastructure domains.

1.6 Dissertation Organization

This dissertation is organized into eleven chapters, each building upon the previous to develop and evaluate a ML framework for optimizing complex, multi-domain System of Systems (SoS) environments. The structure follows a logical progression from problem definition through theoretical framing, methodology, empirical validation, and concluding synthesis:

- **Chapter 1 – Introduction:** Establishes the background, research problem, objectives, and scope. It introduces the importance of integrating ML and SoS concepts for intelligent decision-making in domains such as semiconductor manufacturing, healthcare, smart cities, and circular economy marketplaces.
- **Chapter 2 – Literature Review:** Reviews the academic foundations across SoS engineering, ML in complex systems, reinforcement learning in manufacturing, NLP for quality control, and AI applications in sustainable infrastructure and circular economy. Research gaps are identified to justify the proposed framework.
- **Chapter 3 – Theoretical Conceptual Framework:** Develops a unifying model for SoS coordination using hierarchical MARL, reward alignment, conflict resolution, and scalable design principles. This chapter provides the theoretical underpinnings for the methodology and simulations.
- **Chapter 4 – Methodology:** Details the research design, simulation setup, agent architecture, reward structures, sampling techniques, and performance evaluation strategies. It operationalizes the conceptual framework and connects it to the case studies.
- **Chapter 5 – Case Studies I: Semiconductor Manufacturing** presents three domain-specific case studies applying the ML-SoS framework in semiconductor contexts:
  - **3D-IC Manufacturing Optimization:** Integration of supervised learning (Random Forest, CNN, LSTM) for defect detection, TSV optimization, and electrical failure prediction.

  - **Hierarchical MARL for Semiconductor SoS:** Simulation of decentralized coordination using multi-agent learning and hierarchical reward systems.
  - **SME Manufacturing Optimization:** Hybrid modeling using system dynamics and linear programming to address planning, cost, and resource challenges in small-scale manufacturing.
- **Chapter 6 – Case Study II: NLP for Quality Control in Electronics:** Applies text analytics and Naïve Bayes classification to technician reports for Printed Circuit Board (PCB) testing, demonstrating the role of ML in human-in-the-loop industrial systems.
- **Chapter 7 – Case Study III: AIoMT-Enabled Healthcare SoS:** Models coordinated decision-making among hospitals, wearables, and telemedicine systems using MARL in a simulated System-of-Systems environment.
- **Chapter 8 – Case Study IV: Smart Cities:** Uses MARL-based SoS coordination to simulate policy learning and cooperation across transportation, energy, and emergency response infrastructure.
- **Chapter 9 – Case Study V: Circular Economy Marketplaces:** Evaluates AI-driven forecasting and reinforcement learning techniques for price stabilization, demand prediction, and resource optimization in digital CE platforms.
- **Chapter 10 – Cross-Domain Synthesis and Discussion:** Synthesizes key findings across all seven case studies, highlighting cross-domain insights, methodological generalization, and practical implications for future ML-SoS integration.
- **Chapter 11 – Conclusion and Future Work:** Summarizes the dissertation's contributions, restates how the research questions were addressed, and outlines future directions for theoretical expansion and real-world deployment.

This structure enables a progressive and cross-domain exploration of how machine learning can be leveraged to model, simulate, and optimize coordination and performance in complex and heterogeneous systems.

1.7 Chapter Summary

This chapter has introduced the context, motivation, and problem space for applying machine learning in complex System of Systems (SoS) environments. It outlined the research objectives and structured questions that guide the investigation, while defining the scope and identifying the intended contributions of this work.

The chapter emphasized the challenges of coordinating autonomous subsystems in high-stakes environments such as semiconductor manufacturing, healthcare, smart infrastructure, and sustainable marketplaces. In response, the research proposes an integrated ML-SoS framework grounded in theoretical and practical needs.

The organization of the dissertation has been presented, setting the stage for a deeper investigation into the academic foundations in the next chapter. **Chapter 2** will review existing work on SoS engineering, reinforcement learning, NLP for quality control, and AI in circular and smart systems, providing the theoretical background for the proposed framework.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents a comprehensive review of the literature at the intersection of Machine Learning (ML) and System of Systems (SoS) engineering, with a focus on their applications across domains such as semiconductor manufacturing, healthcare, smart cities, circular economy platforms, and industrial quality control. As modern infrastructures become increasingly complex, data-driven, and interdependent, the integration of ML techniques into SoS frameworks has emerged as a critical enabler of adaptive decision-making, scalable coordination, and system-wide optimization.

The purpose of this chapter is threefold. First, it surveys the evolution of SoS engineering concepts and coordination models, highlighting their relevance in domains where subsystems must operate autonomously yet coherently toward shared goals. Second, it examines the diverse range of ML techniques—including supervised learning, reinforcement learning, and natural language processing—that have been deployed to support optimization, prediction, and control in dynamic environments. Third, it identifies recurring challenges and cross-domain gaps that motivate the integrated frameworks proposed in this dissertation.

2.1.1 Structure of the Chapter

This chapter is organized into seven main sections. Section 2.2 introduces the foundational concepts of System-of-Systems (SoS) engineering, including its definitions, coordination models, and recent frameworks. Section 2.3 provides an overview of ML techniques relevant to complex systems, including supervised learning, reinforcement learning, and cross-domain applications. Sections 2.4 through 2.6 present domain-specific literature in manufacturing, natural language quality control, and smart city/circular economy settings.

Section 2.7 synthesizes cross-domain integration gaps and motivates the unified framework presented in Chapter 3, while Section 2.8 concludes with a summary and transition.

The chapter is organized into six major sections:

1. **Section 2.2** introduces the foundational principles of SoS engineering, including its defining characteristics, coordination models, implementation challenges, and recent theoretical frameworks such as WAVE, ROPE, and X-SEM.
2. **Section 2.3** reviews the application of ML techniques in complex systems. It includes supervised models (Random Forests, CNNs, LSTMs), reinforcement learning (Q-learning, MARL), and cross-domain use cases in manufacturing, healthcare, and smart infrastructure.
3. **Section 2.4** focuses specifically on ML applications in semiconductor manufacturing. It covers defect detection, yield optimization, resource planning for SMEs, and implementation barriers such as data quality, system integration, and real-time responsiveness.
4. **Section 2.5** discusses the use of natural language processing (NLP) for quality control in electronics manufacturing, including text mining applications, probabilistic classification with Naïve Bayes, and challenges in leveraging unstructured technician reports.
5. **Section 2.6** explores the role of ML in optimizing infrastructure in smart cities and enabling sustainable resource management in circular economy marketplaces. Emphasis is placed on the use of MARL, dynamic pricing, and AI-driven digital marketplaces.
6. **Section 2.7** synthesizes the cross-cutting gaps identified throughout the chapter. It discusses the limitations of current domain-specific solutions, the need for generalizable architectures, and how this dissertation addresses those needs through SoS-aligned, ML-enabled frameworks.

By drawing from both domain-specific innovations and cross-domain challenges, this chapter lays the conceptual foundation for the dissertation's central contribution: a unified,

scalable, and modular approach to ML-SoS integration for real-world decision-support in complex systems.

2.2 System-of-Systems Engineering

2.2.1 Defining SoS and Core Characteristics

System-of-Systems (SoS) refers to a class of complex systems composed of multiple, operationally and managerially independent constituent systems that collaborate to achieve higher-level goals unattainable by individual systems alone [1]. These systems exhibit distinct characteristics, including operational and managerial independence, evolutionary development, geographical distribution, and emergent behavior [1, 2]. The autonomy of each constituent system allows them to operate independently, while interdependence enables coordinated action toward shared objectives.

Modern engineering environments increasingly rely on SoS frameworks to manage dynamic, cross-disciplinary infrastructures. Domains such as healthcare, aerospace, transportation, and semiconductor manufacturing have embraced SoS architectures to accommodate interoperability, adaptability, and scalability in the face of real-time complexity. The Smart City and AIoMT healthcare domains, in particular, demonstrate how distributed systems—including energy grids, hospitals, and wearable technologies—interact dynamically and asynchronously within a larger SoS environment [3].

Emergence is a hallmark of SoS, where macro-level system behavior arises from the interactions of micro-level agents. This emergent behavior presents both opportunities and challenges, as it enables adaptive performance but complicates predictability and control [1]. Consequently, understanding and managing emergence is central to effective SoS engineering.

2.2.2 Coordination Models in SoS

SoS coordination models describe the extent of centralization and the relationships among constituent systems. Four canonical types—Directed, Acknowledged, Collaborative, and Virtual—are widely recognized in the literature [1].

- **Directed SoS** features centralized authority and clearly defined objectives. A central manager dictates roles and monitors compliance, making this model suitable for mission-critical domains such as defense and emergency response. However, it often limits flexibility and responsiveness in complex environments.
- **Acknowledged SoS** retains centralized oversight but allows more autonomy to constituent systems. Governance is enforced through contracts or shared objectives, enabling balance between alignment and flexibility. Applications in healthcare logistics and manufacturing scheduling have demonstrated the value of this structure [4].
- **Collaborative SoS** is based on mutual agreements, with no single authority. Constituents coordinate voluntarily to pursue aligned goals. This model enhances adaptability but can suffer from coordination inefficiencies and slower convergence.
- **Virtual SoS** lacks centralized control entirely. Systems join or leave dynamically, and cooperation is emergent rather than planned. Though highly flexible, Virtual SoS are particularly vulnerable to instability and coordination failure [5].

Coordination models directly influence system design choices, including communication structures, policy alignment, and reward mechanisms in reinforcement learning frameworks. The MARL-Healthcare and Smart City case studies explored in this dissertation demonstrate the practical implications of these models for multi-agent coordination and policy optimization.

2.2.3 Challenges in SoS: Scalability, Emergence, and Control

SoS design and deployment face enduring challenges in scalability, emergent behavior management, and decentralized control. As the number of constituent systems increases,

the complexity of coordination and learning grows exponentially. Traditional system design methods, such as the Vee model or monolithic MBSE, often fail to account for the dynamic reconfiguration and policy adaptation required in SoS [6, 7].

Scalability is especially problematic for reinforcement learning-based SoS architectures. In multi-agent environments, the joint state-action space grows combinatorially with the number of agents, making policy convergence computationally expensive and slow [8]. Moreover, non-stationarity arises as agents learn simultaneously, constantly altering the learning environment for each other.

Another challenge lies in emergent phenomena. While emergence can lead to innovative behaviors and adaptive robustness, it can also produce unpredictable or undesirable outcomes if not well-managed. In semiconductor manufacturing SoS, for instance, misaligned local optimizations—such as over-prioritizing yield or minimizing test time—may degrade overall system performance if coordination is absent or poorly regulated [4].

Effective SoS frameworks must reconcile the tension between local autonomy and global coherence. Designing reward structures, communication protocols, and governance mechanisms that enable such reconciliation is a central focus of recent literature and this dissertation's framework design.

2.2.4 Recent Frameworks: WAVE, ROPE, and X-SEM

To address the limitations of traditional SoS models, new frameworks have emerged that incorporate modularity, adaptability, and agent-oriented architectures. Three noteworthy paradigms are WAVE, ROPE, and X-SEM.

The **WAVE** model (Workflow, Agent, View, and Environment) emphasizes semantic interoperability and dynamic reconfiguration. It supports decentralized agent collaboration and leverages ontological descriptions for self-organizing behavior [9]. The model has shown promise in domains such as transportation SoS and Smart Cities, where evolving mission objectives necessitate runtime adaptation [10].

ROPE (Role-Oriented Programming Environment) facilitates coordination via dynamic role assignment and policy-based agent interaction. Originally developed for software engineering, ROPE is now adapted for policy-governed SoS coordination. It aligns well with MARL-based architectures that depend on role-dependent reward shaping and cooperative learning [11, 12].

X-SEM (Extended System Engineering Methodology) expands MBSE principles to account for emergent behaviors and heterogeneous system evolution. It incorporates mission-driven modularity, agent-specific models, and simulation-driven integration. Applications in smart city infrastructure and SoS healthcare coordination demonstrate X-SEM's utility in aligning operational autonomy with long-term system coherence [13, 14].

Together, these frameworks reflect a paradigm shift toward hybridized SoS design methodologies—those that blend traditional engineering formalisms with learning-driven, decentralized intelligence. The present dissertation extends this movement by integrating MARL, reward coordination, and hierarchical system modeling into a generalized SoS optimization architecture.

2.3 Machine Learning in Complex Systems

2.3.1 Overview of ML Techniques in SoS Contexts

Machine Learning (ML) has emerged as a cornerstone in the optimization and control of complex, dynamic systems, particularly those that exhibit the interdependencies and decentralized behaviors characteristic of System of Systems (SoS). Across domains such as semiconductor manufacturing, healthcare, and circular economy platforms, ML techniques have demonstrated utility in enhancing predictive performance, resource coordination, and system resilience under uncertainty.

Supervised learning models—including Random Forests, Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks—have shown particular promise in high-dimensional engineering applications. In the context of 3D-IC manufacturing, CNNs

and Random Forests have been employed for defect classification during bumped die inspection, effectively managing feature-rich image and test data to increase detection accuracy [15]. LSTMs further extend this capability by capturing temporal dependencies in electrical signal traces to detect post-assembly failures, improving reliability and enabling data-driven failure prediction in time-series contexts [16, 17].

In natural language-based applications, the Naïve Bayes classifier has been effectively deployed in printed circuit board (PCB) quality control to transform unstructured technician reports into actionable defect classifications [18]. The probabilistic nature of Naïve Bayes allows for efficient and interpretable classification in text-rich environments where labeled data is limited.

Reinforcement learning (RL), particularly in the form of Q-learning and Multi-Agent Reinforcement Learning (MARL), plays a vital role in supporting adaptive decision-making within decentralized SoS. In AIoMT-enabled healthcare networks, MARL agents coordinate across hospitals, wearable sensors, and telemedicine systems to optimize resource allocation and policy convergence under diverse SoS typologies—such as Directed, Collaborative, and Virtual systems [19]. In circular economy applications, Q-learning-based agents adjust dynamic pricing and supply-demand matching strategies in digital marketplaces, enabling sustainability under economic volatility [20].

In sum, the convergence of supervised learning, deep learning, and reinforcement learning within SoS environments has enabled cross-domain optimization at multiple levels—from localized fault detection to global policy alignment. The integration of these methods provides a foundation for scalable, intelligent decision-making across domains characterized by partial observability, heterogeneity, and emergent system behavior.

2.3.2 Supervised Learning in Engineering Systems

Random Forest and CNN for Defect Detection

Supervised learning methods play a pivotal role in defect detection within high-precision manufacturing environments. In 3D-IC manufacturing systems, the inspection of bumped die surfaces requires the simultaneous processing of visual and electrical test data. Traditional rule-based approaches struggle with complex noise patterns and high feature dimensionality, prompting the adoption of hybrid machine learning frameworks that leverage the strengths of both ensemble learning and deep learning models.

Random Forest Classifiers (RFCs) have proven particularly effective for defect detection tasks due to their robustness to overfitting, capacity to handle noisy data, and interpretability in high-dimensional feature spaces. In the bumped die inspection stage, RFCs trained on structured attributes—such as test coverage, probe pressure, and X-ray measurements—have shown high classification accuracy and improved generalization, especially when combined with class-balancing techniques like SMOTE [15].

To enhance spatial feature extraction, Convolutional Neural Networks (CNNs) have been integrated into the defect classification pipeline. CNNs are capable of learning spatial hierarchies of features from raw image data, thereby improving detection sensitivity and reducing reliance on manual feature engineering. The hybrid CNN-RF architecture used in 3D-IC inspection processes dynamically adjusts classification thresholds based on yield-optimized objectives and downstream process interactions. This configuration enables the system to maintain consistent detection performance across variable lot conditions and tool setups.

By combining the interpretability of RFCs with the representational power of CNNs, the defect detection framework not only achieves higher precision but also enhances its adaptability to emerging defect classes and evolving process parameters. This supervised learning integration exemplifies the broader potential of ensemble-deep hybrid models in engineering SoS environments where domain knowledge, data diversity, and real-time responsiveness converge.

LSTM for Failure Prediction

Long Short-Term Memory (LSTM) networks, a class of recurrent neural networks (RNNs), are particularly well-suited for modeling sequential dependencies in time-series data. In complex engineering and manufacturing systems, LSTMs enable the early detection of failures by capturing temporal trends, lag patterns, and non-linear interactions across multiple input streams.

In the context of 3D-IC manufacturing, LSTM models have been effectively deployed during the electrical testing phase to identify anomalies such as signal degradation, leakage currents, and transient behavior in voltage or current profiles. These models are trained on time-sequenced sensor readings from probe and final test stages, enabling real-time failure prediction prior to product shipment [15]. The predictive output is dynamically integrated with upstream defect detection and TSV process metrics, allowing for subsystem-level thresholds to be adjusted in alignment with global yield objectives.

Beyond manufacturing, LSTM architectures have also been applied in circular economy marketplaces for demand-supply forecasting and resource availability prediction. In these cases, LSTMs were used to model multivariate time-series data capturing economic indicators, user demand cycles, and inventory dynamics [20]. The temporal forecasting capability provided by LSTMs enhanced the responsiveness of AI-driven digital marketplaces, particularly in managing inventory allocation and mitigating price volatility.

These cross-domain implementations highlight the versatility of LSTM networks in failure and anomaly prediction. Their ability to maintain memory across time steps allows them to outperform traditional time-series models like ARIMA or feedforward neural networks, particularly in environments characterized by temporal noise, data gaps, and fluctuating operational conditions.

Naïve Bayes in Text Classification

Naïve Bayes classifiers have proven effective for text classification tasks in industrial quality control settings, particularly when working with sparse, high-dimensional data such as

technician reports and defect logs. Rooted in Bayes' theorem, the classifier assumes conditional independence among features, allowing for efficient computation even with limited training samples [21, 22].

In printed circuit board (PCB) testing and inspection, textual defect reports generated by technicians contain valuable contextual information about anomalies observed during assembly and testing. Our study applied a Naïve Bayes framework to classify these unstructured textual descriptions into structured defect categories, such as misalignment, solder bridging, or open circuits [18]. The classifier was trained on preprocessed text inputs, incorporating tokenization, lemmatization, and normalization steps based on foundational natural language processing (NLP) techniques [23, 24].

Empirical results demonstrated that the Naïve Bayes approach could rapidly and accurately classify real-world textual inputs in real time, with high efficiency and minimal data requirements. Its performance was particularly robust in noisy and variable-length text environments, where more complex models like deep neural networks often suffer from overfitting or require extensive labeled data [25, 26].

By integrating this classifier into a real-time quality control pipeline, the framework enabled proactive decision-making and reduced reliance on manual defect categorization. This approach not only streamlined the inspection process but also enhanced feedback loops to upstream manufacturing stages, contributing to improved yield and reduced rework rates.

2.3.3 Reinforcement Learning and MARL

Reinforcement Learning (RL) has emerged as a key paradigm for decision-making in complex, dynamic environments where agents learn optimal policies through interaction and feedback. In the context of complex systems and System-of-Systems (SoS), RL enables constituent agents to adaptively coordinate and optimize actions based on changing environmental conditions [27].

Traditional single-agent RL frameworks have been successfully applied to isolated components of smart cities, healthcare systems, and circular economy platforms. For instance, in healthcare, Q-learning has been used for adaptive resource allocation, patient monitoring, and decision support in AIoMT-enabled systems [19, 20]. Similarly, in circular economy marketplaces, RL techniques have stabilized real-time pricing and facilitated sustainable resource distribution in dynamic trading environments [20]. These studies underscore RL's capacity to support decentralized, goal-aligned decision-making under uncertainty.

Multi-Agent Reinforcement Learning (MARL) extends these capabilities by allowing multiple agents—each with local objectives and partial observability—to learn coordinated behaviors through shared environments. MARL frameworks are especially suited for SoS applications where inter-agent collaboration, reward alignment, and scalability are essential. In smart city applications, MARL algorithms such as Multi-Agent Actor–Critic (MA2C) and RegionSTLight have been used for traffic signal control and autonomous vehicle coordination, achieving improved safety, efficiency, and congestion reduction [28, 29].

However, the effectiveness of RL and MARL in SoS contexts hinges on the design of hierarchical control architectures and reward structures. Studies in 3D-IC manufacturing and AIoMT healthcare systems highlight the challenges of aligning local agent incentives with system-wide goals, managing non-stationary environments, and coordinating heterogeneous agents with varying levels of autonomy [15, 30].

To address these challenges, recent research has proposed hybrid approaches that combine MARL with model-based systems engineering (MBSE), digital twins, and structured policy hierarchies. These methods facilitate real-time adaptability, multi-scale coordination, and robust learning in distributed systems. Our work builds on this foundation by proposing a hierarchical MARL framework with tiered rewards to optimize cooperation, conflict resolution, and global performance in SoS environments.

2.3.4 ML across Domains: Manufacturing, Healthcare, Cities

Machine Learning (ML) has become a cornerstone of innovation across diverse sectors, enabling predictive analytics, adaptive control, and intelligent decision-making in increasingly complex environments. In System-of-Systems (SoS) contexts, ML facilitates localized autonomy while aligning subsystem behavior with global objectives, making it especially relevant for sectors such as manufacturing, healthcare, and smart cities.

In semiconductor manufacturing, ML techniques have been instrumental in improving yield, reducing defects, and optimizing process parameters. For instance, Convolutional Neural Networks (CNNs) and Random Forests have enhanced defect detection by integrating image and electrical data, while Long Short-Term Memory (LSTM) models support anomaly detection in electrical testing by modeling temporal dependencies in sensor data [15]. These models not only improve predictive accuracy but also contribute to system-wide optimization when embedded within coordinated SoS architectures.

In the healthcare domain, the integration of Artificial Intelligence of Medical Things (AIoMT) with ML has enabled real-time patient monitoring, early diagnosis, and personalized treatment. Supervised learning models, such as support vector machines and ensemble classifiers, have been widely used for disease classification. Meanwhile, Reinforcement Learning (RL), particularly Q-learning and Multi-Agent RL (MARL), has been applied to optimize resource allocation, adapt treatment plans, and coordinate autonomous agents across hospital networks [19, 20]. However, these implementations often lack a cohesive SoS framework to unify decision-making across diverse healthcare subsystems.

In smart city infrastructure, ML supports a range of applications from traffic signal control and public safety to energy distribution and environmental monitoring. MARL frameworks such as MA2C and RegionSTLight have been used for scalable, decentralized control of traffic and transit systems [28, 29]. Predictive models like Gradient Boosted Regression (GBR) and ARIMA have been applied to energy demand forecasting, while anomaly detection algorithms support fault identification in utility networks [31, 32].

Despite these advances, the cross-domain integration of ML in SoS remains underdeveloped. Most studies focus on domain-specific implementations, neglecting the interoperability and scalability challenges that arise in complex, multi-agent ecosystems. The convergence of SoS engineering with ML offers a pathway to bridge these gaps by providing the structural and architectural coherence needed for cross-domain generalizability. This research contributes to that goal by proposing ML-driven frameworks that unify diverse subsystems under a coordinated SoS architecture.

2.4 ML in Manufacturing Systems

2.4.1 Defect Detection and Yield Optimization

Recent advances in machine learning (ML) have significantly enhanced defect detection and yield optimization capabilities within semiconductor manufacturing systems, particularly in complex domains such as Three-Dimensional Integrated Circuits (3D-ICs). Traditional inspection techniques, such as Scanning Acoustic Microscopy (SAM) and X-ray imaging, though accurate, are constrained by their computational intensity and limited adaptability to diverse defect types [33]. In contrast, Convolutional Neural Networks (CNNs) have emerged as scalable alternatives, offering improved performance by integrating image data with electrical test outputs [34, 35].

To address imbalanced data distributions, hybrid approaches incorporating generative adversarial networks (GANs), semi-supervised techniques like TripletMatch, and knowledge-augmented classifiers have been proposed to bolster wafer-level defect detection accuracy [36, 37]. Additionally, Random Forest classifiers have demonstrated effectiveness in detecting surface and bonding defects during the bumped die attachment stage, enabling interpretable and low-latency predictions that are well suited for deployment in real-time quality control pipelines.

Yield optimization is further supported through machine learning applications in Through-Silicon Via (TSV) process tuning. CNN-driven adaptive weighting techniques dynamically

adjust manufacturing parameters in response to process variability, addressing limitations associated with rule-based and heuristic optimization approaches [38, 39]. Together, these strategies improve defect mitigation while enhancing the robustness of inter-layer connections in 3D-IC stacks.

Moreover, Long Short-Term Memory (LSTM) networks have been employed to detect electrical failures by modeling temporal dependencies in test signal data [16, 40]. When integrated within system-level frameworks, LSTM-based models complement earlier-stage inspection methods by identifying anomalies such as leakage currents and latent interconnect faults. This layered application of supervised learning enhances overall fault coverage and supports continuous improvements in yield and reliability.

2.4.2 Resource Optimization via Simulation and Metaheuristics

Resource efficiency remains a central concern for Small and Medium Enterprises (SMEs) in the semiconductor industry, where constrained budgets and production capacities necessitate agile and cost-effective optimization strategies. Traditional optimization methods such as rule-based heuristics and genetic algorithms, though effective in large-scale operations, often prove computationally burdensome for SMEs [41]. Recent literature emphasizes the role of system-dynamics modeling as a strategic tool for capturing nonlinear interactions across production, logistics, and procurement functions [42, 43].

Simulation-based approaches are frequently combined with metaheuristic methods to explore complex solution spaces. Dual annealing, for example, has been successfully applied to fine-tune lot sizing, scheduling, and tool allocation under constraints common to small-batch semiconductor production environments. Linear programming and bi-criterion optimization further contribute to decision support by balancing cost-efficiency against supply chain responsiveness [44, 45]. These models enable SMEs to mitigate disruptions and align resource utilization with market demand without incurring excessive operational overhead.

The integration of predictive analytics into these frameworks adds a layer of anticipatory intelligence. Applications of regression models, decision trees, and physical-statistical hybrids—such as TCAD (Technology Computer-Aided Design) combined with machine learning—facilitate accurate forecasting of production bottlenecks, equipment failures, and yield risks [46–48]. These insights support more agile responses to variability, ultimately reducing cost and improving throughput.

2.4.3 Challenges: Data, Integration, and Timeliness

Despite the promise of ML in semiconductor manufacturing, several implementation challenges remain prevalent in both high-tech fabs and SME contexts. A major barrier is data quality, as manufacturing environments often generate sparse, noisy, or inconsistently labeled datasets. This issue is especially pronounced in SME settings, where data infrastructure may be rudimentary or fragmented across legacy systems [47]. Additionally, class imbalance in defect detection datasets reduces model generalizability, requiring the use of augmentation techniques or synthetic data generation strategies [36].

System integration poses another significant challenge. Many ML models operate in isolation due to interoperability constraints between equipment, sensors, and databases. Without seamless data flows across production stages, real-time coordination and optimization become difficult to achieve. This is particularly limiting in contexts such as TSV tuning or final test verification, where upstream decisions directly influence downstream outcomes [49, 50].

Real-time deployment constraints further restrict the applicability of deep learning in production settings. While CNNs and LSTMs offer high accuracy, their inference latency and hardware requirements can hinder integration into inline quality assurance systems. Lightweight models such as Random Forests or regression-based predictors may offer more feasible alternatives in low-latency environments, though they often trade off accuracy for speed [41, 51].

Overall, the literature reflects a growing recognition of the need for practical, integrated ML frameworks that balance predictive power, interpretability, and resource efficiency. Addressing these challenges is critical for unlocking the full potential of ML in next-generation semiconductor manufacturing systems.

2.5 NLP and Text Analytics in Quality Control

2.5.1 Applications of Text Mining in Industrial Settings

The application of text mining in electronics manufacturing has garnered significant attention due to its potential to improve the efficiency, accuracy, and scalability of defect detection processes. Text analytics, when applied to operator reports and test logs, facilitates the transformation of unstructured narrative descriptions into structured formats that can be used for automated quality control. This is particularly valuable in environments like printed circuit board (PCB) assembly and packaged chip testing, where manual interpretation of textual data can be time-consuming and error-prone.

Several studies have demonstrated the utility of advanced text mining techniques in this context. For instance, context-aware text analytics have been employed to enhance error code recommendations by learning from historical defect descriptions [52]. Unsupervised variable selection techniques have improved the identification of relevant defect features from unstructured data [53], while deep learning models have achieved significant accuracy improvements in identifying failure modes from textual narratives [54].

Additionally, semantic framework technologies [55] and linguistic equation models [56] have been utilized to extract structured information from free-form descriptions, enabling more nuanced and actionable insights into root causes. These methodologies collectively underscore the potential of natural language processing (NLP) in supporting data-driven decision-making in industrial settings. A summary of key contributions from the literature is presented in Table 2.1.

Table 2.1: Summary of Key Studies in Text Analytics for Defect Detection

Study	Methodology	Key Findings
[52]	Context-aware text analytics	Improved error code recommendation
[53]	Unsupervised variable selection	Enhanced defect identification
[54]	Deep learning algorithms	Increased accuracy in defect detection
[55]	Semantic framework technology	Structured information extraction from defect descriptions
[56]	Linguistic equations	Improved characterization of defects

2.5.2 Text Analytics Models and Pipelines

Effective deployment of text analytics in manufacturing quality control requires robust methodologies for processing unstructured textual inputs and classifying defect types accurately. Central to this approach is the development of a preprocessing pipeline tailored for domain-specific language encountered in electronics testing reports. Common preprocessing steps include tokenization, normalization, lemmatization, and stop-word removal, which collectively transform raw textual inputs into structured representations suitable for downstream modeling [23].

In the proposed framework, the Naïve Bayes classifier is employed as the core model for defect classification. This probabilistic model assumes conditional independence among textual features, enabling efficient computation and robust performance even in high-dimensional text spaces [21]. Its simplicity and low data requirements make it well-suited for industrial applications, where labeled datasets may be limited. The classifier estimates the likelihood of a defect category given a set of textual features, facilitating real-time defect categorization in manufacturing environments.

The choice of Naïve Bayes aligns with prior work demonstrating its effectiveness in textual classification tasks and its integration with structured modeling frameworks. For instance,

earlier studies have highlighted the utility of similar probabilistic models for structured information extraction in manufacturing defect analysis [22, 24].

This methodology is further strengthened by parallels to real-time decision-support systems in other domains, such as the ANFIS-based NLP framework used for energy-efficient resource allocation in cloud computing [26]. Such examples validate the broader applicability of lightweight, interpretable machine learning models in dynamic, data-scarce environments. Accordingly, the proposed Naïve Bayes–based approach offers an interpretable and computationally tractable foundation for real-time quality control in electronics manufacturing.

2.5.3 Challenges in Leveraging Unstructured Data

While text analytics offers substantial promise for quality control in manufacturing, leveraging unstructured data introduces a distinct set of technical and operational challenges. One major limitation is the variability and inconsistency in how defects are described by human operators across different manufacturing stations or shifts. These free-form narratives often include non-standard abbreviations, typos, and domain-specific jargon, complicating efforts to standardize inputs for machine learning pipelines [53, 56].

Another key challenge is the scarcity of labeled data. In contrast to image-based defect detection, where annotated datasets are relatively abundant, textual data often lacks consistent labeling, particularly in real-time production environments. This limitation restricts the applicability of supervised learning models and hinders the generalization of trained classifiers across production settings. Semi-supervised and unsupervised learning methods may partially address this issue but are less interpretable and harder to deploy in operational settings with strict traceability requirements [25].

Integrating NLP systems with real-time defect detection pipelines also demands computational efficiency and low-latency processing. Traditional NLP models may struggle to meet these constraints, especially when deployed in edge computing environments or on legacy industrial infrastructure. Moreover, existing quality management systems are rarely designed

to handle unstructured inputs, creating a misalignment between defect reporting interfaces and analytics tools.

Data privacy and security further complicate NLP adoption in manufacturing. Operator-entered defect logs can sometimes include sensitive product or client information, necessitating anonymization techniques that may degrade the quality of semantic analysis. Finally, the lack of domain-specific NLP resources—such as annotated corpora, specialized lexicons, or pretrained models—slows down the development of tailored solutions for electronics manufacturing contexts [57].

Overall, addressing these challenges requires a multidisciplinary effort that combines domain expertise, NLP innovation, and systems integration. Future work must prioritize the development of lightweight, explainable models, along with standardized data practices to facilitate scalable and secure deployment of text analytics in real-time industrial settings.

2.6 ML in Smart Cities and Circular Economy

2.6.1 Smart City Infrastructure Optimization with MARL

Reinforcement Learning (RL), particularly Multi-Agent Reinforcement Learning (MARL), has demonstrated substantial promise in optimizing complex smart city infrastructure by enabling decentralized, adaptive decision-making. In urban transportation systems, MARL frameworks such as RegionSTLight decompose global traffic control into localized tasks, improving congestion management and travel efficiency [29, 58]. These approaches outperform static models by dynamically adapting to evolving traffic patterns through inter-agent communication and shared environmental feedback.

Similarly, the Multi-Agent Actor–Critic (MA2C) framework has been successfully employed in scenarios involving both autonomous and human-driven vehicles, enhancing traffic flow and safety through coordinated lane-change behaviors and collision avoidance strategies [28]. Beyond transportation, MARL has also been utilized to manage distributed energy systems. Algorithms like the intelligent probabilistic wavelet fuzzy neural network–deep RL

algorithm (IPWFNN-DRLA) allow microgrids to autonomously balance renewable energy inputs, battery storage, and real-time demand fluctuations [31].

Despite these advancements, MARL applications in smart cities frequently operate within siloed subsystems, lacking broader system-of-systems integration. These limitations constrain their scalability and ability to respond to interdependent urban challenges. This has motivated research toward hybrid frameworks that integrate MARL with structured systems engineering models to enhance cross-domain coordination and real-time adaptability in smart city environments.

2.6.2 Forecasting and Pricing in Circular Economy Marketplaces

Artificial Intelligence (AI) has also emerged as a critical enabler for circular economy (CE) platforms by improving resource forecasting, waste reduction, and pricing optimization. Digital marketplaces powered by machine learning facilitate the real-time matching of supply and demand, supporting the CE objectives of closing material loops and reducing environmental impact [59]. Predictive modeling techniques, such as gradient boosting and agent-based simulation, have been deployed to forecast material flows and optimize resource allocation strategies [60].

AI-driven dynamic pricing models allow CE platforms to adjust prices in response to real-time fluctuations in supply, demand, energy costs, and economic indicators, enhancing system efficiency and resilience to volatility [61, 62]. Blockchain-backed digital marketplaces further support transparency and secure transactions, reinforcing stakeholder trust and promoting decentralized participation [63, 64].

However, the real-time integration of these technologies with upstream industrial data and external macroeconomic variables remains underexplored. Most existing AI models focus on discrete resource flows or isolated sustainability metrics, lacking unified frameworks for large-scale CE coordination under fluctuating economic and policy conditions.

2.6.3 Scalability, Fairness, and Real-Time Constraints

Across both smart cities and circular economy contexts, several critical challenges persist in scaling ML-based optimization frameworks. Scalability remains a core limitation in MARL applications, particularly when expanding to city-wide infrastructure or multi-agent CE systems. Parameter-sharing and decentralized architectures offer partial solutions, yet real-world deployments still face computational bottlenecks and coordination overheads [65, 66].

Fairness and equity also present ongoing concerns, especially in AI-driven marketplaces. Algorithmic bias in pricing models or resource recommendations can exacerbate inequalities in access and participation. Ensuring ethical AI practices—including fairness, transparency, and accountability—has become imperative for sustainable AI deployment in CE applications [67].

Real-time responsiveness is another shared constraint. Whether managing urban traffic or forecasting surplus material availability, AI models must continuously adapt to dynamic inputs. This necessitates robust simulator environments, reliable data pipelines, and lightweight inference models capable of delivering decisions with minimal latency.

Addressing these cross-cutting challenges requires integrated ML frameworks that blend the adaptability of MARL with the structure of systems engineering and the ethical safeguards of explainable AI. Future research must prioritize modular, scalable architectures that facilitate dynamic coordination across agents, subsystems, and stakeholders within smart cities and circular economies.

2.7 Cross-Domain Gaps and Integration Opportunities

2.7.1 General Gaps Across Domains

Despite significant advances in applying Machine Learning (ML) across domains such as manufacturing, healthcare, smart cities, and circular economy systems, several persistent gaps challenge the deployment of ML in System-of-Systems (SoS) environments. These gaps span architectural, methodological, and operational dimensions.

One recurring limitation is the lack of coordination among constituent systems. In domains like 3D-IC manufacturing, localized ML models—for defect detection, parameter tuning, or failure prediction—often operate in silos, with minimal feedback or data sharing across stages [15]. Similarly, AIoMT-enabled healthcare systems face fragmentation between hospitals, telemedicine platforms, and wearable devices, undermining real-time decision-making and cross-entity resource allocation [20].

Another common issue is the difficulty in aligning local ML decisions with global objectives. For example, hierarchical reward design in semiconductor MARL systems aims to reconcile local autonomy with global efficiency, yet balancing this trade-off remains an open challenge. Circular economy platforms also exhibit this tension, as agents optimizing local pricing or resource matching often overlook system-wide sustainability goals [20].

Furthermore, domain-specific ML models are rarely generalizable. Text analytics in PCB manufacturing, for instance, requires unique preprocessing and classifiers that are not easily portable to healthcare or smart city data [18]. Even when similar ML models (e.g., CNNs, LSTMs, Q-learning) are used across domains, their integration into SoS frameworks remains context-dependent and bespoke, limiting scalability and reuse.

Lastly, few frameworks support comparative evaluation of SoS configurations (e.g., Directed vs. Collaborative), which constrains architectural learning and system-level optimization. This issue is especially critical in healthcare and smart city domains, where emergent behaviors depend heavily on coordination structure but remain empirically underexplored [5, 20].

2.7.2 Need for Unified Frameworks and Scalable Architectures

Addressing these cross-domain gaps requires a generalizable, scalable SoS framework that integrates ML decision-making across heterogeneous agents and subsystems. Several studies in this dissertation have highlighted the need for such integration:

In 3D-IC manufacturing, coordinated use of CNNs, Random Forests, and LSTMs achieved local optimization, but required a dynamic SoS loop to improve global yield and failure mitigation.

In Hierarchical Reinforcement Learning (HRL) simulations, global, main-agent, and sub-agent rewards were necessary to align decentralized decisions with high-level manufacturing goals.

For SMEs, scalable optimization demanded combining system dynamics, linear programming, and predictive analytics—an approach highly dependent on feedback loops and inter-agent coordination.

Each of these contributions reveals that local ML effectiveness is not sufficient; what matters is the ability to compose, coordinate, and adapt these ML-driven subsystems within a broader architectural logic. The lack of a unified modeling approach that bridges supervised, unsupervised, and reinforcement learning across constituent systems limits progress in both theory and practice.

Moreover, scalable ML-SoS architectures must be able to handle heterogeneity and partial observability, adapt to data-scarce environments (as in SMEs or low-resource health settings), and support hybrid governance (e.g., Directed vs. Collaborative coordination). This requirement reinforces the necessity of incorporating SoS principles such as emergent behavior management, modularity, and policy-driven design into ML pipelines.

2.7.3 Role of This Dissertation in Addressing Gaps

This dissertation addresses the aforementioned gaps by proposing and validating a set of integrated ML-driven frameworks that explicitly embed System-of-Systems (SoS) design principles across six domains:

In 3D-IC manufacturing, the dissertation contributes a unified optimization loop coordinating CNN-based defect detection, TSV optimization, and LSTM-driven electrical failure prediction, resulting in measurable yield improvements and system coherence.

The hierarchical RL case study introduces a three-level reward structure to manage decentralized agents in a semiconductor SoS, offering a novel framework for balancing local decision-making with global optimization.

In SME manufacturing, a lightweight, interpretable system combines simulation, linear programming, and predictive analytics, demonstrating ML-SoS integration under low-resource constraints.

For PCB quality control, the integration of human-entered reports into a real-time Naïve Bayes classifier exemplifies how unstructured data can enhance system-level quality management when linked to upstream/downstream subsystems.

In AIoMT healthcare, the study explores four canonical SoS models, evaluating trade-offs between autonomy and coordination via MARL, and providing design insights for scalable digital health infrastructures.

The smart city and circular economy chapters extend these lessons to urban-scale SoS, integrating MARL for traffic/energy management and reinforcement learning for price stabilization in CE marketplaces.

Across all six domains, the dissertation demonstrates that ML performance is maximized when embedded in feedback-rich, modular SoS architectures. It contributes novel frameworks, reward structures, and modeling methodologies that unify decision-making across distributed, heterogeneous agents. These contributions help advance the field by offering a blueprint for future cross-domain, ML-enabled SoS engineering.

2.8 Chapter Summary

This chapter presented a comprehensive literature review on the integration of Machine Learning (ML) and System-of-Systems (SoS) engineering across multiple application domains, including semiconductor manufacturing, healthcare, smart cities, circular economy platforms, and quality control in electronics. The review emphasized both domain-specific innovations and cross-domain challenges in applying ML techniques—such as supervised

learning, reinforcement learning, and natural language processing—within SoS environments characterized by decentralized coordination, partial observability, and emergent behavior.

The chapter began by defining core SoS principles and coordination models, highlighting the need for architectures that can support autonomous subsystem operation while maintaining global coherence. Subsequent sections reviewed the application of ML in complex systems, identifying key methods (e.g., CNNs, LSTMs, MARL) and their contributions to fault detection, resource optimization, policy learning, and text analytics across diverse industrial and urban contexts.

Across domains, several recurring challenges were identified: the siloed deployment of ML models, misalignment between local agent decisions and global system objectives, lack of generalizable integration frameworks, and limited evaluation of SoS typologies. These limitations underscore the need for unified, modular, and scalable ML-SoS architectures capable of dynamic coordination and cross-layer feedback.

The final section synthesized these insights into a set of cross-domain gaps and integration opportunities. It emphasized the importance of embedding ML subsystems within feedback-rich SoS frameworks and demonstrated how local predictive or control capabilities can be elevated through architectural integration, coordinated reward structures, and hybrid decision mechanisms.

This synthesis of cross-domain insights lays the foundation for the theoretical contributions presented in the following chapter. **Chapter 3** introduces a unified conceptual framework for ML-enabled System-of-Systems (SoS) optimization, offering the theoretical and architectural basis that informs the methodology and domain-specific implementations developed in the subsequent chapters.

Chapter 3

Theoretical Conceptual Framework

3.1 Introduction

3.1.1 Purpose of the Chapter

The purpose of this chapter is to establish the theoretical and conceptual foundations that guide this dissertation's integration of Machine Learning (ML), Multi-Agent Reinforcement Learning (MARL), and System-of-Systems (SoS) approaches. This framework provides a coherent lens for understanding the complex dynamics of subsystem coordination, hierarchical decision-making, and global yield optimization in semiconductor manufacturing and other interconnected domains.

By defining and synthesizing the constructs underpinning intelligent agent coordination, this chapter helps position the proposed solution within the broader landscape of interdisciplinary research. It bridges high-level theory and implementation logic, serving as a precursor to the methodology (Chapter 4) and empirical case studies (Chapters 5–9).

3.1.2 Relevance to the Research Problem

The framework introduced in this chapter directly addresses the research gaps identified in Chapter 2, especially in coordinating heterogeneous semiconductor subsystems using learning-based agents. It outlines a structured mechanism by which agent-level behaviors, when governed by carefully designed reward signals, can contribute to system-level objectives. This approach is particularly relevant for environments that exhibit distributed control, emergent behavior, and real-time constraints—core attributes of both SoS and complex manufacturing systems.

3.1.3 Structure of the Chapter

This chapter is organized into the following sections:

- **Philosophical Underpinnings and Paradigm Justification:** Discusses the epistemological and ontological stance of the research, justifying the choice of a pragmatic paradigm for addressing interdisciplinary, simulation-driven challenges.
- **Core Framework Elements:** Introduces the foundational concepts of Systems of Systems (SoS) and Multi-Agent Reinforcement Learning (MARL), with emphasis on autonomy, distributed control, and learning-based coordination.
- **Reward Structures and Hierarchical Coordination:** Defines the multi-level reward architecture and formalizes the relationship between agent-level (intrinsic) and system-level (extrinsic) objectives, including reward alignment strategies.
- **Conflict and Cooperation in SoS Agents:** Presents mechanisms for agent-level coordination, conflict detection, and resolution, along with metrics to evaluate policy alignment and convergence.
- **Variable Operationalization:** Details how agent states, actions, and rewards are represented in simulation, and how subsystem metrics aggregate into global optimization targets.
- **Performance Evaluation Logic:** Describes how effectiveness is assessed through reward trends, cooperation scores, sensitivity and conflict metrics.
- **Cross-Domain Framework Generalization:** Demonstrates how the framework scales across all six empirical case studies, supporting reuse and domain adaptability.
- **Summary and Transition:** Recaps the conceptual integration and transitions to Chapter 4, which details how the framework is implemented and validated.

Taken together, these sections define a robust and extensible framework that guides the dissertation's technical approach, simulation experiments, and empirical validation.

3.2 Philosophical Underpinnings and Paradigm Justification

This research adopts a pragmatic philosophical stance, aligning with the interdisciplinary and applied nature of the problem domain—semiconductor manufacturing optimization using machine learning within a System-of-Systems (SoS) framework. Pragmatism emphasizes practical solutions and actionable insights over rigid adherence to any single theoretical framework. This makes it especially well-suited for studies that integrate data-driven techniques, agent-based modeling, and real-time decision support systems.

The complexity of modern semiconductor manufacturing necessitates a methodological approach that accommodates multiple perspectives—engineering systems, machine learning, and operations research. Pragmatism supports this plurality by allowing the research to flexibly combine quantitative simulation with system-level reasoning, aligning well with the need to balance local subsystem performance with global fab-level optimization.

Moreover, the use of simulation-based analysis, reinforced by empirical performance metrics, aligns with pragmatic principles by prioritizing utility and applicability. Rather than seeking universal laws or purely theoretical contributions, this research focuses on developing and validating frameworks that are directly relevant to real-world manufacturing constraints, including defect detection, resource optimization, and coordination among interdependent subsystems.

In summary, the pragmatic paradigm enables the study to:

- Integrate multiple modeling approaches (e.g., machine learning, agent-based control, heuristic optimization).
- Evaluate research outcomes through performance improvements and trade-off analysis.
- Align local agent behaviors with system-wide goals without prescribing a singular ontological structure.

This paradigm is well-suited for exploring the emergent behaviors, adaptability, and performance implications of complex manufacturing systems guided by machine learning-based decision frameworks.

3.3 Core Framework Elements

3.3.1 Systems of Systems (SoS) in Manufacturing

Systems of Systems (SoS) in the context of semiconductor manufacturing refers to a paradigm wherein independently functioning subsystems—such as defect inspection, electrical testing, and process control—interact dynamically to achieve higher-order objectives. Unlike monolithic systems, SoS architectures emphasize autonomy, distributed decision-making, and emergent behavior. Each subsystem operates with its own management and goals, yet contributes to global outcomes such as yield optimization and resource efficiency.

Key characteristics of SoS relevant to manufacturing include:

- **Autonomy:** Subsystems such as fault detection and material handling can act independently while remaining loosely coordinated.
- **Emergent Behavior:** Collective behaviors arise from interactions among subsystems, such as yield shifts based on cumulative inspection policies or cross-subsystem load balancing.
- **Distributed Control:** No central controller dictates all actions; instead, coordination emerges through localized policies and inter-subsystem signaling.
- **Dynamic Interaction:** Subsystems adapt based on changes in upstream or downstream systems, with actions that reflect environmental, temporal, and operational contexts.

This decentralized and modular structure makes SoS an ideal fit for semiconductor fabs, which require scalable coordination, adaptive decision-making, and resilience to local disturbances.

3.3.2 Multi-Agent Reinforcement Learning (MARL)

Multi-Agent Reinforcement Learning (MARL) provides a computational framework for modeling and optimizing decision-making in environments involving multiple interacting agents. In the context of SoS manufacturing, MARL agents represent autonomous decision-makers within different subsystems, such as inspection units or yield optimizers. These agents learn through trial and error by interacting with the environment and adjusting their policies to maximize cumulative rewards.

Relevant concepts in MARL include:

- **Policy Learning:** Agents develop action-selection strategies (*policies*) to optimize their behavior under uncertainty and incomplete information.
- **Exploration vs. Exploitation:** MARL balances between exploring new strategies and exploiting known successful actions, which is crucial in environments with high variability like manufacturing.
- **Actor-Critic Architectures:** These models separate the decision-making process (actor) from the value estimation (critic), improving policy stability and convergence. Actor networks propose actions, while critic networks evaluate them based on long-term value.
- **Inter-Agent Interaction:** Agents must learn not only from their own experiences but also in the context of other agents' actions, requiring cooperative, competitive, or mixed strategy formulations.

By integrating MARL with SoS, the research aims to coordinate subsystem agents such that local improvements support global manufacturing objectives, enabling scalability, responsiveness, and robustness in complex operational settings.

3.4 Reward Structures and Hierarchical Coordination

3.4.1 Multi-Level Reward Architecture

In complex Systems of Systems (SoS), effective coordination relies on a reward architecture that supports both localized autonomy and global coherence. This study adopts a hierarchical reward design featuring both **intrinsic rewards** (agent-level) and **extrinsic rewards** (system-level).

- **Intrinsic Rewards** are tailored to each agent's specific operational role. For instance, a defect inspection agent may receive rewards for minimizing false negatives and improving throughput. These rewards incentivize localized performance and encourage learning policies optimized for domain-specific goals.
- **Extrinsic Rewards** are assigned based on aggregated outcomes that align with broader manufacturing objectives, such as yield improvement or system-level throughput. These are distributed top-down, often from a coordinating or global agent, to enforce alignment across the entire SoS.

The integration of intrinsic and extrinsic reward signals enables agents to optimize their local decisions while contributing to collective outcomes. This dual structure reduces the risk of local optima and promotes emergent cooperative behaviors across agents operating in different subsystems.

3.4.2 Global vs. Local Objectives

In SoS manufacturing environments, conflicts often arise between global and local objectives. An agent optimizing for local metrics may take actions that inadvertently degrade system-wide performance. For example, an inspection unit maximizing its own throughput could increase undetected defect rates, undermining downstream testing or yield control.

To mitigate such conflicts, the reward formulation includes weighted components that reconcile competing priorities. The global reward for a given subsystem R_j^{global} is computed

as:

$$R_j^{\text{global}} = w_j R^{\text{system}} + \sum_{j' \neq j} \gamma_{j,j'} \mathcal{I}_{j,j'} \tag{3.1}$$

where:

- R^{system} is the overall system-level reward.
- w_j is the weight assigned to the subsystem j's contribution to the global outcome.
- $\gamma_{j,j'}$ represents the influence or coordination weight between subsystem j and other subsystems j'.
- $\mathcal{I}_{j,j'}$ is an interaction metric quantifying coordination or conflict.

This formulation ensures that local agents not only consider their own rewards but are also influenced by their impact on system-wide objectives and inter-agent relationships. By carefully tuning the weights w_j and coordination parameters $\gamma_{j,j'}$, the system promotes behaviors that optimize both individual and collective performance.

This reward alignment strategy underpins the hierarchical coordination mechanism, guiding agents toward joint optimization in the face of competing objectives and uncertain environments.

3.5 Reward Structure and Mathematical Formulation

This section formalizes the hierarchical reward mechanisms underlying the MARL-based SoS coordination framework. The reward structure is designed to support multi-level alignment, encompassing sub-agent actions, subsystem contributions, and global system optimization. The following mathematical models represent the decomposition of rewards, cooperation incentives, conflict resolution, and policy evaluation processes.

3.5.1 Global SoS Reward Formulation

The global Systems-of-Systems (SoS) reward aggregates contributions from individual subsystems and their cooperation dynamics. It is mathematically defined as:

$$R_{\text{global}} = \sum_{j=1}^{M} w_j (R_{j,\text{intrinsic}} + r_{j,\text{inter}}) \tag{3.2}$$

where M is the total number of constituent systems, w_j represents the assigned importance weight for subsystem j, $R_{j,\text{intrinsic}}$ denotes the intrinsic reward obtained by subsystem j, and $r_{j,\text{inter}}$ captures the additional reward resulting from inter-agent cooperation among different subsystems.

Inter-Agent Cooperation Reward

Subsystem cooperation is modeled through an inter-agent reward function, capturing the interaction between different constituent systems:

$$r_{j,\text{inter}} = \sum_{j' \neq j} \gamma_{j,j'} I_{j,j'} \tag{3.3}$$

where $\gamma_{j,j'}$ represents the cooperation weight between subsystems j and j', and $I_{j,j'}$ is an interaction metric quantifying the level of coordination achieved.

Conflict Resolution Reward

To ensure fairness and minimize imbalance among subsystems, a conflict resolution reward term is incorporated:

$$R_{\text{conflict}} = \delta \sum_{j=1}^{M} \sum_{j' \neq j} |R_{j,\text{total}} - R_{j',\text{total}}| \tag{3.4}$$

where δ is a positive scaling factor that determines the strength of penalization based on reward disparities across subsystems.

Total Global Reward

The overall global reward combines the intrinsic rewards and conflict penalties:

$$R_{\text{global,total}} = R_{\text{global}} + R_{\text{conflict}} \tag{3.5}$$

This formulation ensures that the system rewards both collective success and fairness across all constituent systems.

3.5.2 Main Agent Reward Formulation

Each main agent corresponding to a subsystem optimizes a total reward that balances local objectives and contributions to the overall SoS:

$$R_{j,\text{total}} = R_{j,\text{intrinsic}} + R_{j,\text{global}} \tag{3.6}$$

where $R_{j,\text{global}}$ represents the subsystem's contribution to the global system performance objective.

3.5.3 Sub-Agent Reward Formulation

Each sub-agent within a subsystem seeks to maximize a comprehensive reward that considers local, subsystem, global, and coordination aspects:

$$r_{k,j,\text{total}} = r_{k,j,\text{local}} + r_{k,j,\text{subsystem}} + r_{k,j,\text{global}} + r_{k,j,\text{coordination}} \tag{3.7}$$

where $r_{k,j,\text{local}}$ is the local task reward, $r_{k,j,\text{subsystem}}$ captures subsystem-level contribution, $r_{k,j,\text{global}}$ aligns actions with system goals, and $r_{k,j,\text{coordination}}$ rewards collaboration with peer agents.

Sub-Agent Coordination Reward

The coordination reward for sub-agents encourages peer collaboration:

$$r_{k,j,\text{coordination}} = \alpha_k \sum_{k' \neq k} C_{k,k'} \tag{3.8}$$

where α_k is the coordination weight and $C_{k,k'}$ measures the degree of cooperation between agents.

Sub-Agent Global Contribution

The contribution of a sub-agent to system-wide goals is captured as:

$$r_{k,j,\text{global}} = \lambda_{k,j} R_{\text{global}} \tag{3.9}$$

where $\lambda_{k,j}$ is the alignment coefficient reflecting how strongly sub-agent k is tied to global objectives.

3.5.4 Reward Feedback and Alignment

A dynamic feedback adjustment ensures sub-agent rewards align progressively with system-wide goals:

$$\Delta r_{k,j} = \eta \left(R^{\text{global}} - r_{k,j}^{\text{total}} \right) \tag{3.10}$$

where η is the feedback sensitivity parameter.

3.5.5 Global Reward Decomposition

The global reward can be decomposed explicitly into local and cooperation-based contributions:

$$R^{\text{global}} = \sum_{j=1}^{M} w_j \sum_{k=1}^{n_j} \left(r_{k,j}^{\text{local}} + r_{k,j}^{\text{cooperation}} \right) + \sum_{j=1}^{M} r_j^{\text{inter}} \tag{3.11}$$

where n_j is the number of sub-agents in subsystem j and $r_{k,j}^{\text{cooperation}}$ captures cooperation rewards at the agent level.

3.5.6 Policy Gradient and Value Functions

Policy Gradient

The policy gradient method optimizes the expected return by directly adjusting the policy parameters. It is mathematically represented as:

$$\nabla_{\theta_k} J_k(\theta_k) = \mathbb{E}_{\pi_k} \left[\nabla_{\theta_k} \log \pi_k(a_k \mid s_k; \theta_k) A_k(s_k, a_k) \right] \tag{3.12}$$

In this formulation, θ_k denotes the trainable policy parameters for agent k, $J_k(\theta_k)$ represents the expected return objective, $\pi_k(a_k \mid s_k; \theta_k)$ defines the probability of taking action a_k in state s_k under the policy, and $A_k(s_k, a_k)$ is the advantage function quantifying the relative benefit of selecting action a_k.

State-Value Function

The state-value function estimates the expected cumulative discounted reward when starting from a given state s_k and following the policy π_k, and is defined as:

$$V_k(s_k) = \mathbb{E}_{\pi_k} \left[\sum_{t=0}^{\infty} \gamma^t r_{k,j}^t \mid s_k^0 = s_k \right] \tag{3.13}$$

Here, $V_k(s_k)$ denotes the expected discounted cumulative reward from state s_k, γ is the discount factor ($0 < \gamma \leq 1$) determining the importance of future rewards, $r_{k,j}^t$ is the reward received by agent k at time step t for subsystem j, and s_k^0 indicates the initial state at $t = 0$.

Action-Value Function

The action-value function extends the concept of the state-value function by explicitly incorporating the immediate action taken. It computes the expected return after executing action a_k in state s_k and thereafter following policy π_k, given as:

$$Q_k(s_k, a_k) = \mathbb{E}_{\pi_k} \left[r_k(s_k, a_k) + \gamma V_k(s_k') \right] \tag{3.14}$$

In this expression, $Q_k(s_k, a_k)$ denotes the expected cumulative reward when taking action a_k in state s_k, $r_k(s_k, a_k)$ is the immediate reward obtained, s'_k is the next state resulting from action a_k, and $V_k(s'_k)$ is the value of the next state under policy π_k.

Advantage Function

The advantage function measures the relative merit of a specific action compared to the average expected value of the state. It is formulated as:

$$A_k(s_k, a_k) = Q_k(s_k, a_k) - V_k(s_k) \tag{3.15}$$

Here, $A_k(s_k, a_k)$ represents the advantage gained by taking action a_k compared to the average expected return from state s_k, where $Q_k(s_k, a_k)$ and $V_k(s_k)$ are the corresponding action-value and state-value functions.

Together, these foundational mathematical formulations define the mechanisms of reward propagation, policy optimization, and decentralized decision-making that underpin the hierarchical SoS–MARL framework. They enable agents to make adaptive, coherent decisions across large-scale, complex system-of-systems environments.

3.6 Conflict and Cooperation in SoS Agents

3.6.1 Coordination Mechanisms

In a Systems of Systems (SoS) environment, agents must interact dynamically to achieve aligned outcomes while maintaining autonomy. Coordination is essential to prevent inefficiencies such as resource contention, task duplication, or inconsistent responses to process anomalies.

Agents coordinate both **vertically** (across hierarchical levels, e.g., sub-agent to main-agent) and **horizontally** (across subsystems, e.g., Defect Inspection Agent coordinating with Electrical Testing Agent). For example, if the Inspection Agent flags a wafer as likely

defective, the Electrical Testing Agent can prioritize those units for high-resolution probing. This minimizes test redundancy and maximizes defect coverage.

Coordination is often achieved through:

- **Shared State Representations:** Agents incorporate information from related agents to inform local decisions.
- **Message Passing:** Structured communication allows agents to broadcast or query states, predictions, or intentions.
- **Reward Shaping:** Rewards incorporate coordination success metrics to encourage synergy across agents.

These mechanisms encourage distributed but aligned behaviors without requiring complete centralization, preserving scalability and modularity within the SoS.

3.6.2 Conflict Resolution Modeling

Despite cooperative design, conflicting behaviors may emerge due to overlapping objectives, asynchronous states, or reward misalignment. For example, a Maintenance Agent prioritizing preventive servicing may schedule downtime during peak production, directly conflicting with throughput goals of an Optimization Agent.

To monitor and resolve such conflicts, the framework implements a **Conflict Resolution Model** using:

- **Conflict Metrics:** Defined as divergence between predicted agent actions and ideal cooperative outcomes. One such metric is:

$$\Delta_{\text{conflict}}(j, j') = \|a_j - a^*_{j'}\| \tag{3.16}$$

where a_j is the action of agent j, and $a^*_{j'}$ is the optimal action of a cooperating agent j' in a shared context.

- **Resolution Logic:** When conflicts exceed a predefined threshold τ, agents adjust their action policies using coordination weights:

$$a_j^{(t+1)} = a_j^{(t)} + \delta_{j,j'} \cdot \left(a_{j'}^{(t)} - a_j^{(t)}\right) \tag{3.17}$$

  where $\delta_{j,j'} \in [0, 1]$ is the adaptability factor between agents.

- **Policy Convergence Checks:** Agents periodically evaluate the stability of coordination using entropy or KL-divergence measures between successive policy distributions. This ensures learning processes trend toward consistent joint behavior rather than oscillating strategies.

These conflict resolution mechanisms ensure that agents not only learn optimal local actions but also adjust adaptively in shared environments. This balance between autonomy and cooperation is fundamental to achieving system-wide harmony in decentralized SoS architectures.

3.7 Variable Operationalization

3.7.1 Agent States, Actions, and Rewards

In the proposed SoS-MARL framework, each agent operates based on a structured set of variables—states, actions, and rewards—that are defined with respect to its role in the semiconductor manufacturing process.

- **States ($s_{k,j}$):** Represent the observable environment from the perspective of sub-agent k within subsystem j. Examples include:
  - Defect density and resolution history (for Inspection Agent)
  - Tool readiness and calibration status (for Maintenance Agent)
  - Inventory level and backlog queues (for Inventory Agent)

- **Actions ($a_{k,j}$):** Encoded as discrete or continuous decisions derived from the state space. For instance:
  - Adjusting scanning resolution (Inspection Agent)
  - Scheduling repair or deferring maintenance (Maintenance Agent)
  - Reordering inventory (Inventory Agent)
- **Rewards ($r_{k,j}$):** Generated based on task-specific performance, incorporating intrinsic and extrinsic signals:

$$r_{k,j}^{\text{total}} = r_{k,j}^{\text{local}} + r_{k,j}^{\text{subsystem}} + r_{k,j}^{\text{global}} \tag{3.18}$$

  These terms capture performance at the individual agent level, the subsystem level (e.g., coordination among agents within a main system), and the system-of-systems (SoS) global coordination layer.

These elements are encoded in simulation using structured state vectors and continuous reward functions, with values constrained by predefined operational ranges.

3.7.2 Metric Alignment Across Scales

To ensure that agent-level performance contributes meaningfully to fab-wide goals, the framework introduces an aggregation mechanism that aligns local rewards with global metrics.

Subsystem-Level Aggregation: Each main agent j aggregates sub-agent performance via:

$$R_j^{\text{intrinsic}} = \sum_{k=1}^{n_j} \left(r_{k,j}^{\text{local}} + r_{k,j}^{\text{subsystem}} \right) \tag{3.19}$$

Global Reward Composition: Subsystem rewards contribute to the overall SoS performance through weighted aggregation:

$$R^{\text{global}} = \sum_{j=1}^{M} w_j R_j^{\text{total}} + \sum_{j \neq j'} \gamma_{j,j'} \cdot \mathcal{I}_{j,j'} \tag{3.20}$$

where:

- w_j is the weight assigned to each main agent based on criticality,
- $\gamma_{j,j'}$ denotes the cooperation weight between subsystems,
- $\mathcal{I}_{j,j'}$ captures interaction efficiency between agents j and j'.

This design ensures upward compatibility, allowing localized metrics such as defect resolution efficiency, test redundancy, or inventory holding cost to scale into comprehensive system-wide metrics like throughput, yield, and resource efficiency. The hierarchical reward structure is thus tightly aligned with both operational and strategic optimization objectives.

3.8 Performance Evaluation Logic

To assess the effectiveness of the proposed Systems of Systems (SoS) coordination and Multi-Agent Reinforcement Learning (MARL) framework, a multi-layered evaluation logic is employed. This includes both temporal performance metrics and structural indicators of coordination and adaptability.

- **Temporal Reward Trends:** The convergence and stability of reward signals over time are tracked at the agent, subsystem, and global levels. Metrics such as cumulative reward, reward volatility, and reward improvement rate are visualized and analyzed to assess learning stability.
- **Cooperation and Conflict Metrics:** Intra- and inter-agent cooperation is quantified through interaction metrics ($\mathcal{I}_{j,j'}$), coordination weights (α_k), and conflict resolution indicators (δ). These reflect the degree of alignment and friction between agent policies.
- **Policy Convergence and Sensitivity:** Evaluation includes the analysis of policy convergence patterns using loss trends and entropy metrics, as well as sensitivity testing across environmental parameters (e.g., defect rate fluctuations, tool readiness shifts).

- **Global Optimization Alignment:** The alignment between local optimizations and global objectives is assessed by comparing subsystem reward contributions (R_j^{total}) with aggregated system performance (R^{global}).

Together, these metrics provide a robust framework to evaluate how well the hierarchical SoS-MARL architecture supports real-time decision-making, resource allocation, and yield improvement in simulated manufacturing environments.

3.9 Cross-Domain Framework Generalization

While this conceptual framework is instantiated in the semiconductor manufacturing domain, its structure is intentionally designed for cross-domain generalization. Each case study in Chapters 5 through 9 implements a variation of this SoS-MARL model adapted to the domain-specific characteristics.

Table 3.1: Generalization of Conceptual Framework Across Domains

Domain	Subsystems/Agents	Optimization Objective
3D-IC Manufacturing	Defect Detection, TSV Optimization, Electrical Testing	Global yield improvement via subsystem coordination
Semiconductor SoS (HRL Simulation)	Wafer Fabrication, Defect Inspection, Maintenance	Dynamic reward alignment and adaptive coordination
SME Manufacturing Optimization	Inventory, Scheduling, Resource Planning	Low-volume efficiency and cost-based planning
PCB Defect Analysis (NLP)	Text Parsing, Defect Classification, Resolution Routing	Quality improvement through unstructured text processing
AIoMT-Enabled Healthcare SoS	Remote Monitoring, Diagnosis, Emergency Response	Patient-centric triage and resource prioritization
Smart Cities and Circular Economy	Mobility, Energy, Waste, Pricing Agents	Fairness, sustainability, and cooperative market coordination

Across these domains, the framework maintains the following invariants:

- Hierarchical agent roles with intrinsic and extrinsic reward coupling.
- SoS coordination with emergent agent behaviors and decentralized control.
- Performance evaluation using both quantitative metrics and qualitative cooperation indicators.

This generalization reinforces the utility of the framework beyond a single domain and underscores its contribution to interdisciplinary applications of machine learning in complex, dynamic environments.

3.10 Chapter Summary

This chapter introduced the conceptual framework guiding the integration of Systems of Systems (SoS) engineering and Multi-Agent Reinforcement Learning (MARL) for coordinated decision-making in semiconductor manufacturing and other complex domains. Anchored in a pragmatist paradigm, the framework is tailored to address the interdisciplinary, simulation-driven nature of the research.

Key contributions of the framework include:

- A structured SoS-MARL architecture that supports decentralized agent coordination, hierarchical decision-making, and alignment between local and global objectives through a multi-level reward system.
- Operationalization of agent states, actions, and rewards, alongside coordination and conflict resolution mechanisms across subsystem boundaries.
- Robust performance evaluation logic based on reward trends, cooperation scores, and sensitivity metrics.
- Demonstrated potential for cross-domain generalization, validated through six diverse case studies.

By formalizing the theoretical underpinnings and structural elements of the SoS-MARL model, this chapter provides a blueprint for implementation. The next **Chapter 4**, builds upon this foundation by detailing the research design, simulation architecture, data generation, and validation strategies used to operationalize and evaluate the proposed framework.

Chapter 4

Methodology

4.1 Introduction

This chapter outlines the methodological framework developed to operationalize the conceptual foundations established in Chapter 3. The goal is to design, implement, and evaluate a Systems of Systems (SoS) model driven by Multi-Agent Reinforcement Learning (MARL) in simulated semiconductor manufacturing environments. The methodology integrates simulation-based experimentation with hierarchical agent coordination to address the research questions outlined in Chapter 1.

To provide a cohesive research structure, this methodology chapter serves as the central blueprint connecting theoretical foundations (Chapter 3) to empirical implementation (Chapters 5–9). The research process follows a six-phase flow:

- **Step 1 – Conceptual Design:** Define the hierarchical SoS-MARL architecture, agent roles, and reward hierarchy (Chapter 3).
- **Step 2 – Methodology:** Develop simulation logic, hypothesis structure, agent categories, and mixed-methods strategy (Chapter 4).
- **Step 3 – Case Studies:** Apply the framework to seven domain-specific case studies using tailored algorithms (Chapters 5–9).
- **Step 4 – Simulation:** Run controlled experiments with agents under varying scenarios and collect structured data (Section 4.7).
- **Step 5 – Evaluation:** Perform quantitative and qualitative assessment at multiple levels (Section 4.8).
- **Step 6 – Synthesis:** Generalize findings across domains, identify trade-offs, and inform future research (Chapters 10 and 11).

This structured process ensures internal consistency while supporting cross-domain adaptation of the proposed ML–SoS framework.

The six-step research flow is visually summarized in Figure 4.1.

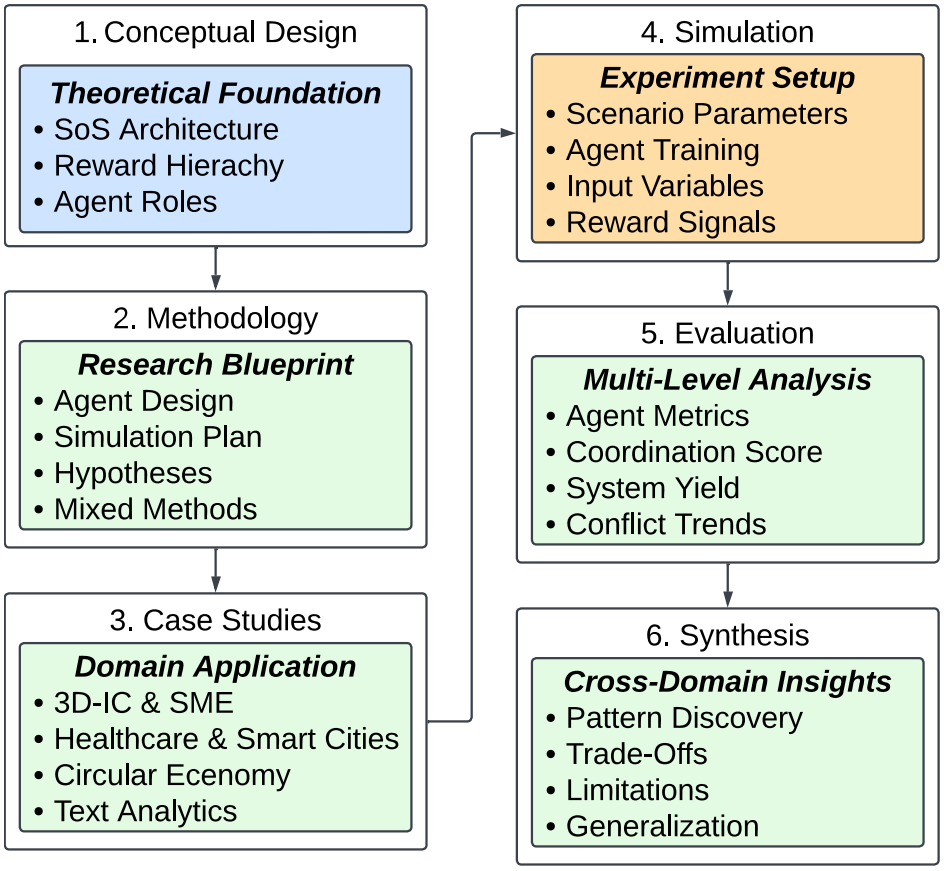

Figure 4.1: End-to-End Research Methodology: From Conceptual Design to Cross-Domain Synthesis

As illustrated in Figure 4.1, the research methodology follows a structured six-phase flow. It begins with conceptual design and methodological planning (Chapters 3 and 4), proceeds through case-specific implementation and simulation (Chapters 5–9), and culminates in multi-level evaluation and cross-domain synthesis (Chapters 10 and 11). This diagram provides an integrated overview of how theoretical modeling, empirical validation, and simulation-based insights are connected throughout the dissertation.

4.1.1 Purpose

The primary purpose of this chapter is to detail the structured, step-by-step methodology used to evaluate the proposed SoS-MARL framework. Specifically, it describes how the research questions are addressed through:

- A hierarchical MARL architecture simulating real-time coordination among autonomous agents.
- Subsystem-specific modeling aligned with SoS design principles.
- Quantitative and qualitative evaluation across local and global performance metrics.

4.1.2 Justification

The choice of a simulation-driven MARL methodology is justified by the need to explore coordination dynamics in complex, interdependent systems where real-world experimentation is infeasible or costly. Key justifications include:

- MARL supports decentralized decision-making and emergent behavior suitable for modeling SoS environments.
- A hierarchical reward system enables the reconciliation of local agent objectives with global fab-level performance.
- Simulation environments allow for controlled experimentation with synthetic data and structured scenario variations.

4.1.3 Scope

The methodology presented in this dissertation spans multiple domains where Machine Learning (ML) and System-of-Systems (SoS) coordination are applied to optimize decentralized, dynamic, and resource-constrained environments. The core simulation framework and agent-based reinforcement learning model are developed in the context of semiconductor manufacturing, where constituent systems are classified as:

- **Directed Systems:** Subsystems with pre-defined operational goals, including defect inspection and electrical testing.
- **Acknowledged Systems:** Autonomous modules such as scheduling and inventory control, operating with minimal coordination.
- **Collaborative Systems:** Maintenance and coordination agents requiring cooperation for shared resource optimization.
- **Virtual Systems:** Simulation, scenario planning, and adaptive policy exploration mechanisms supporting learning-based decision logic.

This agent-based, hierarchical reinforcement learning approach forms the methodological backbone of the semiconductor and smart city case studies, as well as the AIoMT-enabled healthcare simulation. However, the methodology also includes non-agent-based components appropriate to the context of each domain. Specifically:

- **Semiconductor Manufacturing:** A fully simulated SoS environment with MARL agents optimizing yield and defect mitigation through coordinated decision-making.
- **Healthcare SoS:** Simulated coordination across hospitals and medical devices using MARL and subsystem categorization similar to semiconductor modeling.
- **Smart Cities:** Application of MARL-based SoS coordination to optimize infrastructure systems such as traffic, energy, and public safety.
- **SME Manufacturing Optimization:** A non-agent-based framework using linear programming and predictive analytics to support planning and cost optimization in small-scale production environments.
- **Text Analytics for Quality Control:** Use of Naïve Bayes classification and NLP-based preprocessing to extract structured insights from technician reports in PCB testing; this case does not involve agents or reinforcement learning.
- **Circular Economy Marketplaces:** A forecasting and decision-support model utilizing LSTM, ARIMA, and gradient boosting regressors. No agent-based architecture is used in this case study.

The methodology does not include physical-level process simulation such as photolithography, TSV formation, or BEOL integration. Nor does it model upstream supply chains, workforce dynamics, or environmental variables. The scope is bounded to simulated and computationally tractable environments, emphasizing coordination, adaptability, and learning performance within domain-relevant abstractions.

To enhance transparency in model selection across heterogeneous case studies, Table 4.1 summarizes the primary AI algorithms employed, along with selection rationale tailored to each domain context.

Table 4.1: AI Algorithm Selection by Case Study and Rationale

Case Study	Algorithm(s)	Selection Rationale
Semiconductor – Defect Detection	Random Forest, SVM, XGBoost	RF chosen for high precision/recall and low inference latency
Semiconductor – TSV Optimization	CNN + Dual Annealing	CNN used for adaptive feature weighting under real-time constraints
Semiconductor – Fault Detection	LSTM	Captures long-term dependencies in test signal sequences
Smart City Optimization	PPO (MARL)	Supports policy learning in high-dimensional, dynamic environments
Healthcare SoS	Q-learning (MARL)	Simple, tunable agent coordination in simulated SoS topologies
SME Manufacturing	LP + Predictive Analytics	Balances optimization with explainability in low-volume settings
NLP for Quality Control	Naïve Bayes + Text Preprocessing	Chosen for interpretability and fast training on technician logs
Circular Economy	LSTM, ARIMA, GBR, Q-learning	Forecasting and dynamic pricing under stochastic demand patterns

In addition to domain fit, AI algorithm selection was also influenced by practical design constraints such as latency, explainability, and coordination complexity. Table 4.2 summarizes these factors and their alignment with model characteristics.

Table 4.2: AI Algorithm Selection Mapped to Design Constraints

Constraint Type	**Preferred Algorithms**	**Rationale and Use Cases**
Low Latency / Real-Time Inference	Random Forest, Naïve Bayes, Q-learning	Fast prediction for inspection, fault detection, and real-time coordination
Interpretability	Naïve Bayes, Decision Trees, LP	Human-in-the-loop analysis in PCB and SME manufacturing environments
Temporal Forecasting	LSTM, ARIMA, GBR	Time-series modeling in circular economy and predictive maintenance
Decentralized Coordination	PPO, Q-learning (MARL)	Distributed decision-making in smart cities and healthcare SoS settings
Reward Alignment	Hierarchical RL, Reward Shaping	Used in semiconductor and smart cities to reconcile local and global goals
Unstructured Text Analysis	NLP Preprocessing + Naïve Bayes	Technician defect reports in PCB quality control workflows
High-Dimensional Sensor Data	CNN, LSTM	Used for image-based defect detection and fault signal modeling in semiconductor SoS

4.2 Hypothesis Construction and Research Alignment

This section outlines the logical foundation and hypothesis-driven structure that aligns the conceptual framework with the research objectives. Grounded in multi-agent system theory and Actor-Critic reinforcement learning, the proposed hypotheses are organized into two tiers: subsystem-level (local) and system-level (global). The operationalization of these hypotheses is carried out through a series of controlled simulations within the semiconductor manufacturing context.

4.2.1 Framework Foundation

The research methodology builds upon two foundational pillars: Multi-Agent Systems (MAS) and Actor-Critic-based Reinforcement Learning (RL). The integration of these paradigms enables a scalable, modular, and adaptive decision-making structure suited to the dynamic coordination challenges of semiconductor SoS environments.

Table 4.3: Summary of MAS and Actor-Critic Roles in Framework

Aspect	Multi-Agent Systems (MAS)	Actor-Critic RL
Focus	Agent autonomy, distributed control	Policy optimization through feedback
Coordination Level	Subsystem-level (e.g., Inspection, Test)	Hierarchical learning with global signals
Learning Approach	Rule-based and adaptive interaction	Continuous feedback with exploration-exploitation balance
Application	Task allocation, collaboration, real-time decision-making	Reward maximization and convergence assurance

4.2.2 Subsystem-Level Hypotheses

Subsystem-level hypotheses are formulated to evaluate the effect of agent-specific decisions on local performance metrics. These hypotheses test whether reinforcement learning policies improve subsystem reliability, efficiency, and responsiveness:

- **H1:** MARL-based defect inspection agents will reduce undetected defect rates compared to static inspection policies.
- **H2:** Adaptive electrical testing agents will improve fault isolation accuracy while minimizing test redundancy.
- **H3:** Statistical Process Control (SPC) agents will reduce wafer-to-wafer variability through localized interventions.

- **H4:** Inventory management agents will reduce holding costs and prevent material shortages through dynamic restocking policies.
- **H5:** Maintenance agents will reduce unplanned downtimes by scheduling preventive interventions based on learned temporal patterns.

4.2.3 System-Level Hypotheses

System-level hypotheses focus on the emergent behavior and fab-wide performance that results from coordinated agent interactions. These hypotheses evaluate the alignment of local agent decisions with overarching manufacturing goals:

- **H6:** Hierarchical coordination among subsystem agents will lead to measurable yield improvements at the fab level.
- **H7:** Global reward aggregation mechanisms will improve overall resource utilization and reduce inter-agent conflicts.
- **H8:** Incorporating feedback from global agents will increase policy stability and performance convergence across agents.
- **H9:** Integrated SoS-MARL systems will outperform isolated subsystem optimization in terms of throughput and reward efficiency.

4.2.4 Operationalizing Hypotheses

To test the above hypotheses, each simulation scenario is constructed with configurable parameters representing different operational conditions (e.g., defect density, tool availability, SPC deviation). Agents are initialized with exploration-capable policies and are exposed to varying environments over multiple episodes.

- Subsystem hypotheses (**H1**–**H5**) are validated by comparing agent performance to baseline heuristics across metrics such as defect detection accuracy, inspection throughput, downtime reduction, and inventory balance.

- System-level hypotheses (**H6**–**H9**) are tested by aggregating global performance indicators such as fab-wide yield, coordination metrics, conflict scores, and convergence timelines.

Reward functions (both intrinsic and extrinsic) are tracked over time to observe improvement trends, stability, and cooperation emergence. Temporal difference (TD) errors and policy value convergence metrics are also used to validate learning efficacy. Cross-case comparison among simulations with different agent configurations helps in triangulating the influence of MARL policies on system performance.

4.3 Research Philosophy and Approach

This section articulates the philosophical foundations and overarching research strategy that inform the study's methodology. By integrating the principles of pragmatism and mixed-methods research design, the approach accommodates the complexity and dynamism of semiconductor Systems of Systems (SoS) environments.

4.3.1 Philosophical Orientation

The philosophical foundation of this study is grounded in **pragmatism**, which prioritizes actionable knowledge and problem-solving over adherence to a single ontological or epistemological stance. Pragmatism is particularly well-suited for interdisciplinary research involving simulation, agent-based modeling, and emergent system behaviors.

In the context of semiconductor manufacturing:

- Pragmatism supports the integration of both quantitative (e.g., performance metrics, reward trends) and qualitative (e.g., cooperation patterns, policy adaptations) insights.
- It enables flexible experimentation across multiple scenarios and supports data-driven validation without overreliance on theoretical idealization.
- The paradigm aligns with the SoS philosophy, where decentralized systems interact dynamically and often unpredictably.

Given the simulation-driven, multi-agent architecture of this research, pragmatism offers a natural epistemic framework to balance rigor, relevance, and adaptability.

4.3.2 Research Approach

The research employs a **mixed-methods approach** that combines deductive and inductive reasoning. This dual strategy enhances both theory validation and discovery:

- **Deductive reasoning** is used to test predefined hypotheses related to subsystem reliability, yield optimization, and policy coordination using quantitative metrics.
- **Inductive reasoning** is employed to analyze emergent behaviors, such as cooperation strategies, agent adaptation under novel constraints, and conflict resolution dynamics that arise during simulation episodes.

This integration ensures both hypothesis validation and theory-building within the same simulation environment. A visual summary of the mixed-methods logic is provided in Figure 4.2.

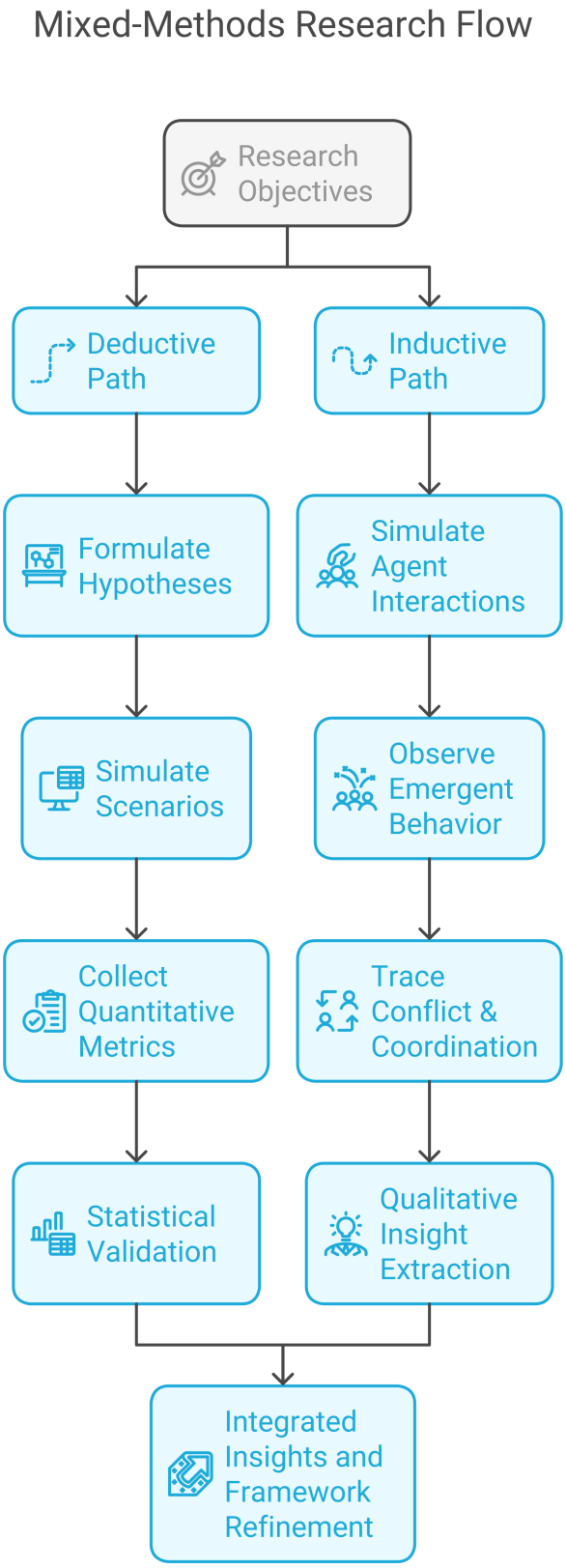

Figure 4.2: Mixed-methods research approach integrating deductive hypothesis testing and inductive behavior discovery. Both paths inform the unified SoS-MARL framework design through integrated insights and iterative refinement.

The research approach accommodates the multifaceted nature of agent-based systems in SoS environments—where both predictable outcomes and emergent dynamics coexist. By blending structured inquiry with open-ended observation, this methodology enables a comprehensive and flexible evaluation of coordination mechanisms in semiconductor manufacturing and beyond.

4.4 Research Design

This section outlines the simulation-based research design, focusing on how SoS principles and hierarchical MARL are operationalized. It details agent classifications, reward mechanisms, policy learning architecture, experimental setup, and the metrics used to evaluate performance.

4.4.1 SoS Agent Mapping

Agents are categorized based on their operational roles within the System-of-Systems (SoS) architecture, mapped into four canonical categories:

- **Directed Systems:** Quality control, defect inspection, and safety compliance agents with top-down command structures and rule-based enforcement.
- **Acknowledged Systems:** Inventory management, order scheduling, and optimization agents that operate semi-independently but align with centralized planning goals.
- **Collaborative Systems:** Maintenance, energy usage, and resource coordination agents that share resources and require negotiation or load balancing.
- **Virtual Systems:** Simulation, prediction, and exploration agents designed to experiment, analyze scenarios, and refine system-wide strategies.

Figure 4.3 illustrates how these agent types are positioned within the broader SoS simulation and their role in information and reward flow.

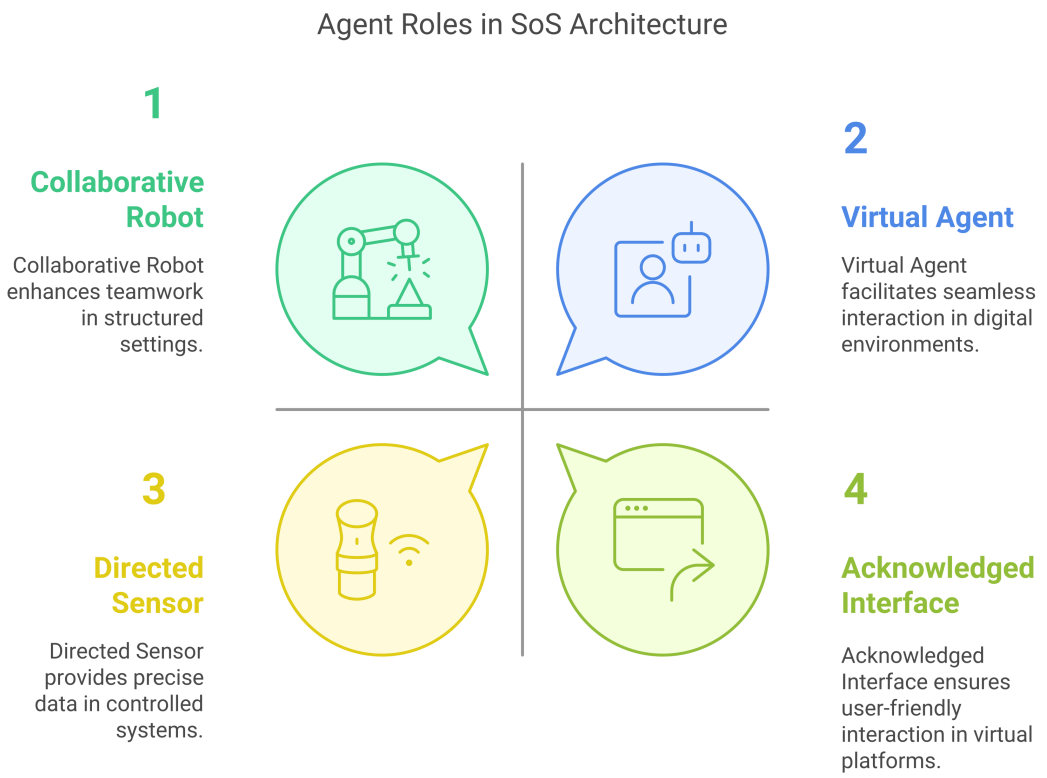

Figure 4.3: Agent roles in SoS architecture across four canonical categories: Directed, Acknowledged, Collaborative, and Virtual Systems.

4.4.2 Reward Structures

The reward system is designed hierarchically to balance local agent autonomy with global system goals. Two core types of rewards are used:

- **Intrinsic Rewards** ($R^{\text{intrinsic}}$): Local agent-specific rewards based on task performance (e.g., defect detection accuracy, inspection throughput, inventory efficiency).
- **Extrinsic Rewards** ($R^{\text{extrinsic}}$): Global feedback based on contribution to fab-wide goals like yield, cooperation, or conflict avoidance.

The total reward $R_{k,j}^{\text{total}}$ for a sub-agent k under main agent j is expressed as:

$$R_{k,j}^{\text{total}} = r_{k,j}^{\text{local}} + r_{k,j}^{\text{subsystem}} + \lambda_{k,j} \cdot r_{k,j}^{\text{global}} \tag{4.1}$$

Global rewards are further weighted across subsystems:

$$R^{\text{global}} = \sum_{j=1}^{M} w_j \cdot R_j^{\text{global}} + \Delta R^{\text{coordination}} \tag{4.2}$$

Figure 4.4 visualizes the interaction between local, subsystem, and global reward signals.

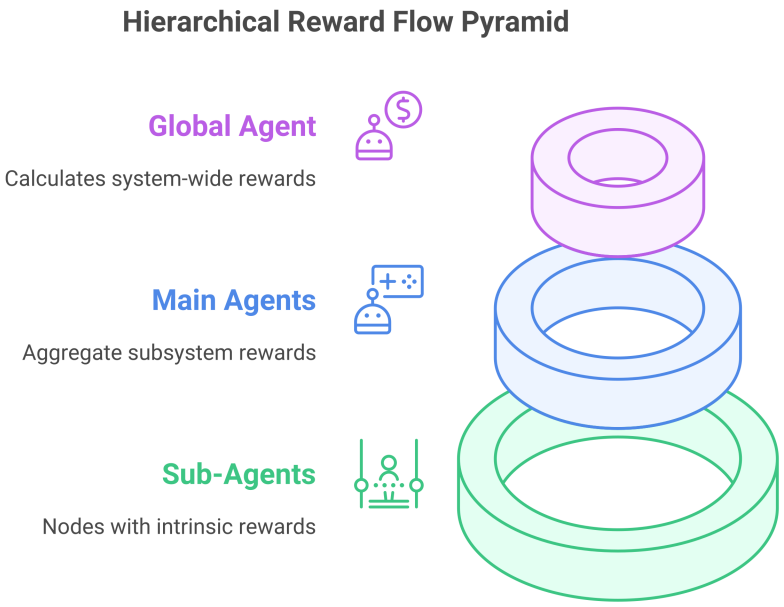

Figure 4.4: Hierarchical reward structure depicting the flow of intrinsic and extrinsic rewards from sub-agents to main agents and the global coordinator.

4.4.3 Hierarchical MARL Design

The simulation employs a three-tiered reinforcement learning design:

- **Global Coordinator (Level 0):** Allocates weights, prioritizes goals, and moderates system-wide coordination.
- **Main Agents (Level 1):** Represent subsystems (e.g., inspection, testing) and manage decision-making for associated sub-agents.

- **Sub-Agents (Level 2):** Operate with localized policies, adapting dynamically to environment conditions using Actor-Critic reinforcement learning.

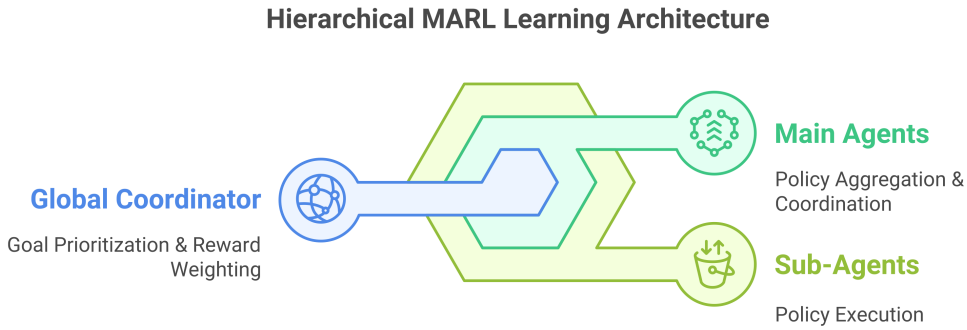

Figure 4.5: Hierarchical MARL learning architecture illustrating policy flow from the global coordinator to main agents and sub-agents.

This architecture supports:

- Policy convergence at both agent and system level.
- Real-time feedback loops.
- Decoupled learning rates for each hierarchy level.

4.4.4 Experimental Framework

The simulation mimics a virtual fab environment, wherein agents operate under discrete time steps across episodes. Sub-agents observe local states and execute actions; main agents monitor their collective behaviors and adjust subsystem strategies; the global agent aggregates rewards and guides macro-level coordination.

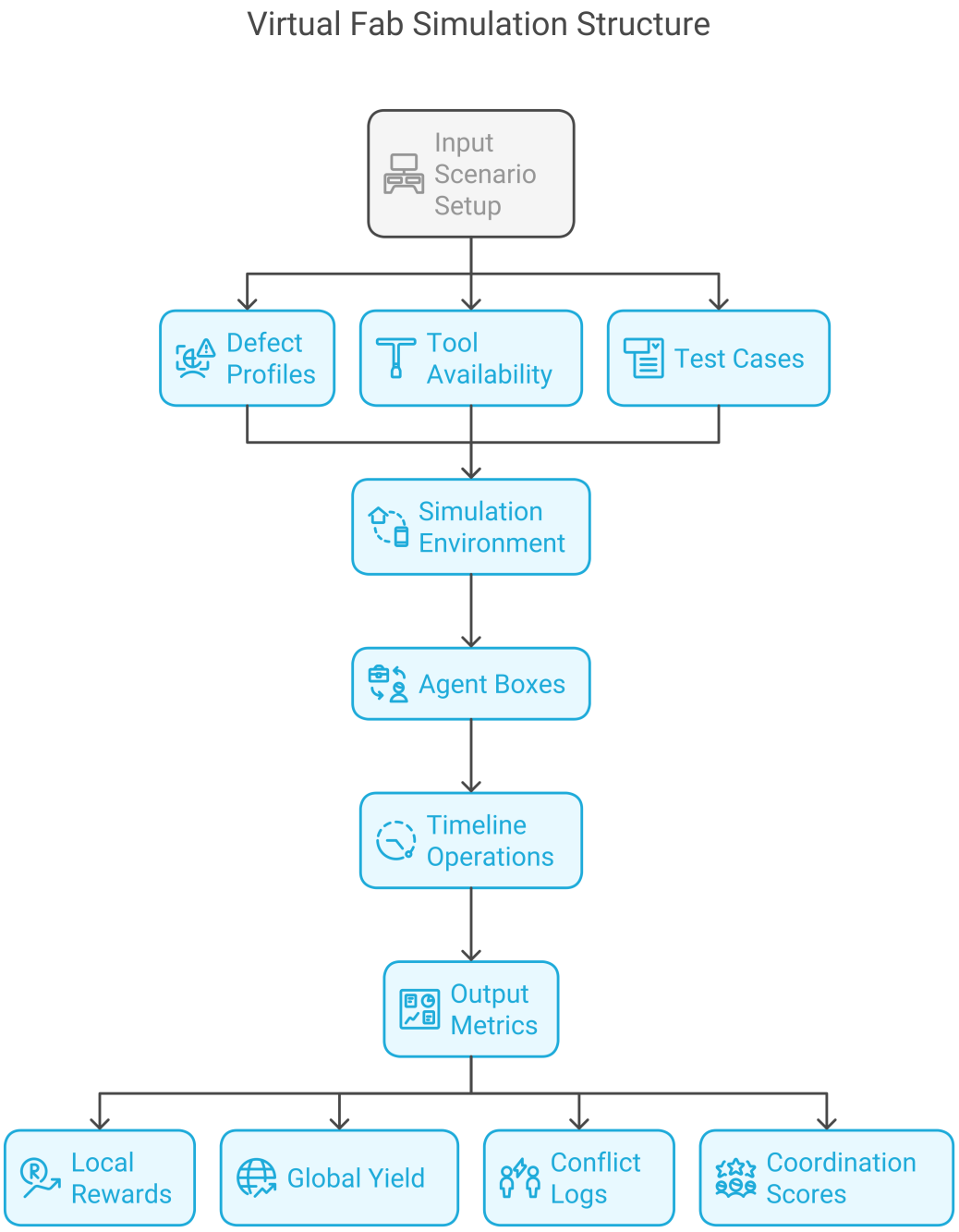

Figure 4.6: Structure of the virtual fab simulation environment showing the flow from input parameters through agent operations to multi-level output metrics.

Key environment characteristics:

- Episodes simulate 24-hour fab operations with 1-hour time steps.

- Each agent operates with partial observability and receives local/global feedback.
- Conflicts (e.g., tool sharing, inspection-test overlap) and cooperative events are logged for analysis.

Effectiveness is assessed using both agent-specific and system-wide metrics:

- **Agent-level Metrics:** Accuracy, resource utilization, reward convergence, action entropy.
- **Subsystem-level Metrics:** Conflict rate, coordination index, reward alignment.
- **System-level Metrics:** Global yield, throughput, R^{global}, and $\Delta R^{\text{coordination}}$.

Sensitivity and ablation analyses are used to measure robustness of coordination logic, policy resilience, and inter-agent dependencies.

4.5 Units of Analysis

To evaluate the performance and effectiveness of the proposed MARL-based Systems of Systems (SoS) framework, the study employs a dual-level unit of analysis. This structure ensures the coherent assessment of both localized subsystem behavior and the emergent properties of global system coordination.

4.5.1 Subsystem-Level

At the subsystem level, the unit of analysis corresponds to individual agents and their associated decision-making policies. Each agent is designed to operate within a distinct SoS category—Directed, Acknowledged, Collaborative, or Virtual—and is evaluated on metrics specific to its functional domain.

- **Agents:** Examples include the Defect Inspection Agent, Electrical Test Agent, Inventory Optimization Agent, and Maintenance Scheduler.
- **Metrics:** Subsystem-level evaluation includes agent-specific performance metrics such as:
  - Defect detection accuracy

  - Test coverage and redundancy elimination
  - Inventory turnover and resource utilization
  - Machine uptime and maintenance efficiency
- **Decision Policies:** The actor-critic reinforcement learning models are monitored for:
  - Reward convergence over training episodes
  - Policy stability and adaptability to environmental variation
  - Action distributions and entropy trends

This level of analysis supports hypothesis testing related to local optimization, such as whether inspection agents improve fault detection under variable defect densities or whether inventory agents reduce material waste under constrained supply conditions.

4.5.2 Global-Level

The global-level unit of analysis aggregates subsystem performance and evaluates the coordination, alignment, and emergent properties of the overall system.

- **Global Coordinator:** A higher-level agent responsible for evaluating and influencing subsystem alignment with fab-wide objectives.
- **System-Wide Metrics:** These include:
  - Total global reward (R^{global})
  - Coordination improvement metrics ($\Delta R^{\text{coordination}}$)
  - Yield improvement rates
  - Conflict resolution and policy compatibility scores
- **Behavioral Assessment:** At this level, emergent behaviors such as conflict resolution patterns, collaborative agent cycles, and convergence trends across subsystems are analyzed to ensure that local optimizations do not undermine global objectives.

This analysis enables the evaluation of hypotheses concerning fab-wide performance, such as whether multi-agent cooperation enhances throughput, or whether reward misalignment among subsystems reduces aggregate yield.

By explicitly structuring the analysis across these two levels, the framework ensures coherence between micro-level agent behaviors and macro-level system outcomes, addressing both the localized and systemic dimensions of semiconductor manufacturing optimization.

4.6 Sampling Strategy

This section outlines the strategy used to design, select, and structure the scenarios that form the basis of the simulation-driven experiments. Given the complexity and hierarchical nature of the Systems of Systems (SoS) framework applied to semiconductor manufacturing, the sampling strategy is tailored to ensure both theoretical rigor and practical relevance.

4.6.1 Population and Scenario Definition

The population under study comprises simulated semiconductor fabrication (fab) environments, modeled as digital ecosystems containing interacting agents. These agents represent subsystems such as defect inspection, electrical testing, inventory management, statistical process control, and maintenance scheduling.

Each simulation run includes:

- A set of wafers with variable defect densities.
- Tool readiness states and processing availability.
- Predefined fault models for test scenarios.
- Synthetic time-series datasets reflecting operational variability across a 24-hour period.

These simulated scenarios are designed to mirror realistic fab dynamics while allowing control over critical input features. They serve as proxies for real-world operational settings to evaluate MARL-based subsystem interactions and global coordination logic.

4.6.2 Sampling Technique

A purposive sampling technique is adopted to ensure the inclusion of high-impact and diverse operational scenarios. This non-probabilistic method prioritizes representativeness

over randomness by intentionally selecting simulation inputs that expose system behavior under varying degrees of stress and complexity.

Key sampling dimensions include:

- **Edge cases:** High-defect wafers, delayed tool availability, or maintenance backlogs.
- **Normal operating conditions:** Moderate defect distributions, on-schedule processing tools, and balanced workload.
- **Anomalous conditions:** Tool failures, surge in defect densities, coordination conflicts, or extreme test loads.

This design ensures that both typical and atypical behaviors are captured and allows the research to evaluate agent robustness, coordination adaptability, and policy convergence across dynamic scenarios.

4.6.3 Sample Size Considerations

Sample size is determined based on the simulation architecture's computational constraints and the need for statistically meaningful results across training episodes and evaluation intervals. Each experimental configuration includes:

- **Simulation runs:** A minimum of 100 independent runs per scenario, each lasting 500 to 1000 agent steps.
- **Agent training episodes:** Each MARL agent undergoes training across 1000 to 3000 episodes, with policy evaluation checkpoints at regular intervals.
- **System configurations:** At least 12 distinct scenario configurations, including both baseline (non-hierarchical) and MARL-enhanced setups.
- **Resource considerations:** Simulations are executed entirely on multi-core CPU environments without the use of GPU acceleration. Computational trade-offs are addressed by balancing model complexity and execution time, with optimization focused on lightweight policy networks suitable for CPU-based training.

This sample size structure ensures convergence of agent policies, meaningful trend detection, and sufficient replication for both quantitative validation and qualitative behavior analysis.

4.7 Data Collection Methods

This section outlines the sources, tools, and techniques used to collect and validate data from the simulation-based experimentation framework. Given the reliance on synthetic environments for modeling agent interactions and evaluating SoS performance, a structured approach was employed to ensure fidelity, reproducibility, and representativeness of the generated data.

4.7.1 Data Sources

The primary data used in this research are synthetically generated through simulation. These data include:

- **Agent-level metrics:** Local rewards, actions, state transitions, and performance indicators (e.g., defect detection rates, inspection throughput, test accuracy).
- **System-level metrics:** Global rewards, coordination scores, yield rates, throughput, and policy convergence indicators.
- **Scenario metadata:** Simulation configuration parameters such as tool readiness levels, defect distributions, and control system variables.

Where relevant, secondary data benchmarks (e.g., typical defect rates or wafer throughput reported in industrial literature) were referenced to calibrate input distributions and validate trend plausibility.

4.7.2 Data Generation Tools

To support high-fidelity simulation and scalable training of MARL agents, the following open-source tools and libraries were employed:

- **PyTorch:** Used for defining and training custom actor-critic neural network models.
- **TensorFlow:** Employed in comparative experiments and for rapid prototyping of lightweight policy networks.
- **Gymnasium:** Provides a structured RL environment interface, enabling modular interaction between agents and fab simulations.
- **Stable-Baselines3:** Offers pre-implemented RL algorithms such as PPO, A2C, and DDPG, which were customized for hierarchical learning.
- **NumPy/pandas:** Used for input preprocessing, feature generation, and structuring agent logs.
- **Matplotlib:** Facilitates visualization of reward trends, episode dynamics, and metric distributions for validation.

All simulations were executed on GPU-enabled servers to support scalable reinforcement learning across large episode counts and multiple agent configurations.

4.7.3 Validation of Input Data

To ensure simulation realism and maintain alignment with semiconductor fab characteristics, the following validation strategies were applied:

- **Statistical sanity checks:** Input variables (e.g., defect rates, queue sizes) were analyzed using descriptive statistics to ensure realistic range coverage and variance.
- **Temporal consistency:** Time-series patterns of key metrics such as defect density and tool readiness were reviewed to reflect plausible operational shifts and production cycles.
- **Visual inspection:** Plots of agent actions, reward trends, and coordination scores were generated to detect anomalies or unrealistic feedback loops.

4.8 Data Analysis Techniques

This section describes the analytical strategies used to evaluate the effectiveness of the proposed SoS–MARL framework. The analysis combines quantitative and qualitative techniques to capture both the statistical behavior of agents and the emergent dynamics of multi-agent coordination. A multi-level lens is adopted to evaluate subsystem performance as well as system-wide outcomes.

4.8.1 Quantitative Analysis

Quantitative methods were employed to assess hypothesis validity and system performance through numerical metrics collected across simulation episodes. Key components of the quantitative evaluation include:

- **Policy Convergence:** Agent policies were monitored for convergence using moving average reward trends and temporal difference (TD) error reduction. Convergence criteria were applied after $N = 500$ episodes to ensure stability.
- **Reward Trend Analysis:** Cumulative reward curves were plotted and compared across agents and episodes. Intrinsic vs. extrinsic reward trajectories were analyzed to determine alignment between local and global objectives.
- **Statistical Testing:** Hypothesis testing (e.g., paired t-tests) was conducted to validate improvements in defect detection accuracy, throughput, and yield under varying coordination settings.
- **Variance Metrics:** Reward variance, coordination scores, and conflict frequency were evaluated to identify fluctuations and instability in agent behavior.
- **Performance Metrics:** Metrics such as defect-free yield, resource utilization, and reward alignment index ($\mathcal{R}_{\text{align}}$) were aggregated and compared across experimental runs.

4.8.2 Qualitative Analysis

While quantitative results offer measurable validation, qualitative methods were used to analyze agent behaviors and system responses under dynamic simulation conditions. These include:

- **Emergent Coordination Patterns:** Logs and visualization dashboards were analyzed to detect coordinated decision-making among agents, such as synchronized inspections or fault isolation reroutes.
- **Conflict Episodes:** Conflict events—where agents attempted incompatible actions (e.g., over-allocating shared tools)—were extracted and examined to assess resolution effectiveness and reward penalties.
- **Adaptive Behaviors:** Observations were made of agent strategies evolving across episodes (e.g., prioritizing critical defects under limited inspection capacity or skipping redundant tests).
- **State-Action Trace Analysis:** Select episodes were traced step-by-step to study how agent decisions responded to environment changes, reward feedback, or coordination signals.

These qualitative insights complement quantitative results by uncovering behaviors that may not be immediately visible through metric summaries alone.

4.8.3 Multi-Level Evaluation

The evaluation framework incorporates both local (agent-level) and global (SoS-level) performance metrics to provide a holistic assessment of the MARL system:

- **Subsystem-Level Metrics:** Accuracy, tool utilization, local reward, and delay time are used to assess agent effectiveness. For example, the defect inspection agent is evaluated based on false negative rate and inspection latency.

- **Global-Level Metrics:** Coordination efficiency, fab-wide yield, system throughput, and conflict resolution rates are used to evaluate the emergent behavior of the system as a whole.
- **Reward Aggregation Formulas:** Local and global metrics are integrated using a weighted reward aggregation:

$$R_{\text{total}} = w_g R_{\text{global}} + \sum_{j=1}^{N} w_j R_j^{\text{local}} \tag{4.3}$$

  where w_g is the weight for global reward, and R_j^{local} is the reward for agent j.
- **Alignment Index:** The reward alignment index $\mathcal{R}_{\text{align}}$ is computed to assess how well local agent policies align with global goals:

$$\mathcal{R}_{\text{align}} = \frac{\sum_j \text{corr}(R_j^{\text{local}}, R_{\text{global}})}{N} \tag{4.4}$$

- **Temporal Comparison:** Episode-wise trend overlays are used to compare local and global metrics over time and assess convergence or divergence.

This multi-level analysis ensures that local optimizations contribute positively to system-wide outcomes, thereby validating the effectiveness of the hierarchical coordination strategy.

4.9 Ethical Considerations

This research upholds high ethical standards in both simulation design and methodological transparency. Although the study does not involve human subjects or proprietary manufacturing data, several ethical considerations were addressed to ensure integrity, reproducibility, and responsible data handling.

Simulation Transparency and Reproducibility

The architecture of the simulation framework—including agent behaviors, reward structures, and environmental parameters—was fully documented to support reproducibility. Codebases were developed using open-source libraries (e.g., PyTorch, Stable-Baselines3, Gymnasium), and simulation scripts were version-controlled and annotated to enable peer replication. Key hyperparameters, agent architectures, and reward formulations are included in Appendices and supplementary materials.

Synthetic vs. Real Data Ethics

The use of synthetic data for training and evaluation avoids any risk of privacy violations, especially given the sensitivity of proprietary data in semiconductor manufacturing. Where real-world patterns were simulated (e.g., defect rates, machine uptime, inspection errors), these were modeled from public sources or generalized industry insights rather than actual fab datasets. In hypothetical scenarios requiring empirical grounding, all references are anonymized and used strictly for academic validation.

Responsible Interpretation

Given that reinforcement learning models can be sensitive to training environments, care was taken not to overstate claims of generalizability. All conclusions are framed within the boundaries of the simulation context, and limitations (Section 4.10) are explicitly discussed to avoid ethical overreach in industrial application. Agent decisions and policies are interpreted within the scope of the learning environment, not real-life deployment outcomes.

Open Science Compliance

This work adheres to the principles of open science and reproducibility by providing simulation configuration details, evaluation metrics, and data analysis pipelines. Where feasible, artifacts will be made available in open repositories to support follow-on studies and independent validation.

4.10 Limitations

While the proposed SoS–MARL framework demonstrates significant potential for coordinating complex systems in semiconductor manufacturing, the study is subject to several limitations, particularly related to simulation constraints and external validity.

Simulation vs. Real-World Fidelity

The framework was developed and tested in a synthetic simulation environment rather than a physical fab. While this allows for fine-grained control and safe experimentation, it cannot capture the full complexity of real-world fabs, such as unplanned equipment failures, labor variability, external supply chain disruptions, or long-term process drift. Thus, the results, while internally valid, require further validation before industrial deployment.

Computational Constraints

The use of hierarchical MARL across multiple agents, with policy optimization occurring at both local and global levels, incurs significant computational overhead. Training and evaluating agents across hundreds of episodes required extensive GPU resources and memory optimization. This constraint limited the exploration of very large or highly granular fab models, and future work may benefit from distributed or cloud-based RL infrastructure.

Scalability and Policy Transfer

While the agent architecture is modular, its scalability to highly complex fabs with dozens of interacting subsystems remains to be empirically validated. Moreover, policies learned in one scenario may not easily transfer to another without retraining or fine-tuning, limiting generalizability. Mechanisms such as transfer learning, continual learning, or federated RL could enhance adaptability across use cases.

Future Validation Pathways

To address these limitations, future research should include:

- Integration of real-world fab logs, where available, to validate simulation dynamics.
- Exploration of human-in-the-loop coordination models and hybrid simulation-lab environments.
- Extension to multi-fab ecosystems and supply chain-level coordination using federated MARL.

These pathways would support the progressive transition from theoretical validation to operational deployment in semiconductor and other cyber-physical system domains.

4.11 Chapter Summary

This chapter outlined the comprehensive methodological framework used to implement and evaluate the proposed System-of-Systems (SoS) and Multi-Agent Reinforcement Learning (MARL) architecture for semiconductor manufacturing optimization. Grounded in a pragmatic philosophy, the research approach blends deductive hypothesis testing with inductive analysis of emergent behaviors, leveraging simulation environments to model subsystem coordination and global fab-wide objectives.

Key contributions of the methodology include:

- The construction of testable hypotheses aligned with both subsystem-level (e.g., defect detection, test accuracy) and system-level (e.g., yield improvement, coordination efficiency) goals.
- The design of a hierarchical reward framework that reconciles intrinsic agent objectives with extrinsic system-wide optimization metrics.
- A multi-layered agent design mapped to Directed, Acknowledged, Collaborative, and Virtual SoS categories, enabling distributed decision-making under shared constraints.
- Detailed simulation architectures, sampling strategies, and data generation protocols using tools such as PyTorch, Gymnasium, and Stable-Baselines3.
- Integration of both quantitative and qualitative data analysis techniques, facilitating a holistic evaluation of performance, coordination, and policy robustness.

The methodology provides the foundation for the empirical analysis presented in the following chapters. **Chapter 5** through **Chapter 9** detail seven case studies that apply the SoS–MARL framework to diverse domains including semiconductor yield optimization, healthcare coordination, SME manufacturing, smart cities, circular economy, and text analytics for quality control. Each case operationalizes the core methodological elements developed in this chapter to address domain-specific challenges while contributing to the broader goals of this dissertation.

Chapter 5

ML for Optimization in Semiconductor SoS

5.1 Introduction

Semiconductor manufacturing presents a complex and data-intensive environment that benefits significantly from machine learning (ML)-driven optimization. This chapter presents three case studies that address distinct challenges within this domain: yield enhancement in 3D-IC fabrication, hierarchical decision-making via reinforcement learning, and system dynamics modeling tailored to small and medium enterprises (SMEs). These studies demonstrate how ML methods can enhance decision support, scalability, and process efficiency in complex System-of-Systems (SoS).

5.2 3D-IC Manufacturing: Yield Optimization Using CNN and LSTM

5.2.1 Problem Overview and Context

Three-Dimensional Integrated Circuit (3D-IC) manufacturing has emerged as a next-generation paradigm in semiconductor engineering, enabling enhanced performance, reduced footprint, and improved energy efficiency through vertical stacking and Through-Silicon Via (TSV) technologies. Despite these advantages, the 3D-IC manufacturing process is characterized by high complexity, involving tightly coupled stages such as bumped die attachment, TSV formation, and electrical testing. These stages exhibit nonlinear interdependencies, sub-micron precision requirements, and high sensitivity to process variability.

Traditional manufacturing control techniques, including rule-based and statistical process controls, struggle to adapt to the high-dimensional and dynamic nature of modern semiconductor fabrication. In particular, TSV formation represents a critical bottleneck,

where variations in parameters such as depth, aspect ratio, and deposition conditions can significantly impact structural integrity and inter-layer connectivity.

This case study focuses on a scalable machine learning-driven framework to address yield and defect challenges in 3D-IC production, with particular emphasis on early-stage defect detection during the bumped die attachment phase. This stage is pivotal for establishing robust downstream connectivity and preventing defect propagation through subsequent layers. The study leverages a System of Systems (SoS) framework to coordinate multiple manufacturing subsystems—defect detection, TSV optimization, and electrical failure prediction—through dynamic, feedback-driven decision-making.

By treating each stage of the manufacturing process as an autonomous yet interlinked subsystem, the framework provides a cohesive strategy for optimizing global performance objectives such as yield, reliability, and defect minimization. This integrated approach supports real-time adaptation to dynamic manufacturing conditions, representing a significant departure from traditional, static optimization techniques.

5.2.2 Research Objectives and Questions

The primary objective of this case study is to develop a machine learning-driven optimization framework for Three-Dimensional Integrated Circuit (3D-IC) manufacturing using a System of Systems (SoS) approach. The framework integrates key subsystem capabilities—defect detection, process parameter optimization, and electrical failure prediction—into a cohesive, feedback-enabled decision-support system designed to enhance yield, reliability, and adaptability in semiconductor production.

Specifically, this case study addresses the following research objectives:

- **Objective 1:** To design and evaluate Random Forest classifiers for early-stage defect detection during the bumped die attachment phase.

- **Objective 2:** To implement an adaptive optimization mechanism for Through-Silicon Via (TSV) formation using simulated annealing and CNN-based dynamic weighting strategies.
- **Objective 3:** To develop and integrate Long Short-Term Memory (LSTM) models for time-series-based electrical failure prediction in downstream processes.
- **Objective 4:** To coordinate defect detection, process optimization, and failure prediction subsystems within a unified SoS framework that aligns local decisions with global manufacturing goals such as yield enhancement and defect minimization.

Based on these objectives, the following research questions guide the investigation:

- **RQ1:** How effectively can CNNs detect manufacturing defects during early-stage 3D-IC assembly processes?
- **RQ2:** In what ways can adaptive optimization techniques improve TSV parameter tuning under varying manufacturing conditions?
- **RQ3:** Can LSTM models provide accurate and timely electrical failure predictions when integrated with upstream defect and process data?
- **RQ4:** How can coordinated decision-making within an SoS framework enhance overall system-level performance in 3D-IC manufacturing?

5.2.3 Methodology

This section outlines our AI-driven decision-making framework designed to optimize 3D-IC manufacturing through a System of Systems (SoS) approach. The framework integrates defect detection during the critical bumped die attachment stage using Random Forest Classifiers, TSV process optimization through advanced mathematical modeling, and electrical failure detection using Long Short-Term Memory (LSTM) networks. These methodologies are supported by iterative optimization techniques to ensure system-wide alignment and scalability.

The methods described in this section are adapted and extended from our previously published study on SoS-driven optimization in 3D-IC manufacturing [15].

The bumped die attachment stage is chosen for its pivotal role in ensuring die placement and interconnection quality, which directly influence the mechanical and thermal reliability of the assembled device.

Figure 5.1 illustrates the workflow of the SoS framework. Subsystem outputs, including TSV parameters and defect classifications, feed into a unified global optimization process. This optimization occurs in an interactive mode, where decisions are updated periodically based on batch-processed manufacturing data rather than real-time streaming. The feedback loop ensures coordinated decision-making and enhances system-wide performance.

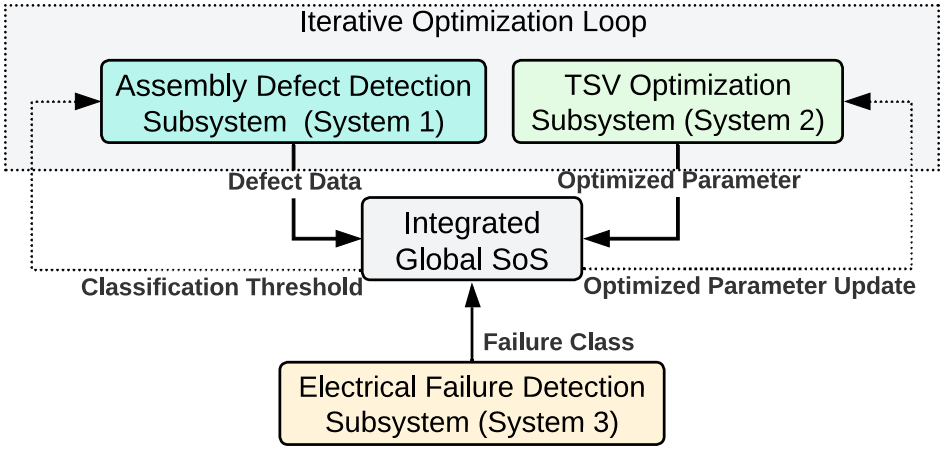

Figure 5.1: Workflow of the System of Systems (SoS) framework for 3D-IC manufacturing optimization. Subsystem outputs are integrated through an iterative optimization loop to align local and global objectives.

Our SoS framework employs a unified objective function with dynamically calibrated weights, enabling seamless integration of subsystem outputs into global optimization. Building on prior work in defect detection and AI-driven decision-making [20, 68], the framework addresses interdependencies between subsystems while balancing yield, defect rates, and resource utilization.

The following subsections detail the models, algorithms, and metrics employed to achieve both subsystem and system-level optimization.

Input Data for Modeling and Simulation Setup

The input data for the modeling and simulation framework is carefully curated to emulate realistic scenarios in 3D-IC manufacturing [69–71]. The dataset includes structured feature data, process parameters, and time-series measurements for three systems and an aggregated global SoS.

System 1: Defect Detection For the bumped die attachment stage, the dataset comprises structured feature data with 2,000 samples, each annotated with alignment offset (range: $-5\,\mu m$ to $5\,\mu m$), solder resistance ($0.1\,\Omega$ to $6.0\,\Omega$), bump height ($25\,\mu m$ to $50\,\mu m$), reflow pressure (50 kPa to 200 kPa), and reflow temperature (200 °C to 300 °C). These features are derived from synthetic experiments replicating variability in real-world processes.

System 2: TSV Optimization The TSV optimization dataset captures process parameters associated with critical fabrication stages. It includes 1,500 records, detailing TSV depth ($30\,\mu m$ to $36\,\mu m$), deposition rate ($0.8\,\mu m$/m to $1.2\,\mu m$/m), etching temperature (35 °C to 39 °C), and aspect ratio (range: 2 : 1 to 10 : 1). These parameters represent controlled variability during TSV manufacturing.

System 3: Electrical Failure Detection The electrical failure detection system analyzes 3,000 time-series sequences (100 timestamps each), with measurements for resistance ($0.1\,\Omega$ to $6.0\,\Omega$), capacitance (20 pF to 60 pF), and leakage current ($0.05\,\mu$A to $5.0\,\mu$A). These sequences capture temporal variations in electrical properties, with annotated fault labels.

Integrated Global SoS At the system level, the framework aggregates subsystem outputs, including defect rates, TSV process optimizations, and electrical failure classifications.

This unified dataset serves as input for global optimization tasks, ensuring alignment with overarching system objectives.

A summary of the input data characteristics is presented in Table 5.1. These datasets form the foundation for validating the proposed framework and demonstrating its applicability to realistic manufacturing scenarios.

Table 5.1: Summary of Input Data for Modeling and Simulation

Subsystem	Data Type	Key Features
System 1	Structured Data	Alignment Offset, Solder Resistance, Bump Height, Reflow Pressure, Reflow Temperature
System 2	Process Parameters	TSV Depth, Deposition Rate, Etching Temperature, Aspect Ratio
System 3	Time-Series Data	Resistance, Capacitance, Leakage Current
Global SoS	Aggregated Data	Defect Rates, TSV Parameters, Electrical Failure Classifications

Model Selection Justification

The choice of machine learning models in our study was guided by their suitability for specific semiconductor manufacturing tasks, balancing interpretability, computational feasibility, and predictive performance.

Defect Detection (System 1) Random Forest was selected for its ability to handle mixed-type feature spaces and its robustness to small dataset variations [72]. While SVM has strong classification capabilities, its computational overhead in hyperparameter tuning and lack of intrinsic feature importance analysis made it less practical for real-time defect classification [73]. Gradient Boosting methods, such as XGBoost, offered competitive per-

formance but required extensive hyperparameter tuning to achieve a recall-precision tradeoff suitable for imbalanced datasets [74].

TSV Process Optimization (System 2) CNN-based adaptive weighting was chosen over traditional boosting methods due to its ability to dynamically adjust parameter importance based on real-time manufacturing constraints. Unlike Gradient Boosting, which requires explicit feature engineering, CNNs inherently capture complex spatial dependencies in process parameters [75]. SVM was tested but was found to scale poorly with high-dimensional process optimization tasks.

Electrical Failure Detection (System 3) LSTM-based models were selected for their ability to capture temporal dependencies in sequential electrical test data [76]. SVM and Random Forest models, while effective for static classification, struggled to model time-dependent relationships, leading to lower recall in detecting electrical anomalies.

This selection process ensures that each model aligns with the specific requirements of its respective subsystem, optimizing defect detection, TSV parameter tuning, and electrical failure classification in an integrated System-of-Systems framework.

Defect Detection Using Random Forest Classifiers

System 1 workflow for defect detection integrates data preprocessing, feature engineering, SMOTE balancing, Optuna-based hyperparameter tuning, and model evaluation using a Random Forest classifier, as illustrated in Figure 5.2.

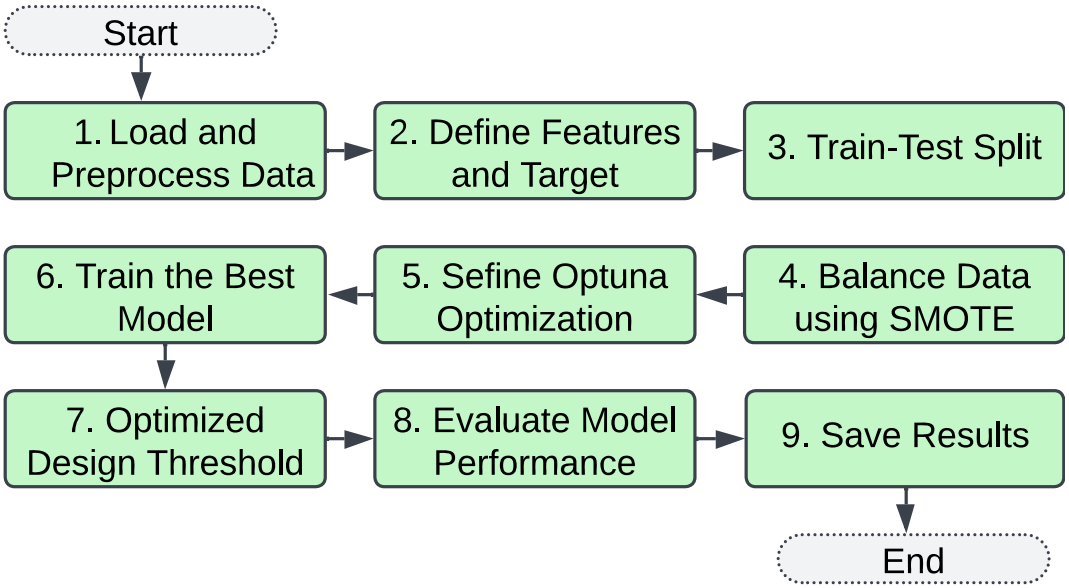

Figure 5.2: System 1 Algorithm Flow Diagram - Workflow for defect detection, including data preprocessing, feature engineering, SMOTE balancing, Optuna hyperparameter tuning, and performance evaluation.

The defect detection process focuses on the bumped die attachment stage, a critical step in 3D-IC manufacturing that directly influences die placement accuracy and interconnection quality. Ensuring reliability at this stage is pivotal to preventing defect propagation in subsequent manufacturing processes. A Random Forest Classifier is employed for its robustness in handling feature interactions, reducing overfitting, and providing interpretable results [20, 72].

Recent advancements in AI-driven defect detection have introduced novel deep learning frameworks and hybrid models aimed at enhancing classification performance in semiconductor manufacturing. Generative adversarial networks (GANs) have been applied to mitigate class imbalance by generating synthetic defect samples, improving fault detection accuracy in highly imbalanced datasets [36]. Additionally, hybrid multistage deep neural networks (SH-DNN) have demonstrated improved detection of microscopic defects by integrating classical computer vision pipelines with deep learning models, achieving enhanced fault classification for semiconductor assembly [38].

The dataset used in our study comprises five structured features: Alignment Offset (μm), Solder Resistance (Ω), Bump Height (μm), Reflow Pressure (kPa), and Reflow Temperature (°C). The data is split 70:30 for training and testing.

Building on prior AI applications in circuit board assembly [68], our study extends machine learning to analyze structured feature data in TSV manufacturing. The proposed framework dynamically refines classification thresholds, accounting for interdependencies between defect detection and subsequent processes. By incorporating state-of-the-art AI-driven defect detection strategies, such as generative modeling and hybrid deep learning architectures, this approach enhances classification robustness while ensuring scalability in real-time semiconductor manufacturing workflows.

Addressing Class Imbalance To address class imbalance, two complementary strategies are employed:

1. **Class Weighting**: Class weights are dynamically adjusted inversely proportional to class frequencies [77]:

$$w_c = \frac{N}{N_c}, \quad c \in \{0, 1\} \tag{5.1}$$

   where w_c represents the weight for class c, N is the total sample size, and N_c is the count of samples in class c.

2. **SMOTE (Synthetic Minority Oversampling Technique)**: Synthetic samples are generated for the minority class using linear interpolation [78]:

$$\mathbf{x}_{\text{synthetic}} = \mathbf{x}_{\text{minority}} + \lambda(\mathbf{x}_{\text{neighbor}} - \mathbf{x}_{\text{minority}}), \quad \lambda \sim \mathcal{U}(0, 1) \tag{5.2}$$

   where $\mathbf{x}_{\text{minority}}$ and $\mathbf{x}_{\text{neighbor}}$ represent feature vectors specific to the minority class, while $\mathbf{x}_{\text{synthetic}}$ denotes the newly generated synthetic sample. The distinction between $\mathbf{x}_{\text{minority}}$ (existing sample) and $\mathbf{x}_{\text{neighbor}}$ (selected neighbor) is critical, as their interpolation defines the synthetic sample's placement within the feature space. This approach

ensures the generation of diverse and realistic synthetic samples, enhancing dataset representativeness and model generalization by mitigating class imbalance.

Model Overview The Random Forest Classifier, comprising T decision trees, predicts class labels based on majority voting [72]:

$$\hat{y} = \text{mode}\{h_t(\mathbf{z}) \mid t = 1, 2, \ldots, T\} \tag{5.3}$$

where $h_t(\mathbf{z})$ denotes the prediction of the t-th tree. Feature importance is computed as the average reduction in Gini impurity across splits for each feature [72]:

$$\text{Importance}(j) = \frac{1}{T} \sum_{t=1}^{T} \sum_{s \in \text{Splits}(t,j)} \Delta G_s \tag{5.4}$$

where j represents a specific feature, T is the total number of trees in the Random Forest Classifier, s is a specific split involving feature j, and ΔG_s denotes the reduction in Gini impurity at split s. This measure quantifies the contribution of each feature to the classifier's decision-making process, providing interpretability and actionable insights for defect detection [20].

Performance Metrics The classifier is evaluated using standard metrics [79]:

- **Accuracy**: Proportion of correctly classified samples:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.5}$$

- **Precision (Defect Class)**: Ratio of true positives to total predicted positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.6}$$

- **Recall (Defect Class)**: Ratio of true positives to actual positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.7}$$

- **F1-Score (Defect Class)**: Harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.8}$$

Here, the term "Defect Class" refers to the minority class representing defective samples in the dataset. Metrics such as precision, recall, and F1-Score are evaluated specifically for this class to assess the model's ability to detect and classify defects, given the critical importance of minimizing undetected defects in manufacturing processes.

This framework integrates effective class imbalance handling, interpretability, and robustness, reflecting the principles of structured feature analysis and adaptive decision-making outlined in [68] and [20].

Process Optimization Modeling and Simulation

System 2 process optimization pipeline employs CNN-based adaptive weight prediction, dual annealing optimization, and convergence analysis for TSV manufacturing, as illustrated in Figure 5.3.

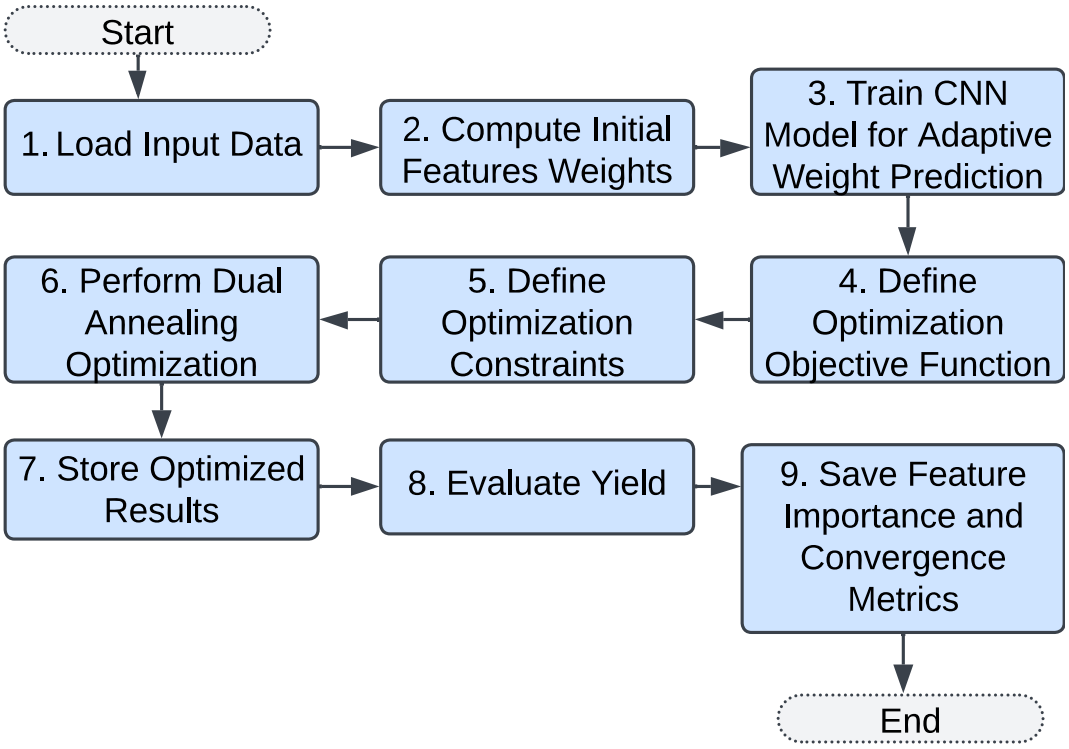

Figure 5.3: System 2 Algorithm Flow Diagram - TSV process optimization pipeline, integrating CNN-based adaptive weight prediction, dual annealing optimization, and convergence analysis.

The TSV process optimization utilizes a hybrid framework that integrates advanced mathematical modeling, adaptive parameter weighting, and iterative search techniques to minimize defect rates and enhance uniformity. Key optimization targets include TSV depth (D), deposition rate (R), and etching temperature (T), aligned with industry standards to ensure scalability and reliability.

Objective Function The optimization is governed by an adaptive objective function that dynamically assigns weights to parameters based on their contributions to defect reduction [80]:

$$f(D,R,T) = w_1 \cdot \frac{|D - D_{\text{ideal}}|}{D_{\text{ideal}}} + w_2 \cdot \frac{|R - R_{\text{ideal}}|}{R_{\text{ideal}}} + w_3 \cdot \frac{|T - T_{\text{ideal}}|}{T_{\text{ideal}}} \tag{5.9}$$

where w_1, w_2, w_3 are dynamically adjusted weights predicted by a CNN module. An additional penalty term ensures uniform aspect ratios:

$$f(D, R, T) = w_1 \cdot \frac{|D - D_{\text{ideal}}|}{D_{\text{ideal}}} + w_2 \cdot \frac{|R - R_{\text{ideal}}|}{R_{\text{ideal}}} + w_3 \cdot \frac{|T - T_{\text{ideal}}|}{T_{\text{ideal}}} \tag{5.10}$$

where λ is a scaling factor, and $AR_{\text{target}} = 5.0$.

Optimization Framework The framework employs a hybrid strategy that combines global search techniques with adaptive parameter tuning:

- **Simulated Annealing**: This technique explores broader parameter spaces and probabilistically accepts suboptimal solutions to avoid local minima.
- **CNN-Based Weighting**: Parameter weights are iteratively adjusted to align with defect reduction objectives while maintaining process feasibility.

This hybrid approach ensures robust convergence, balancing exploration and precision in parameter tuning.

Simulation Assumptions Simulated input data reflects realistic variability in TSV manufacturing, with parameter ranges defined as:

$$\begin{aligned} 30\,\mu m \leq D \leq 36\,\mu m, \\ 0.8\,\mu m/\text{m} \leq R \leq 1.2\,\mu m/\text{m}, \\ 35\,^{\circ}\text{C} \leq T \leq 39\,^{\circ}\text{C}. \end{aligned}$$

Computational Framework Parallel processing and CNN-driven adaptive weighting enable efficient optimization. By dynamically adjusting priorities during iterations, the framework achieves robust parameter estimations aligned with defect reduction goals.

This hybrid methodology builds on foundational techniques in AI-driven manufacturing [20, 68], offering a scalable and adaptable solution for high-variability environments in 3D-IC production.

Electrical Failure Detection Using LSTM-Based Modeling

System 3 electrical failure detection framework utilizes LSTM modeling, SMOTE balancing, model training, evaluation, and feature importance analysis, as illustrated in Figure 5.4.

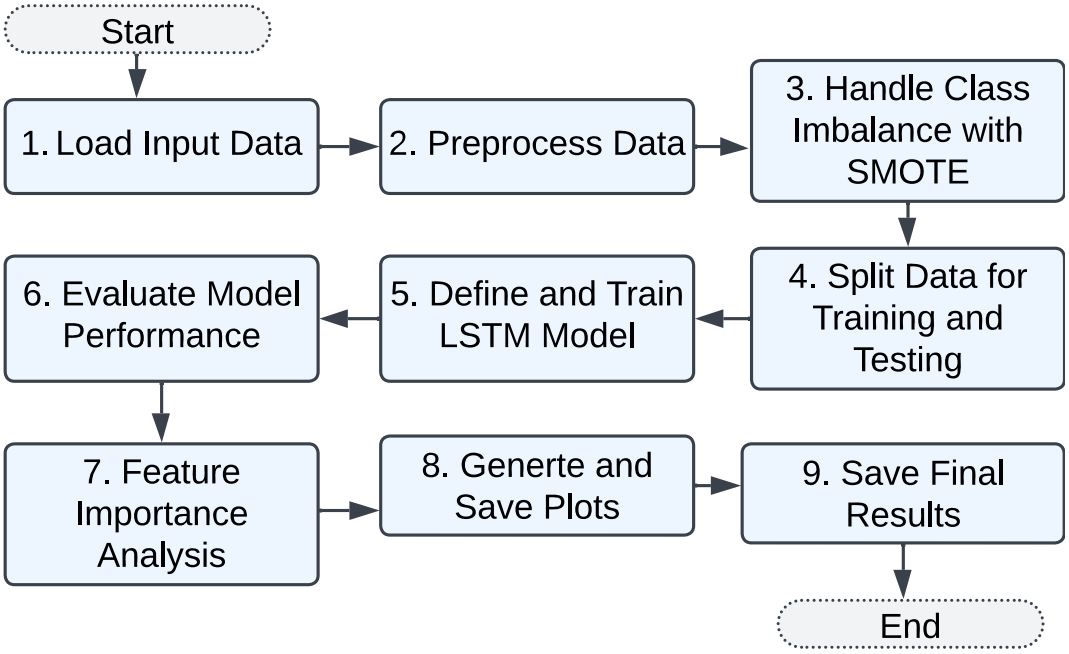

Figure 5.4: System 3 Algorithm Flow Diagram - Electrical failure detection framework utilizing LSTM modeling, SMOTE balancing, model training, evaluation, and feature importance analysis.

Electrical failure detection during final testing is modeled using a Long Short-Term Memory (LSTM) network [20,76], leveraging its ability to capture temporal dependencies in time-series data. This approach enhances anomaly detection for resistance (Ω), capacitance (pF), and leakage current (µA).

Recent advancements integrate hybrid models, combining LSTMs with isolation forests to improve early failure detection in semiconductor testing [17]. Additionally, deep learn-

ing frameworks incorporating log-based event data have demonstrated improved predictive accuracy by leveraging non-uniform timestamp sequences [81].

Our study enhances LSTM-based modeling by dynamically adjusting failure detection thresholds based on subsystem outputs from TSV optimization and defect detection. The framework ensures system-wide coherence by integrating predictive analytics, improving real-time adaptability and reliability in semiconductor testing.

Each time-series sample $\mathbf{X}_i$ is represented as follows [76]:

$$\mathbf{X}_i = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1F} \\ x_{21} & x_{22} & \cdots & x_{2F} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\tau 1} & x_{\tau 2} & \cdots & x_{\tau F} \end{bmatrix}, \quad y_i \in \{0, 1\}, \tag{5.11}$$

where τ is the number of time steps, F is the number of features, and y_i is the binary label indicating electrical failure. This structured representation enables the LSTM to effectively model temporal patterns and dependencies within the data.

Class Imbalance Handling To address class imbalance, the framework employs two strategies:

Class Weighting: As defined in 5.1, class weights are adjusted inversely to class frequencies to reduce the dominance of the majority class. This technique, initially used in defect detection, is adapted here for electrical failure detection.

SMOTE: The Synthetic Minority Oversampling Technique (SMOTE), outlined in 5.2, generates synthetic samples for the minority class, enhancing dataset diversity and balancing class distribution. This supports better model generalization and improves the detection of electrical failures.

LSTM Architecture The LSTM model consists of an input layer with 64 memory cells, employing a hyperbolic tangent (tanh) activation function. This is followed by a dense layer

with 32 neurons using the ReLU activation function and a sigmoid output layer that maps predictions to a binary probability:

$$\hat{y}_i = \frac{1}{1 + e^{-z}}, \quad z = \mathbf{w}^\top \mathbf{h} + b, \tag{5.12}$$

where $\mathbf{h}$ is the output of the dense layer, and $\mathbf{w}, b$ are the weights and bias of the output layer. Binary cross-entropy loss optimizes the model [82]:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right], \tag{5.13}$$

where θ represents the model parameters, y_i the true label, and $\hat{y}_i$ the predicted probability.

Optimization and Scalability The model was trained using an 80:20 train-test split with the Adam optimizer. Dropout regularization mitigated overfitting, and hyperparameters were optimized through grid search. This design ensures that the LSTM generalizes across diverse manufacturing lots, enhancing scalability. By integrating SMOTE and class weighting, the framework achieves robust performance under class imbalance, laying the foundation for broader applicability in dynamic manufacturing contexts.

Global SoS Integration and Optimization Framework

Our SoS framework integrates outputs from three interconnected manufacturing subsystems—defect detection during bumped die attachment, TSV process optimization, and electrical fault detection—into a unified global optimization process for 3D-IC production. Each subsystem operates autonomously while contributing to the collective global state, defined at time t as

$$S_t = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}, \tag{5.14}$$

where $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ (as previously defined) represent the respective states of the subsystems. These states propagate localized decisions, aligning them with global objectives.

The global yield metric Y_{global} quantifies overall manufacturing performance and is expressed as the product of individual subsystem yields [83]:

$$Y_{\text{global}} = \prod_{i=1}^{3} Y_i, \tag{5.15}$$

where Y_i denotes the yield of subsystem i. The corresponding global defect rate D_{global} is given by

$$D_{\text{global}} = 1 - Y_{\text{global}}. \tag{5.16}$$

To address varying subsystem criticalities, the framework employs a weighted global yield $\mathcal{O}_w(S_t)$, defined as [84]

$$\mathcal{O}_w(S_t) = \sum_{i=1}^{3} w_i Y_i, \quad w_i = \frac{1/D_i}{\sum_{j=1}^{3}(1/D_j)}, \tag{5.17}$$

where w_i is the weight of subsystem i, normalized by the inverse of its defect rate D_i. This ensures that subsystems with lower defect rates exert greater influence on the weighted yield, aligning their outputs with global manufacturing objectives.

Sensitivity analysis quantifies the impact of each subsystem on overall performance. The sensitivity of subsystem i is calculated as

$$\text{Sensitivity}_i = \frac{|Y_{\text{global}} - Y_i|}{Y_{\text{global}}} \times 100, \tag{5.18}$$

indicating the relative influence of subsystem yield variations. Additionally, the yield loss contribution of subsystem i is given by

$$\text{Loss Contribution}_i = \frac{1 - Y_i}{D_{\text{global}}} \times 100, \tag{5.19}$$

providing insight into the proportion of global defects attributable to each subsystem.

This unified framework dynamically integrates subsystem outputs, balancing localized performance with global objectives. By leveraging weighted contributions, sensitivity analysis, and defect rate optimization, the SoS framework establishes a robust methodology for enhancing system-wide yields and reducing defect rates in 3D-IC manufacturing.

5.2.4 Results and Analysis

The performance of the proposed AI-driven SoS framework was evaluated across its three subsystems and their contributions to the global SoS. Table 5.2 provides a high-level summary of subsystem roles, inputs, outputs, and interconnections.

Table 5.2: Workflow and Interactions in the SoS Framework

System	Description	Inputs
System 1: Defect Detection	Detects assembly defects using Random Forest models.	Defect data
System 2: TSV Optimization	Optimizes TSV depth, deposition rate, and temperature.	TSV parameters
System 3: Failure Detection	Identifies electrical failures using LSTM-based models.	Test data
Integrated Global SoS: System Coordination	Aggregates subsystem outputs and coordinates optimization.	Subsystem data

Results and Analysis: System 1 3D Defect Detection

The defect detection model for System 1 was evaluated using a class-weighted Random Forest classifier. The dataset included five key features: Alignment Offset (μm), Solder Resistance (Ω), Bump Height (μm), Reflow Pressure (kPa), and Reflow Temperature (°C),

split 70:30 for training and testing. To address class imbalance, SMOTE was applied to augment the minority class (defects), enabling the model to achieve improved detection performance.

SMOTE vs. No SMOTE Comparison The application of SMOTE significantly enhanced the representation of the defect class (Class 1), as shown in Fig. 5.5. Here, Class 0 refers to non-defective samples, while Class 1 represents defective samples. Without SMOTE, the dataset was dominated by the non-defective class (Class 0), leading to biased predictions. After applying SMOTE, the defective class was upsampled, balancing the dataset and enabling the model to achieve higher recall for defects. This adjustment allowed the model to detect minority class defects more effectively without severely compromising overall precision.

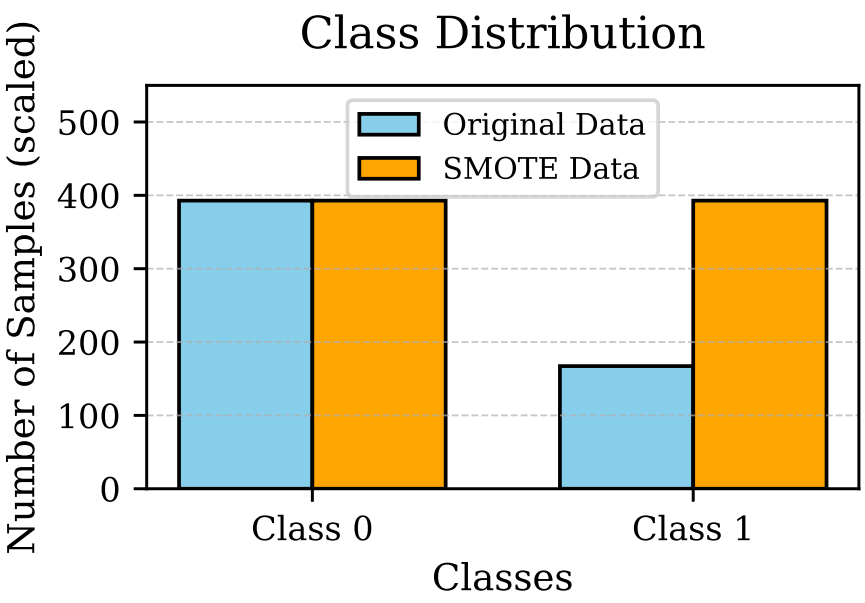

Figure 5.5: System 1 class distribution before and after applying SMOTE. Class 0 refers to non-defective samples, while Class 1 represents defective samples.

Feature Importance and Loss Curve Feature importance analysis (Fig. 5.6) revealed Ln Solder Resistance (0.42 Ω) as the most critical predictor, highlighting its strong influence on defect detection due to its direct relationship with electrical conductivity variations in solder joints. Alignment Offset (0.18 μm) ranked second, underscoring the significance of precise alignment in minimizing defect occurrences during assembly. Bump Height (0.12 μm) and Reflow Temperature (0.10 °C) were identified as important features, reflecting their impact on structural integrity and material behavior during reflow. Lastly, Reflow Pressure (0.08 kPa), Temp-Pressure Interaction (0.05), and Height-Pressure Ratio (0.03) showed moderate contributions, capturing nuanced interactions between process parameters.

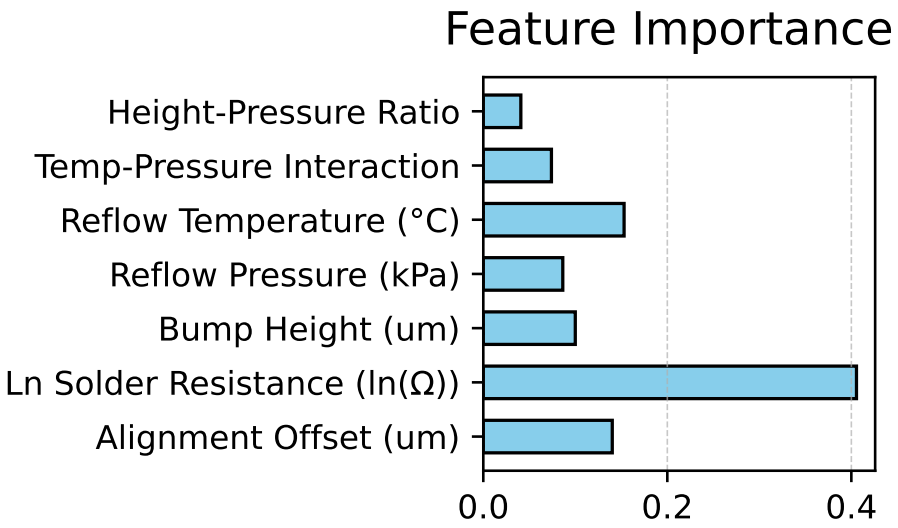

Figure 5.6: Feature importance for System 1 defect detection.

The training and validation loss curves (Fig. 5.7) demonstrated stable convergence, indicating consistent model performance and effective learning.

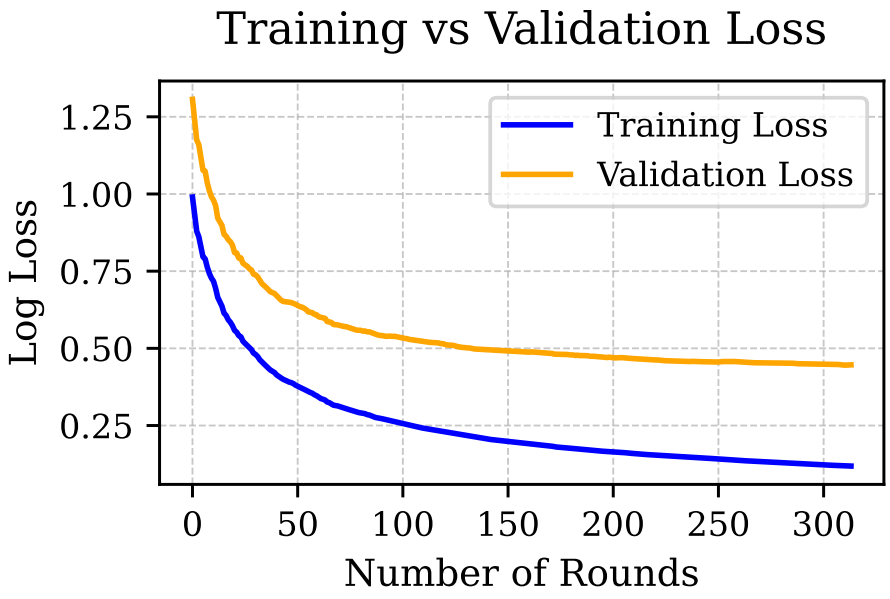

Figure 5.7: Training and validation loss curves for defect detection for System 1.

Precision-Recall and ROC Analysis The precision-recall curve (Fig. 5.8) highlights improved performance, with an AUC-PR of 0.89. The ROC curve (Fig. 5.9) further supports robust defect detection, achieving an AUC-ROC of 0.94.

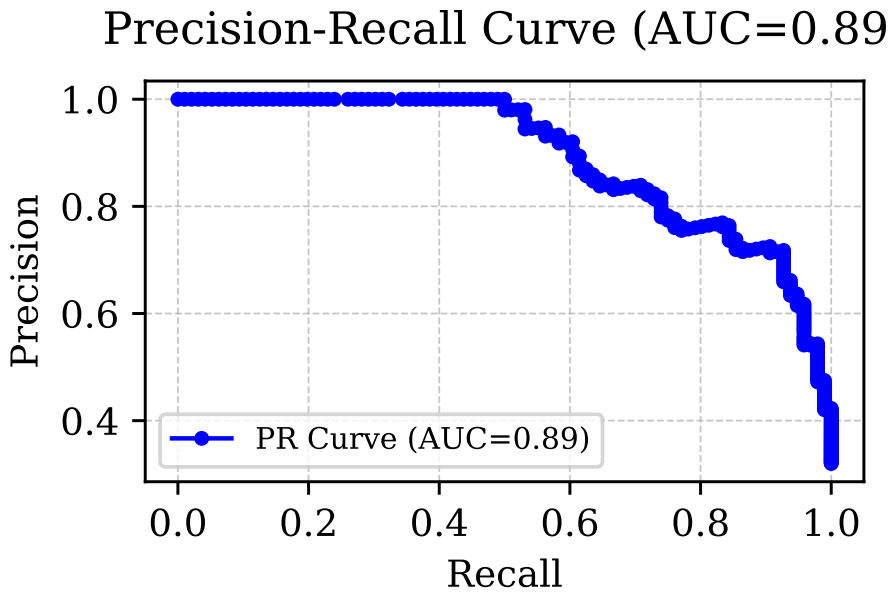

Figure 5.8: Precision-recall curve (AUC-PR = 0.89) for System 1 defect detection.

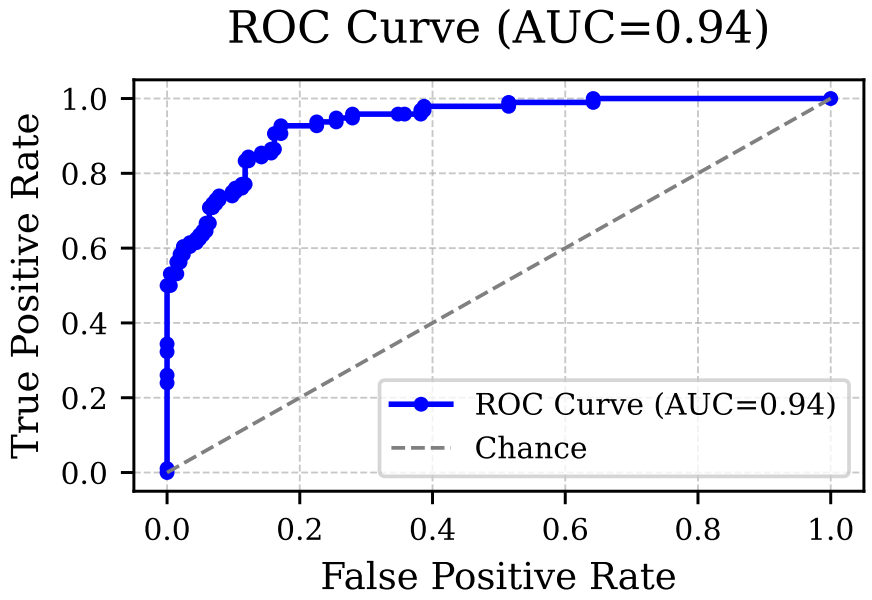

Figure 5.9: ROC curve (AUC-ROC = 0.94) for System 1 defect detection.

Model Performance and Optimized Threshold The updated model achieved an overall accuracy of 86%, with a macro-average F1-score of 85% and a weighted average of 86% (Table 5.3). Precision-recall analysis improved with an optimized threshold of 0.493 (Fig. 5.8), increasing recall for the defect class to 93% while maintaining a high AUC-PR of 0.89. Similarly, ROC analysis indicates a robust AUC-ROC of 0.94 (Fig. 5.9), demonstrating reliable classification performance for both classes.

Table 5.3: Classification Report for System 1 Defect Detection

Class	Precision	Recall	F1-score	Count
0 (Non-defect)	0.96	0.83	0.89	204
1 (Defect)	0.72	0.93	0.81	96
Macro Avg	0.84	0.88	0.85	300
Weighted Avg	0.88	0.86	0.86	300
Accuracy: 0.86				

Confusion Matrix Analysis The confusion matrix (Table 5.4) revealed effective classification, with 88 of 96 defect samples correctly identified. Only 35 non-defect samples were misclassified as defects, highlighting a manageable trade-off between precision and recall.

Table 5.4: Updated Confusion Matrix for System 1 Defect Detection

True/Predicted	Non-defect (0)	Defect (1)
Non-defect (0)	169	35
Defect (1)	8	88

Results and Analysis: System 2 TSV Optimization

System 2 optimization aims to minimize defect rates by balancing TSV depth, deposition rate, etching temperature, and aspect ratio. Using an adaptive CNN-based framework in combination with simulated annealing, the process achieved a defect rate reduction from 10.19% to 0.98%, corresponding to a yield improvement from 89.81% to 99.02%. Feature importance analysis highlighted the relative contributions of Aspect Ratio (26%), Deposition Rate (25%), TSV Depth (24%), and Etching Temperature (24%), emphasizing the interconnected nature of TSV manufacturing.

Convergence Metrics Defect rate convergence across multiple lots (Fig. 5.10) demonstrated consistent reductions post-optimization, underscoring the robustness of the optimization process.

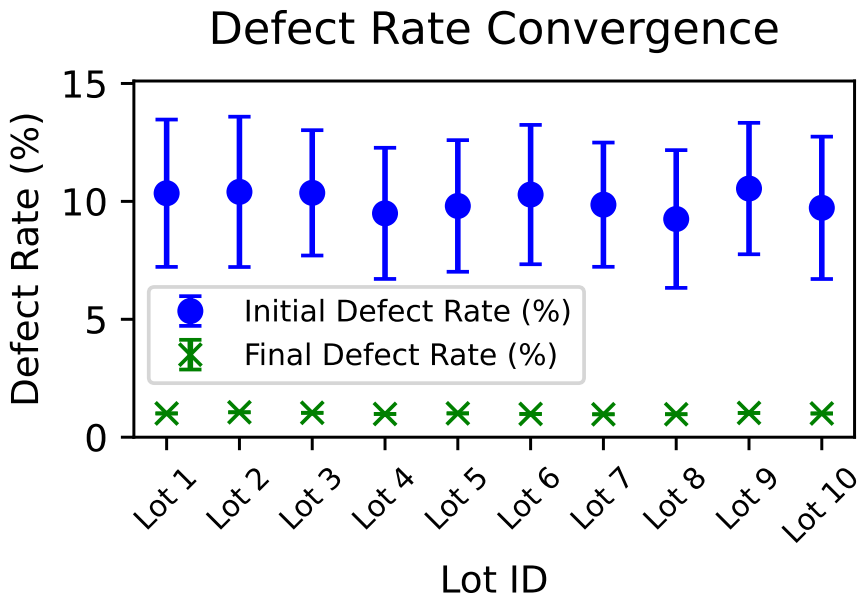

Figure 5.10: Defect rate convergence across lots for System 2.

Parameter Distributions The optimization process resulted in distinct shifts in TSV Depth, Deposition Rate, Etching Temperature, and Aspect Ratio (Figures 5.11 through

5.14). Each parameter distribution includes two green bars, representing stratified clusters optimized for specific design objectives. These adjustments demonstrate the process's capacity to account for diverse TSV configurations and ensure manufacturing consistency.

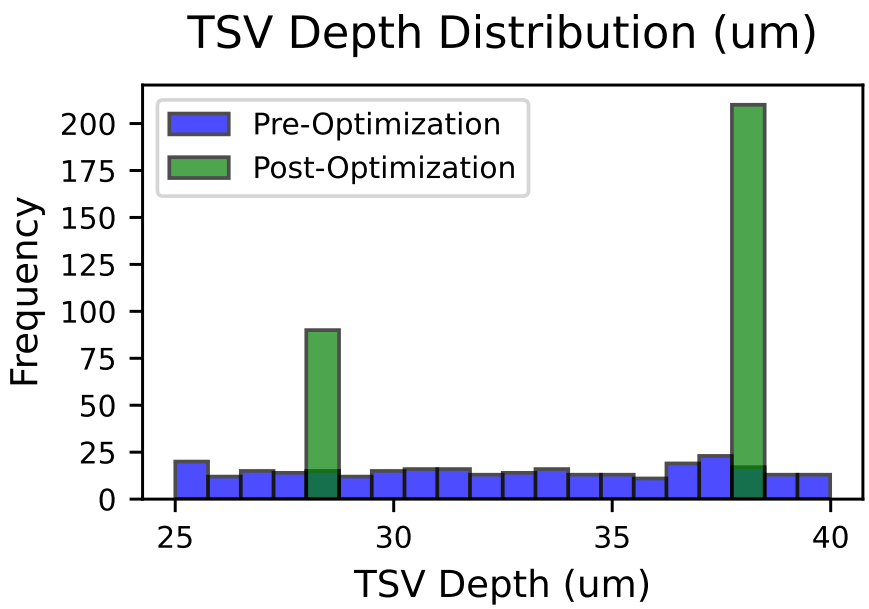

Figure 5.11: Pre- and post-optimization distribution of TSV depth. The mean depth shifted from 32.54 μm to 35.00 μm, reflecting improved trench uniformity.

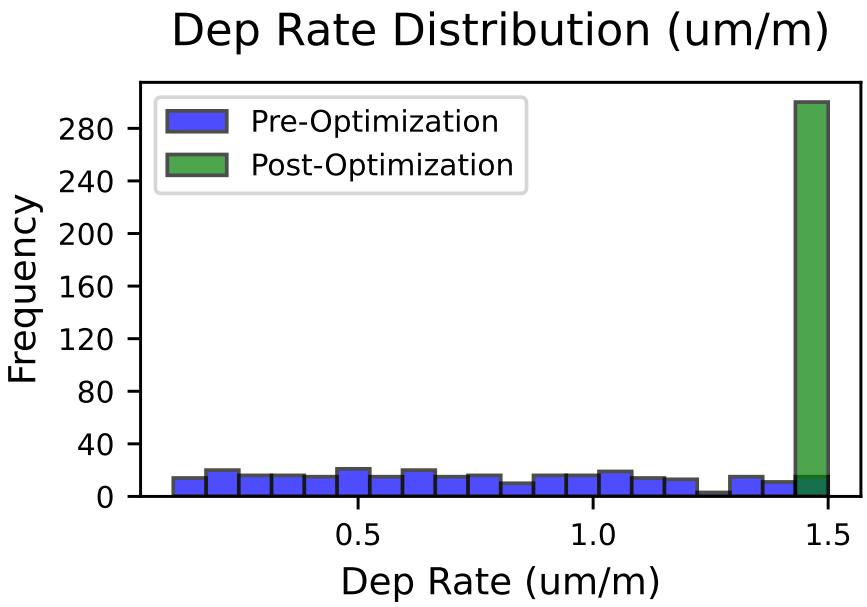

Figure 5.12: Pre- and post-optimization distribution of deposition rate. The mean rate increased from 0.76 μm/m to 1.50 μm/m, indicating enhanced deposition efficiency.

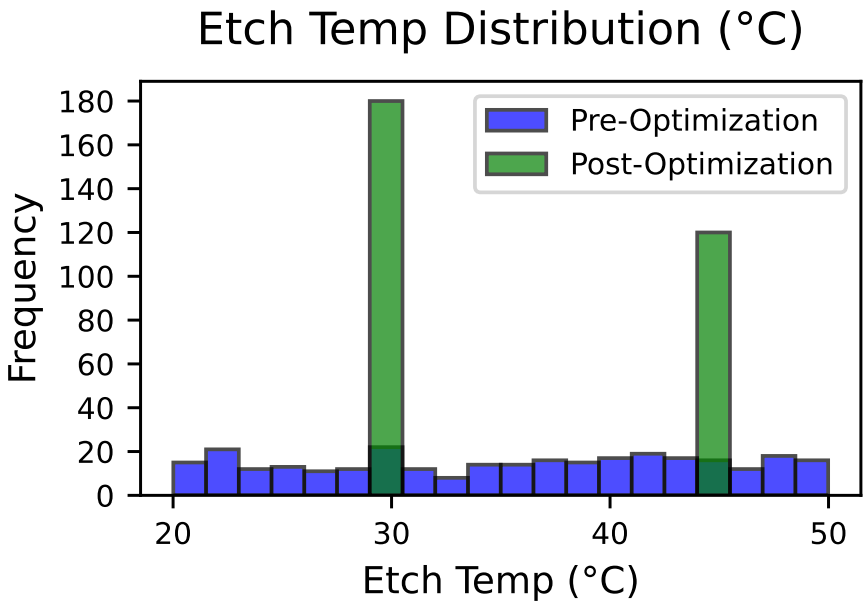

Figure 5.13: Pre- and post-optimization distribution of etching temperature. The mean increased from 35.33°C to 35.97°C, ensuring stable etching conditions.

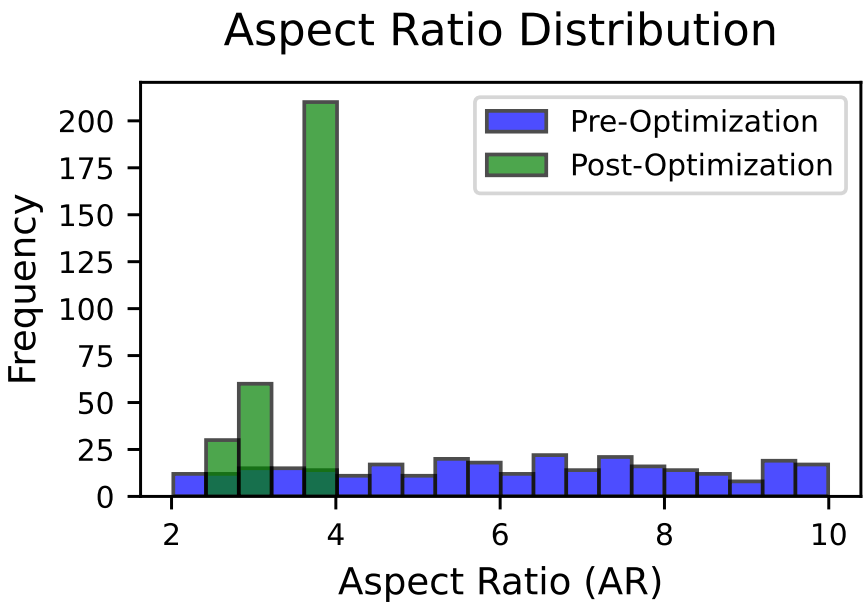

Figure 5.14: Pre- and post-optimization distribution of TSV aspect ratio. The mean decreased from 6.12 to 3.51, reflecting improved TSV uniformity.

Optimization Results The pre- and post-optimization parameter means are summarized in Table 5.5, reflecting significant parameter shifts aligned with manufacturing objectives. The pre-optimization values represent realistic process conditions derived from controlled variability in TSV manufacturing, as outlined in Section 5.2.3. These values emulate parameter ranges typically observed in industrial settings, ensuring that the improvements achieved are grounded in credible baselines rather than artificially exaggerated contrasts.

For TSV Depth, the mean increased from 32.54 μm to 35.00 μm, improving trench uniformity (Fig. 5.11). The two green bars in the post-optimization distribution represent distinct parameter clusters for shallow and deep TSV configurations, balancing mechanical stability and electrical performance. Deposition Rate rose from 0.76 μm/m to 1.50 μm/m (Fig. 5.12), with the dual green bars reflecting distinct process conditions optimized for different TSV dimensions.

Etching Temperature showed a slight increase from 35.33°C to 35.97°C (Fig. 5.13). The dual green bars indicate stratified temperature clusters tailored to achieve etching uniformity

across varying TSV designs. Aspect Ratio decreased from 6.12 to 3.51 (Fig. 5.14), reflecting a shift toward uniform TSV dimensions. The two green bars correspond to low- and high-aspect-ratio clusters, optimizing manufacturing reliability across diverse configurations.

Table 5.5: Pre- and Post-Optimization Means for TSV Parameters

Feature	**Pre-Op Mean**	**Post-Op Mean**
TSV Depth (μm)	32.54	35.00
Deposition Rate (μm/m)	0.76	1.50
Etching Temperature ($^{\circ}C$)	35.33	35.97
Aspect Ratio	6.12	3.51

Feature Importance Analysis Feature importance analysis (Fig. 6.2) highlighted Aspect Ratio (26%), Deposition Rate (25%), TSV Depth (24%), and Etching Temperature (24%) as key factors influencing defect reduction. The nearly equal contributions of these features underscore the multifactorial nature of TSV optimization, emphasizing the importance of balancing all parameters to ensure reliable and efficient manufacturing outcomes.

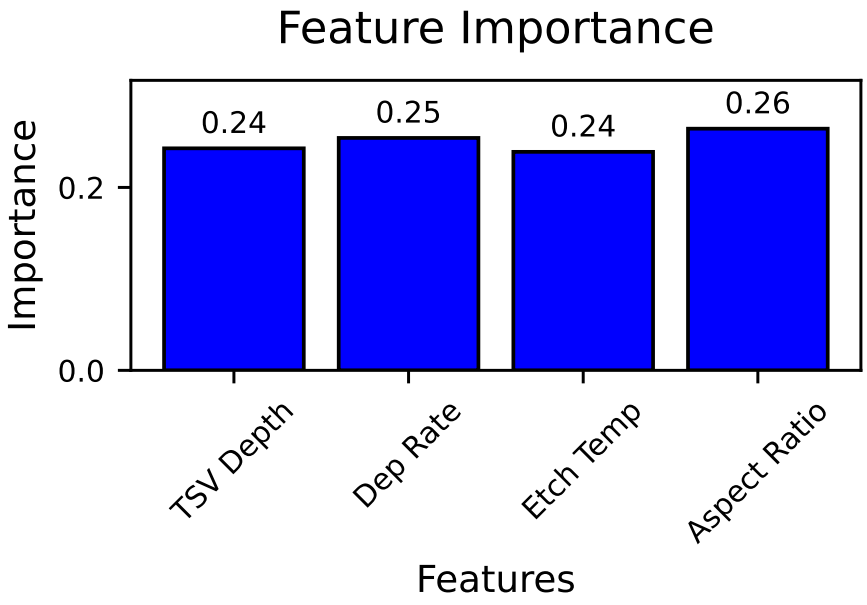

Figure 5.15: Feature importance for System 2 TSV optimization.

Results and Analysis: System 3 Electrical Failure Detection

System 3 uses an LSTM-based architecture to detect electrical failures, analyzing time-series data of resistance (Ω), capacitance (pF), and leakage current (µA). To address class imbalance, SMOTE is applied to augment the minority class (electrical failures), resulting in a balanced dataset. The class distribution before and after applying SMOTE is illustrated in Fig. 5.16, highlighting the effectiveness of SMOTE in equalizing the sample sizes.

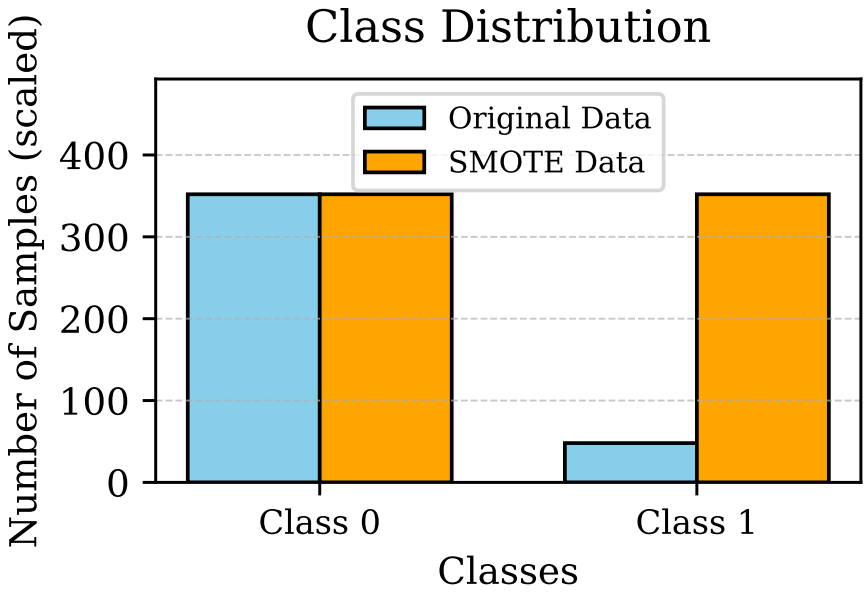

Figure 5.16: Class distribution: Original vs. SMOTE for System 3.

Performance Metrics The model achieves an accuracy of 72.73%, as detailed in Table 5.6. Precision for the failure class (Class 1) reaches 97.62%, indicating minimal false positives, while recall improves to 46.59%, reflecting better identification of failures. The F1-score and ROC-AUC values demonstrate a balanced trade-off between precision and recall.

Table 5.6: Performance Metrics for System 3 Electrical Failure Detection (Class 1: Failures)

Metric	Value
Accuracy	72.73%
Precision (Class 1)	97.62%
Recall (Class 1)	46.59%
F1-Score (Class 1)	63.08%
ROC-AUC	84.34%

Confusion Matrix Analysis The confusion matrix in Table 5.7 shows that 87 non-failure instances (Class 0) are correctly classified as negative, while 41 electrical failure instances (Class 1) are correctly classified as positive. However, 47 failure cases are misclassified as non-failures, highlighting a need for improved recall for the minority class.

Table 5.7: Confusion Matrix for System 3 Electrical Failure Detection

Actual/Predicted	Predicted Negative	Predicted Positive
Actual Negative	87	1
Actual Positive	47	41

SMOTE vs. No SMOTE Comparison The application of SMOTE significantly improves the ability of the model to learn from the minority class (Class 1), representing failures. Without SMOTE, the dataset is highly imbalanced, dominated by the majority class (Class 0), representing non-failures. Fig. 5.16 demonstrates how SMOTE equalizes the class distribution, enabling the model to achieve higher precision for failure detection. Despite the improvement in precision, the recall remains constrained due to the inherent complexity of identifying subtle electrical failures.

Feature Importance Analysis Feature importance analysis (Fig. 5.17) identifies resistance as the most critical feature, followed by capacitance and leakage current. Resistance contributes significantly to failure classification, aligning with its known role in characterizing electrical anomalies.

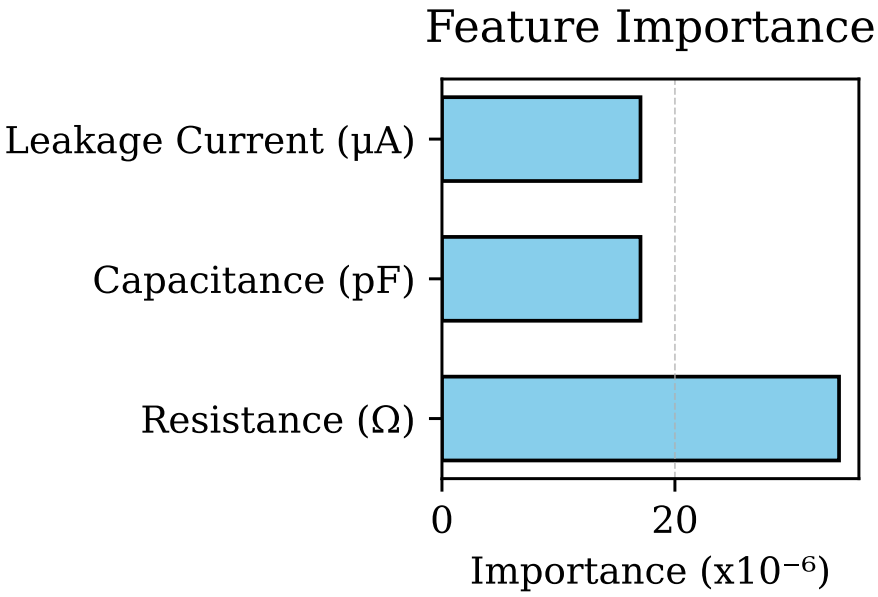

Figure 5.17: Feature importance for System 3 electrical failure detection.

Model Evaluation Curves The ROC curve in Fig. 5.18 illustrates strong discriminative performance, achieving an AUC of 83.39%. The curve highlights high sensitivity at low false-positive rates, critical for minimizing undetected failures. The precision-recall curve in Fig. 5.19 demonstrates high precision at varying recall levels, reflecting the model's effectiveness in reducing false positives.

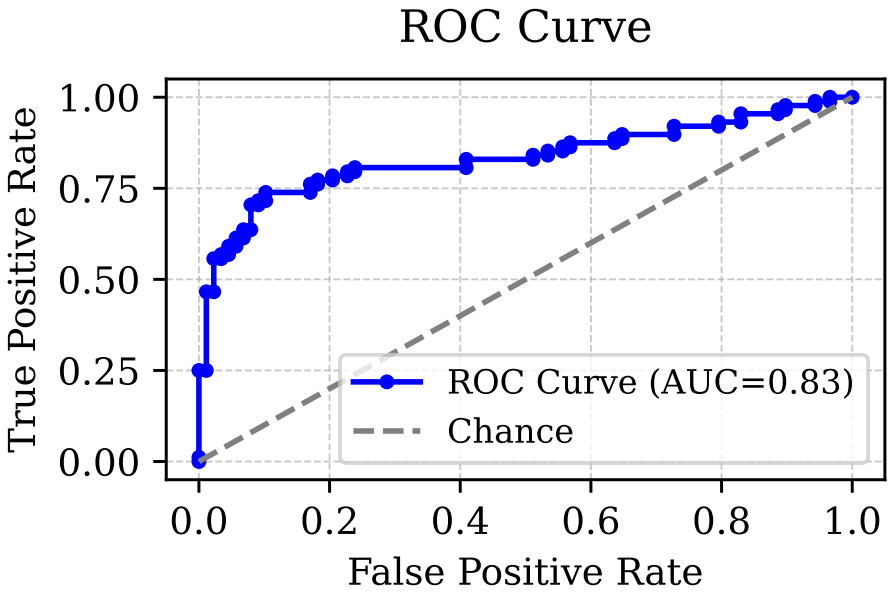

Figure 5.18: ROC curve for System 3 (AUC = 83.39%).

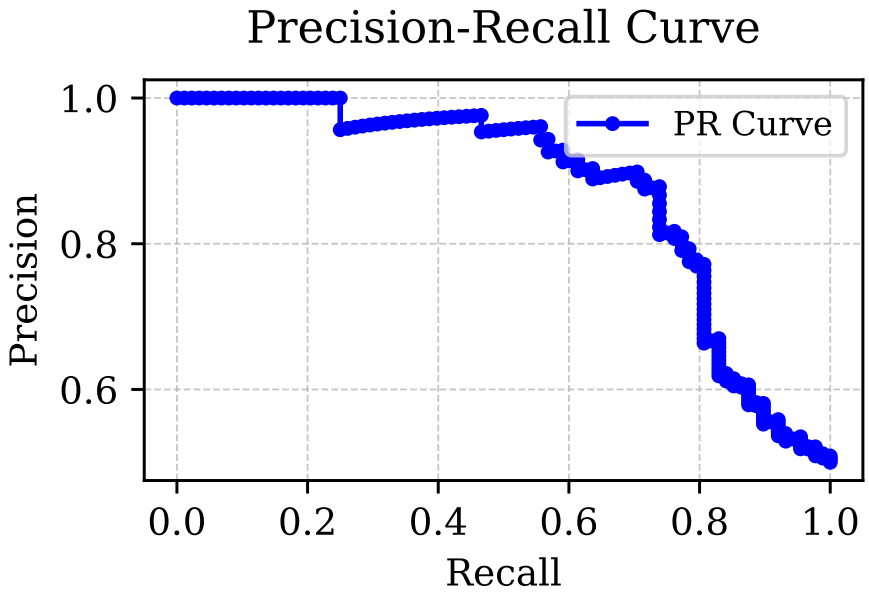

Figure 5.19: Precision-recall curve for System 3.

Training and Validation Performance Training and validation loss curves (Fig. 5.20) show stable convergence over 30 epochs, with training loss stabilizing at approximately 0.10 and validation loss slightly higher at 0.12. This minimal gap suggests effective generalization and no significant overfitting.

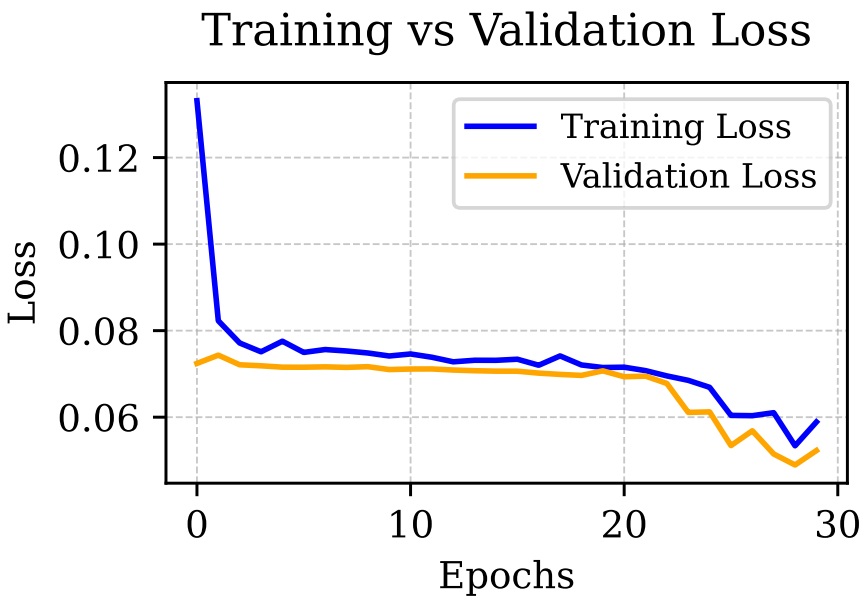

Figure 5.20: Training and validation loss curves for System 3 electrical failure detection.

Results and Analysis: Global System of Systems (SoS)

The integration of defect detection (System 1), TSV optimization (System 2), and electrical fault detection (System 3) within the SoS framework demonstrates measurable improvements in global yield and subsystem performance. Key metrics, including yields and defect rates, are summarized in Table 5.8.

Table 5.8: Global SoS Metrics and System Performance

Metric	Value
System 1 Yield	69.50%
System 2 Yield (Pre-Optimization)	89.99%
System 2 Yield (Post-Optimization)	98.99%
System 3 Yield	72.73%
Global Yield (Pre-Optimization)	77.41%
Global Yield (Post-Optimization)	80.41%
Weighted Global Yield	83.70%
Improvement in Global Yield	3.00%

System 1 and System 3 achieve yields of 69.50% and 72.73%, respectively. System 2 exhibits a yield increase from 89.99% to 98.99% post-optimization, contributing to a global yield improvement from 77.41% to 80.41% (+3.00%). Figure 5.21 highlights subsystem yields and global performance. Additionally, the “Delta Sys2" metric reflects the relative contribution of the optimization in System 2 to defect reduction in the global system, emphasizing its impact.

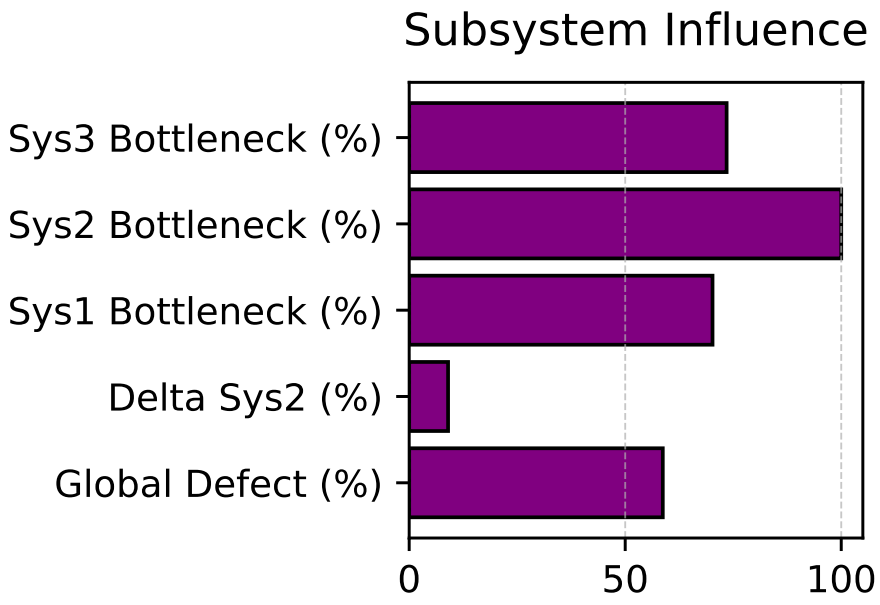

Figure 5.21: System yields and Global SoS yield improvement.

Yield loss analysis (Fig. 5.22) identifies System 1 as the largest contributor (45.00%), followed by System 3 (40.20%) and System 2 (14.80%). Optimization in System 2 significantly reduces its defect impact.

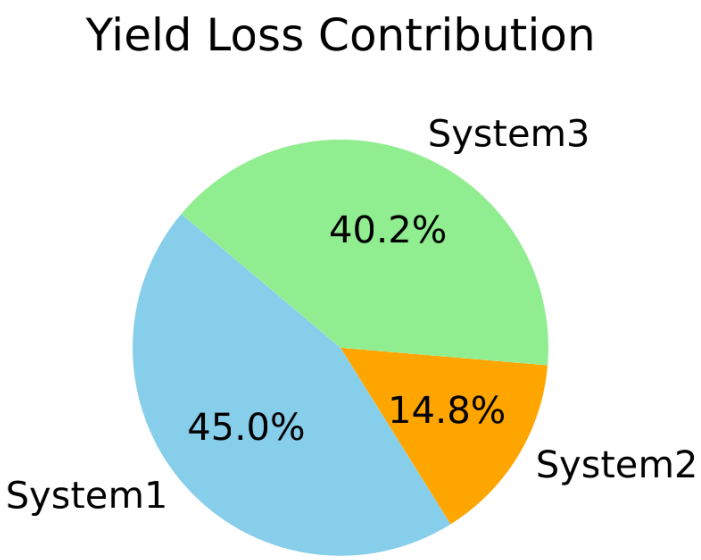

Figure 5.22: System contributions to Global SoS yield loss.

Sensitivity analysis (Fig. 5.23) reveals System 2 as the most critical to global yield variations, followed by System 1 and System 3. This underscores the pivotal role of System 2 in maintaining overall system efficiency.

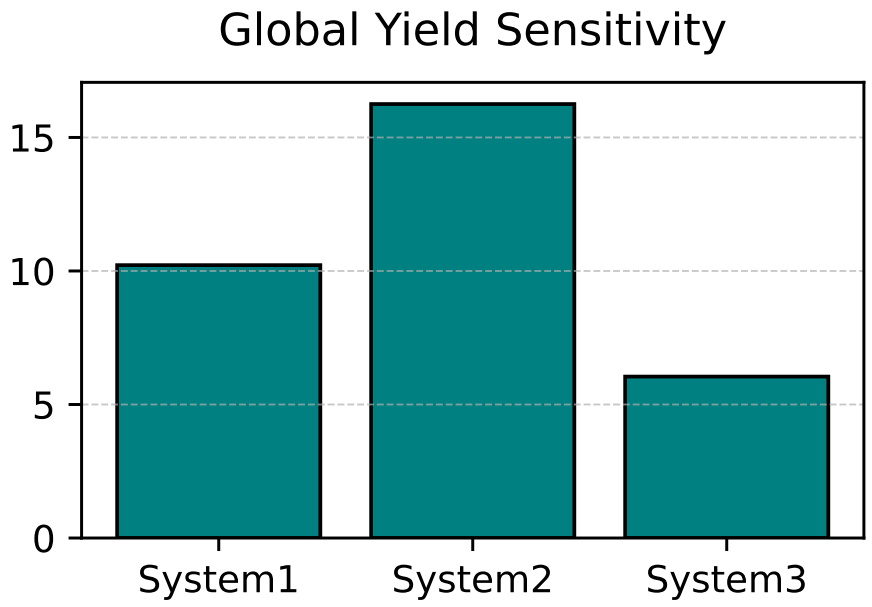

Figure 5.23: System sensitivity to Global SoS yield.

Finally, weighted analysis using defect-adjusted metrics calculates the global yield at 83.70%, further demonstrating the optimization impact of individual subsystems on system-wide performance.

Comparative Performance Summary To provide a holistic evaluation of the AI-driven SoS framework, Table 5.9 presents a comparative summary of key performance metrics across the three subsystems. The defect detection model (System 1) demonstrated high accuracy and recall, particularly benefiting from class balancing techniques. The TSV optimization model (System 2) achieved a significant reduction in defect rates, improving yield from 89.81% to 99.02%. The electrical failure detection model (System 3) exhibited strong precision but highlighted challenges in recall due to the complexity of time-series anomaly detection. The summarized metrics offer insights into the trade-offs and strengths of each AI-driven subsystem within the global SoS framework.

Table 5.9: Comparative Performance Summary Across Subsystems

Metric	Value	System
Accuracy	86.00%	System 1
Recall (Defect Class)	93.00%	System 1
F1-Score (Defect Class)	81.00%	System 1
AUC-ROC	0.94	System 1
Pre-Optimization Yield	89.81%	System 2
Post-Optimization Yield	99.02%	System 2
Defect Rate Reduction	10.19% to 0.98%	System 2
Accuracy	72.73%	System 3
Precision (Failure Class)	97.62%	System 3
Recall (Failure Class)	46.59%	System 3
AUC-ROC	84.34%	System 3

5.2.5 Discussion

Our proposed AI-driven System of Systems (SoS) framework offers significant advancements in optimizing critical stages of 3D-IC manufacturing. By integrating defect detection during die assembly, TSV process optimization, and electrical failure detection, our framework aligns subsystem outputs with system-wide performance objectives.

Key Findings and Implications

Defect detection during the bumped die attachment stage was pivotal in ensuring alignment and interconnection quality. The Random Forest Classifier demonstrated high inter-

pretability and robust feature handling, enhancing defect identification. This improvement reduced downstream variability, benefiting the TSV and electrical failure detection subsystems.

Our investigation involving System 1 highlighted the challenges posed by class imbalance in defect detection tasks. Despite class-weighted learning and SMOTE, achieving high recall for the minority class remained difficult. SMOTE improved recall (6% to 17%) and F1-score (10% to 14%) for the minority class but reduced overall accuracy from 87% to 76%. These trade-offs emphasize the need for alternative strategies, such as cost-sensitive learning or domain-specific data augmentation, to enhance performance in imbalanced datasets. Feature importance analysis identified Reflow Temperature, Reflow Pressure, and Bump Height as key contributors, highlighting potential areas for process refinement.

In System 2, the adaptive TSV optimization model achieved substantial improvements, reducing the defect rate from 10.19% to 0.98%, with the global yield increasing from 89.81% to 99.02%. Parameter distributions highlight the process's capability to accommodate manufacturing tolerances. For instance, the dual green bars in TSV Depth (Fig. 5.11) and Deposition Rate (Fig. 5.12) reflect stratified clusters, representing optimizations for varying TSV geometries. Similarly, the Aspect Ratio distribution (Fig. 5.14) showcases distinct clusters that enhance dimensional uniformity and reliability, with the mean decreasing from 6.12 to 3.51. For Etching Temperature (Fig. 5.13), stratified clusters indicate tailored process conditions, ensuring uniform etching performance across diverse configurations.

These findings validate the potential of AI-driven techniques to optimize interconnected processes and improve TSV reliability [85]. The aspect ratio improvements align with industrial targets, highlighting the interconnected nature of optimization, where parameter adjustments collectively contribute to process stability.

The LSTM-based electrical failure detection achieved an accuracy of 72.73% and an AUC-ROC score of 0.84, aligning with prior research on leveraging time-series models for

fault detection [68]. This validates its applicability to dynamic manufacturing environments, effectively identifying temporal dependencies in electrical failures.

Computational Feasibility and Scalability Implications

The computational complexity assessment confirms that the proposed AI-driven decision-making framework is suitable for real-time semiconductor manufacturing. Training durations for the three subsystems ranged from 15 to 22 minutes on a CPU-based system (Intel i7, 32GB RAM), ensuring feasibility within standard engineering workflows. Once trained, all models performed inference in milliseconds, enabling real-time decision-making for defect detection, process optimization, and electrical test validation.

System 1, responsible for defect detection, utilized an XGBoost classifier optimized with Optuna, requiring 15.4 minutes for training. Its inference speed enabled rapid classification, facilitating on-the-fly defect identification. System 2, which focused on TSV process optimization, combined dual annealing-based optimization with an adaptive CNN-based weighting mechanism, converging within 22 minutes. This demonstrates its capability to dynamically refine process parameters in semiconductor fabrication. System 3, designed for electrical test fault detection, leveraged an LSTM-based time-series classification model. The training phase lasted 17.5 minutes with SMOTE-balanced data, and inference was completed within milliseconds, ensuring timely fault identification.

Key System-Level Insights and Performance Evaluation

The AI-driven System-of-Systems framework substantially enhanced defect detection, process optimization, and fault classification in semiconductor manufacturing. System 1 revealed that while SMOTE improved recall and F1-score for minority defect classes, it slightly reduced overall accuracy, emphasizing the trade-offs associated with class imbalance mitigation techniques. Feature importance analysis identified critical process parameters such as reflow temperature, reflow pressure, and bump height, which directly affect interconnection quality.

System 2's TSV optimization model significantly lowered defect rates from 10.19% to 0.98%, raising the overall yield to 99.02%. The integration of adaptive CNN-based weighting ensured parameter tuning specific to TSV geometries, thereby improving manufacturing uniformity. The optimized TSV depth and deposition rates contributed to enhanced process stability, reinforcing the framework's adaptability to evolving manufacturing constraints.

System 3's LSTM-based fault detection model achieved an accuracy of 72.73% and an AUC-ROC score of 0.84, effectively capturing temporal dependencies in electrical failures. The use of time-series modeling enabled early anomaly detection, supporting proactive test validation strategies. These findings validate the AI-driven framework's ability to integrate learning-based decision-making across semiconductor manufacturing stages, improving both precision and adaptability in defect detection and fault classification.

Scalability Considerations and Future Enhancements

The scalability of the framework is reinforced by its ability to leverage computational efficiencies through parallelization, multi-core execution, and distributed processing. XGBoost and dual annealing optimization techniques can be parallelized to reduce training times, while deep learning components, including CNN-based adaptive weighting and LSTM time-series classification, can be further accelerated using GPU or TPU environments. Additionally, batch processing strategies can be employed to enhance inference efficiency, facilitating high-throughput applications in large-scale semiconductor manufacturing.

Recent advancements in deep learning frameworks have introduced segmentation models optimized for multi-scale feature extraction and generative AI-based defect modeling, which could enhance defect detection accuracy by leveraging cross-domain knowledge transfer [86]. Hybrid models that integrate time-series learning with uncertainty quantification are demonstrating improved predictive reliability in failure classification, offering potential enhancements for adaptive thresholding and anomaly detection [17]. Furthermore, generative approaches for class imbalance mitigation have shown effectiveness in improving fault classification robustness, particularly in low-defect-rate scenarios [36].

Future research will focus on enhancing real-time adaptability by integrating streaming data processing techniques and distributed training strategies. Extending the framework to incorporate additional semiconductor process steps, such as environmental monitoring and contamination detection, will further validate its robustness and applicability across broader manufacturing environments.

Comparison with Prior Literature

Our study extends existing research by integrating subsystem optimizations into a unified SoS framework. Unlike traditional methods focusing on isolated stages, our iterative optimization loop dynamically adjusts subsystem parameters, aligning them with global objectives. This expands upon methodologies that emphasize subsystem interdependencies [20].

The use of structured feature data during die assembly, rather than image-based detection, enhances scalability and adaptability. Our results demonstrate the efficacy of data-driven strategies in addressing challenges such as real-time decision-making and interdependency modeling, which remain underexplored in prior work [40].

Advancements in AI-driven optimization in other domains, such as circular economy digital marketplaces [20], further underscore the transformative potential of data-driven frameworks. For instance, reinforcement learning and gradient boosting regressor models have demonstrated success in managing complex interdependencies and adapting to dynamic conditions in resource allocation and demand-supply matching tasks [20]. These insights reinforce the scalability and adaptability of AI-based approaches, providing inspiration for addressing the inherent complexities of 3D-IC manufacturing.

Practical Contributions

The modular design of our framework enables scalability, allowing the inclusion of additional subsystems like material handling and thermal management. This adaptability positions the approach as a valuable solution for evolving manufacturing landscapes, improving reliability and efficiency.

The unified optimization loop bridges the gap between subsystem- and system-level objectives. Dynamically refining parameters such as TSV depth and electrical failure thresholds minimizes defect propagation and enhances yield. By incorporating stratified optimizations reflected in the dual green bars, the framework highlights its adaptability to specific manufacturing tolerances, enhancing robustness.

Limitations and Future Work

While our framework demonstrated significant improvements, limitations remain. Challenges in defect detection underscore the need for more robust methods to address class imbalance. Future work could explore ensemble learning, cost-sensitive algorithms, and domain-specific data augmentation to further improve detection.

The current implementation relies on iterative optimization rather than real-time adaptability. Future research should incorporate streaming data and real-time optimization to handle dynamic manufacturing variations effectively.

Scalability remains a critical focus. Incorporating additional subsystems, such as environmental monitoring and material logistics, could validate the versatility of our framework. Addressing domain-specific challenges, including TSV contamination and failure propagation, offers opportunities for further refinement and broader applicability.

5.2.6 Summary

Three-dimensional integrated circuit (3D-IC) manufacturing is a complex, multi-stage process characterized by intricate interdependencies among subsystems. This case study proposed a Machine Learning-driven optimization framework using a System of Systems (SoS) approach to improve 3D-IC manufacturing outcomes, with a focus on early-stage defect detection during die placement and stacking. By addressing defects at their origin, the framework minimizes error propagation, thereby enhancing system-wide yield and reliability.

The integration of Random Forest Classifiers for defect detection, Convolutional Neural Networks (CNNs) for adaptive Through-Silicon Via (TSV) optimization, and Long Short-

Term Memory (LSTM) networks for electrical failure detection demonstrated measurable improvements in process uniformity, yield, and predictive performance. Notably, the framework dynamically aligned subsystem outputs with global objectives, highlighting the pivotal role of assembly-stage defect detection in driving system-wide enhancements.

Key contributions include the development of a unified optimization loop that bridges subsystem-level decisions with overarching system-wide objectives. These results validate the scalability and adaptability of ML-driven frameworks in high-complexity manufacturing environments and underscore the importance of early-stage diagnostics in establishing downstream process robustness.

Future research directions include extending this SoS framework to additional semiconductor subsystems such as material logistics, thermal management, and contamination monitoring, as well as enhancing real-time adaptability through streaming data and distributed learning.

While this section focused on early-stage yield optimization through integrated machine learning subsystems, the next section explores decentralized coordination in semiconductor manufacturing using hierarchical reinforcement learning. This continuation extends the SoS paradigm into dynamic, agent-based decision environments.

5.3 Hierarchical Reinforcement Learning for Decision-Making in Semiconductor Manufacturing

5.3.1 Problem Overview and Context

System-of-Systems (SoS) environments often involve decentralized, independently operated subsystems that must collaborate toward shared global goals—particularly in domains like semiconductor manufacturing, where operational complexity, dynamic constraints, and real-time decision requirements pose significant coordination challenges. These challenges

become particularly pronounced in resource allocation, process optimization, and scheduling tasks across constituent manufacturing units, each with partially autonomous behavior.

Traditional optimization techniques often fall short in managing the trade-offs between local autonomy and global coordination, especially when decision-making must adapt in real time to fluctuating workloads, system faults, or supply chain disruptions. In such scenarios, effective coordination mechanisms are critical to avoiding conflicts, resolving interdependencies, and ensuring that local actions contribute constructively to system-wide performance.

To address these gaps, this case study explores a hierarchical reinforcement learning (HRL) framework that integrates multi-agent systems (MAS) with structured reward mechanisms across three levels: global, main-agent, and sub-agent. This hierarchy is designed to align local decision-making with overarching system objectives while preserving agent-level flexibility. The approach is demonstrated in the context of semiconductor manufacturing, where agents represent process control units, equipment schedulers, and yield monitoring systems working in tandem.

By embedding AI-driven reward shaping and reinforcement learning into this hierarchical architecture, the proposed framework enables scalable, adaptive coordination for complex SoS decision environments.

5.3.2 Research Objectives and Questions

The primary objective of this case study is to develop and evaluate a hierarchical reinforcement learning (HRL) framework for coordinating decision-making across decentralized agents in semiconductor manufacturing modeled as a System of Systems (SoS). The proposed architecture integrates global, main-agent, and sub-agent reward structures to align local policies with system-wide goals such as yield optimization, resource efficiency, and operational resilience.

Specifically, this case study addresses the following research objectives:

- **Objective 1:** To design a hierarchical reward mechanism that integrates global, main-agent, and sub-agent levels of control in a semiconductor SoS, enabling multi-scale decision-making.
- **Objective 2:** To enable decentralized reinforcement learning agents to learn local policies that are sensitive to both subsystem constraints and global optimization objectives.
- **Objective 3:** To evaluate the effectiveness of inter-agent cooperation and conflict resolution mechanisms in improving coordination, reducing reward disparity, and enhancing collective system performance.
- **Objective 4:** To assess the scalability and adaptability of the proposed framework under dynamic conditions, including fluctuating workloads and coordination demands.

Based on these objectives, the following research questions guide the investigation:

- **RQ1:** How can hierarchical reinforcement learning be structured to support scalable, decentralized decision-making in semiconductor SoS environments?
- **RQ2:** What impact does reward decomposition across global, main-agent, and sub-agent levels have on agent coordination, system responsiveness, and overall performance?
- **RQ3:** Can hierarchical reward shaping and inter-agent cooperation mechanisms reduce trade-offs between local efficiency and global optimization?
- **RQ4:** To what extent can the proposed HRL framework maintain robust coordination and performance under dynamic, multi-agent manufacturing scenarios?

5.3.3 Methodology

This section outlines the hierarchical reinforcement learning (HRL) framework developed to enable decentralized optimization and coordination in semiconductor manufacturing modeled as a System of Systems (SoS). The proposed framework structures decision-making across three levels: Global System Objectives, Constituent Systems (Main Agents), and

Sub-Agents. Each level is associated with a distinct reward formulation, facilitating policy learning and inter-agent cooperation through multi-agent reinforcement learning (MARL).

Figure 5.24 depicts the hierarchical structure used in the case study, where sub-agents perform local tasks (e.g., tool operation, inspection), main agents represent constituent systems (e.g., fabrication, testing), and the global layer governs system-wide objectives such as yield maximization and cost minimization.

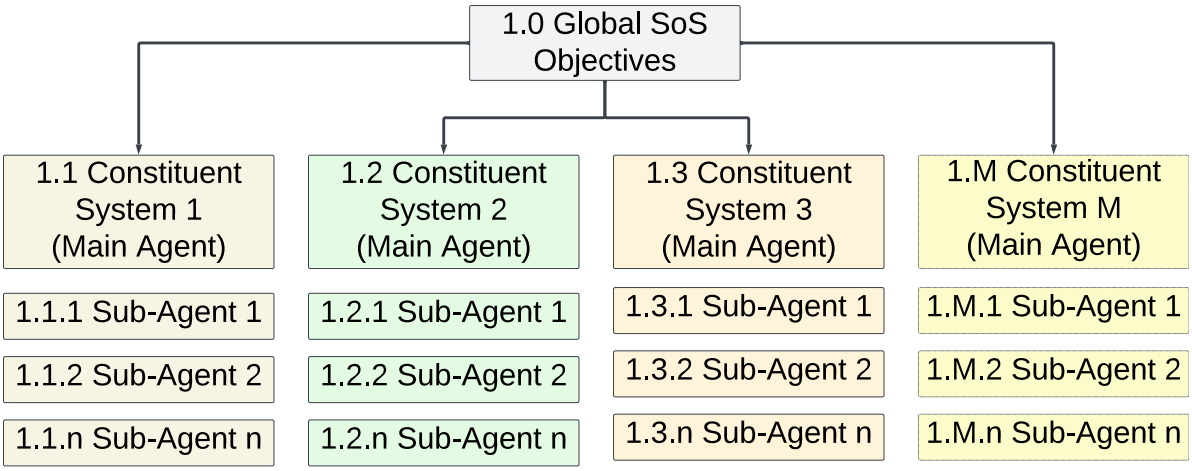

Figure 5.24: Hierarchical System of Systems (SoS) with Global Objectives, Constituent Systems (Main Agents), and Sub-Agents.

At the global level, the reward is structured to balance both intrinsic subsystem performance and inter-agent cooperation:

$$R_{\text{global}} = \sum_{j=1}^{M} w_j (R_{j,\text{intrinsic}} + r_{j,\text{inter}}) \tag{5.20}$$

where M is the number of main agents, w_j is the weight for main agent j, $R_{j,\text{intrinsic}}$ represents the intrinsic reward of agent j, and $r_{j,\text{inter}}$ denotes the inter-agent cooperation reward [87]. Cooperation is modeled as:

$$r_{j,\text{inter}} = \sum_{j' \neq j} \gamma_{j,j'} I_{j,j'} \tag{5.21}$$

where $I_{j,j'}$ is the interaction metric between agents j and j', and $\gamma_{j,j'}$ is the cooperation weight.

To manage disparities in reward distribution, a conflict resolution term is introduced:

$$R_{\text{conflict}} = \delta \sum_{j=1}^{M} \sum_{j' \neq j} |R_{j,\text{total}} - R_{j',\text{total}}| \tag{5.22}$$

where δ is the conflict resolution weight. The complete global reward thus incorporates both cooperative gains and conflict penalties:

$$R_{\text{global,total}} = R_{\text{global}} + R_{\text{conflict}} \tag{5.23}$$

At the **Main Agent level**, rewards incorporate both local performance and global objectives:

$$R_{j,\text{total}} = R_{j,\text{intrinsic}} + R_{j,\text{global}} \tag{5.24}$$

where $R_{j,\text{global}}$ reflects contributions from global objectives. Local rewards at the sub-agent level are aggregated as:

The intrinsic reward for each subsystem is derived by aggregating contributions from its sub-agents:

$$R_{j,\text{intrinsic}} = \sum_{k=1}^{n_j} (r_{k,j,\text{local}} + r_{k,j,\text{subsystem}}) \tag{5.25}$$

where n_j is the number of sub-agents within main agent j, and $r_{k,j,\text{local}}$ denotes the local reward of sub-agent k.

At the sub-agent level, each agent's total reward integrates local, subsystem, global, and coordination elements:

$$r_{k,j,\text{total}} = r_{k,j,\text{local}} + r_{k,j,\text{subsystem}} + r_{k,j,\text{global}} + r_{k,j,\text{coordination}} \tag{5.26}$$

Coordination reward is shaped by interactions among peer agents within the same subsystem:

$$r_{k,j,\text{coordination}} = \alpha_k \sum_{k' \neq k} C_{k,k'} \tag{5.27}$$

The global reward influence on each sub-agent is scaled by its alignment factor:

$$r_{k,j,\text{global}} = \lambda_{k,j} R_{\text{global}} \tag{5.28}$$

Multi-Agent Reinforcement Learning (MARL) Integration

Reinforcement learning formalizes agent behaviors within the hierarchical framework [88, 89]. Each agent's policy maps states to actions:

$$\pi_i(a_i|s_i) = P(a_i|s_i), \tag{5.29}$$

where $P(a_i|s_i)$ represents the probability distribution of actions given the state [90]. The joint state and action spaces are:

$$S = \{s_1, s_2, \ldots, s_n\}, \tag{5.30}$$

$$A = \{a_1, a_2, \ldots, a_n\}, \tag{5.31}$$

where S and A encapsulate states and actions for all agents. The value function estimates the expected return starting from state s under policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right] \tag{5.32}$$

where γ is the discount factor for future rewards. The Bellman update expresses this value recursively in terms of actions and successor states:

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{\pi}(s')\right] \tag{5.33}$$

These equations are foundational to reinforcement learning, as described in [27]. To improve the policy, gradients of the expected return with respect to policy parameters are computed:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\pi}\left[\nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s,a)\right] \tag{5.34}$$

where θ are the policy parameters [91].

Summary of Reward Mechanisms

Table 5.10 summarizes the formulation of rewards across hierarchy levels.

Table 5.10: Summary of Reward Mechanisms

Level	**Variable**	**Formula**
Global SoS	R_{global}	$\sum_{j=1}^{M} w_j (R_{j,\text{intrinsic}} + r_{j,\text{inter}})$
Global SoS	$r_{j,\text{inter}}$	$\sum_{j' \neq j} \gamma_{j,j'} I_{j,j'}$
Global SoS	R_{conflict}	$\delta \sum \lvert R_{j,\text{total}} - R_{j',\text{total}} \rvert$
Main Agent	$R_{j,\text{total}}$	$R_{j,\text{intrinsic}} + R_{j,\text{global}}$
Main Agent	$R_{j,\text{intrinsic}}$	$\sum_{k=1}^{n_j} (r_{k,j,\text{local}} + r_{k,j,\text{subsystem}})$
Sub-Agent	$r_{k,j,\text{total}}$	$r_{k,j,\text{local}} + r_{k,j,\text{subsystem}} + r_{k,j,\text{global}}$
Sub-Agent	$r_{k,j,\text{coordination}}$	$\alpha_k \sum_{k' \neq k} C_{k,k'}$

5.3.4 Results and Analysis

The System of Systems (SoS) framework consists of a Global Agent overseeing two Main Agents: Agent 1 (Wafer Fabrication Agent) and Agent 2 (Defect Detection Agent). Each Main Agent has two Sub-Agents, where Sub-Agent 1 (Etch Platform) and Sub-Agent 2 (Deposition Platform) belong to Agent 1, while Sub-Agent 1 (Inspection Sub-Agent) and Sub-Agent 2 (Electrical Failure Sub-Agent) belong to Agent 2. This hierarchical structure aligns with the agent numbering used in the plots, ensuring consistency in coordination and cooperation analysis. The agents' behaviors and interactions are guided by the hierarchical reward framework described earlier. Simulation variables and their respective ranges are outlined in Table 5.11 to provide a clear basis for replicating the scenario.

Table 5.11: Simulation Variables and Ranges

Variable	**Description**	**Range/Value**
w_j	Global weight for main agent j	$[0.4, 0.6]$
γ	Discount factor for future rewards	$[0.8, 0.99]$
δ	Conflict resolution weight	$[0.1, 0.5]$
α_k	Coordination weight for sub-agent k	$[0.1, 0.5]$
$\lambda_{k,j}$	Global contribution weight for sub-agent k	$[0.1, 0.3]$
$r_{k,j,\text{local}}$	Local reward for sub-agent k	$[50, 100]$
$C_{k,k'}$	Coordination metric between sub-agents k, k'	$[0, 1]$
$I_{j,j'}$	Interaction metric between main agents j, j'	$[0, 1]$
n_j	Number of sub-agents in main agent j	2 (fixed)
M	Number of main agents	2 (fixed)
$s_{k,j}$	State of sub-agent k	Episode context
$a_{k,j}$	Action by sub-agent k	Depends on s

Global and Agent-Level Reward Trends

Figure 5.25 illustrates the reward dynamics across the global, main agent, and sub-agent levels over 500 episodes. The global reward fluctuates between 160 and 180, averaging approximately 170, reflecting the framework's robustness in maintaining alignment with overarching system goals.

The Wafer Fabrication Agent exhibits an average reward around 155, indicating stable operations supported by sub-agents like the Etch Platform. In contrast, the Defect Detection Agent averages 150, with greater variability, highlighting the challenges of tasks such as electrical fault detection.

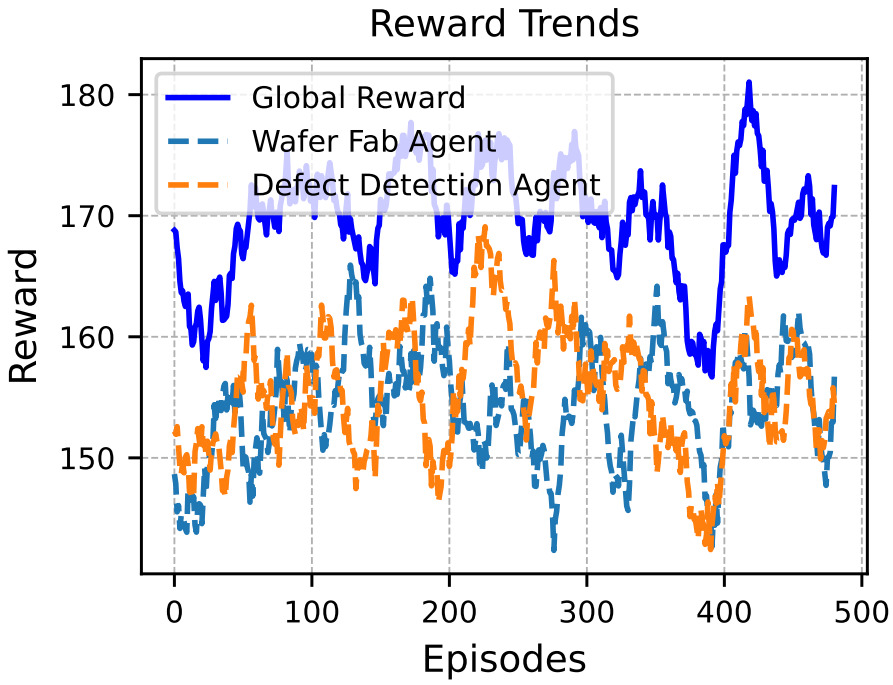

Figure 5.25: Reward Trends Across Global, Wafer Fabrication, and Defect Detection Agents.

Cooperation and Coordination Metrics

Fig 5.26 shows cooperation metrics between main agents (Wafer Fabrication and Defect Detection), while Fig 5.27 depicts sub-agent coordination. The cooperation metric fluctuates

between 0.15 and 0.32, averaging around 0.25, reflecting moderate collaboration. Both main agents exhibit dynamic cooperation without clear dominance, suggesting adaptive behavior.

Sub-agent coordination trends show the Etch Platform peaking at 0.35, outperforming the Deposition Platform, which remains below 0.25. Within the Defect Detection Agent, coordination between Inspection and Electrical Failure sub-agents remains comparable, with Inspection occasionally leading. These results validate the reward mechanism's role in task prioritization.

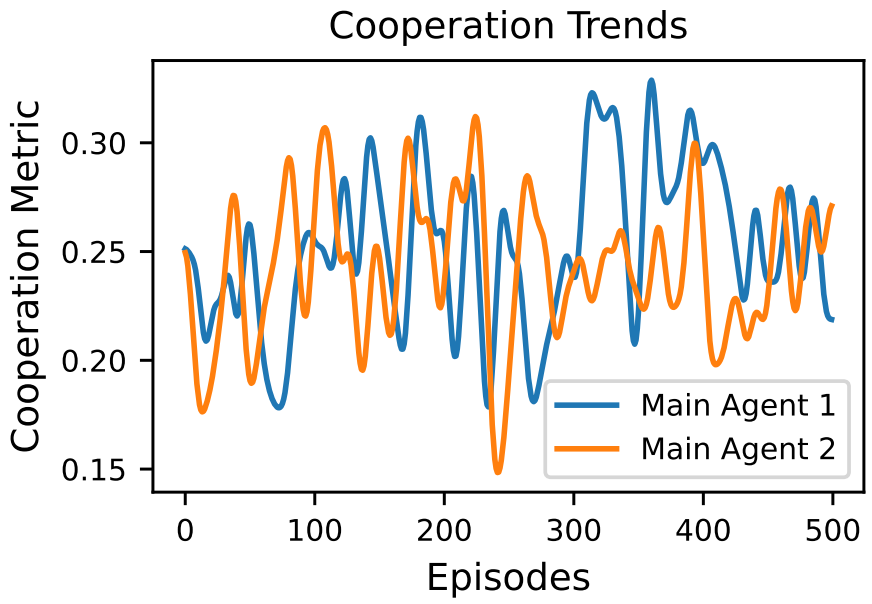

Figure 5.26: Cooperation Trends Between Main Agents.

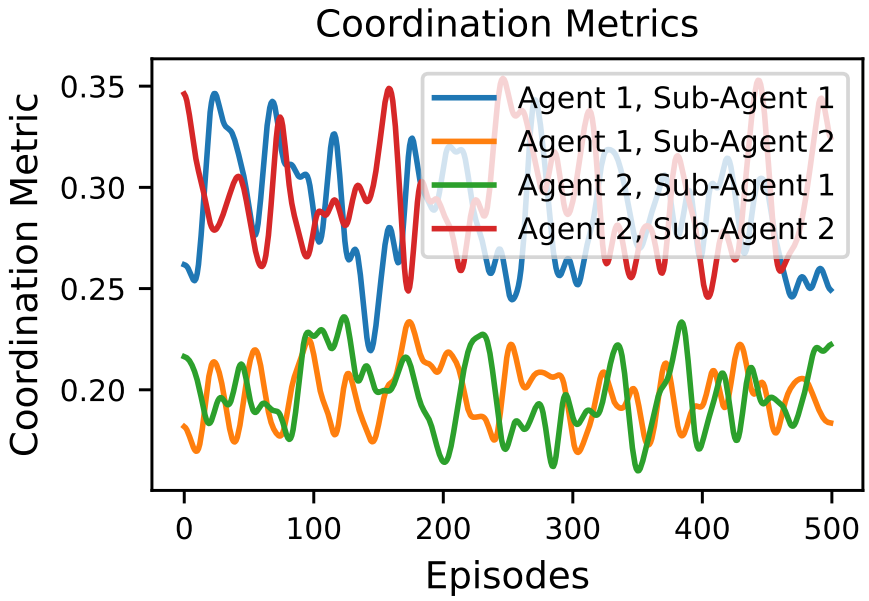

Figure 5.27: Coordination Metrics Across Sub-Agents.

Sensitivity Analysis

Fig 5.28 illustrates the impact of coordination weight (α) and discount factor (γ) on average rewards. At $\gamma = 0.99$, rewards peak at 55 when $\alpha = 0.5$, indicating that higher coordination enhances efficiency. For $\gamma = 0.90$ and $\gamma = 0.94$, rewards peak at $\alpha = 0.2$ and $\alpha = 0.3$, respectively, before declining, suggesting diminishing returns. In contrast, $\gamma = 0.85$ remains stable beyond $\alpha = 0.3$, while $\gamma = 0.80$ follows the lowest trajectory, showing limited sensitivity to coordination tuning.

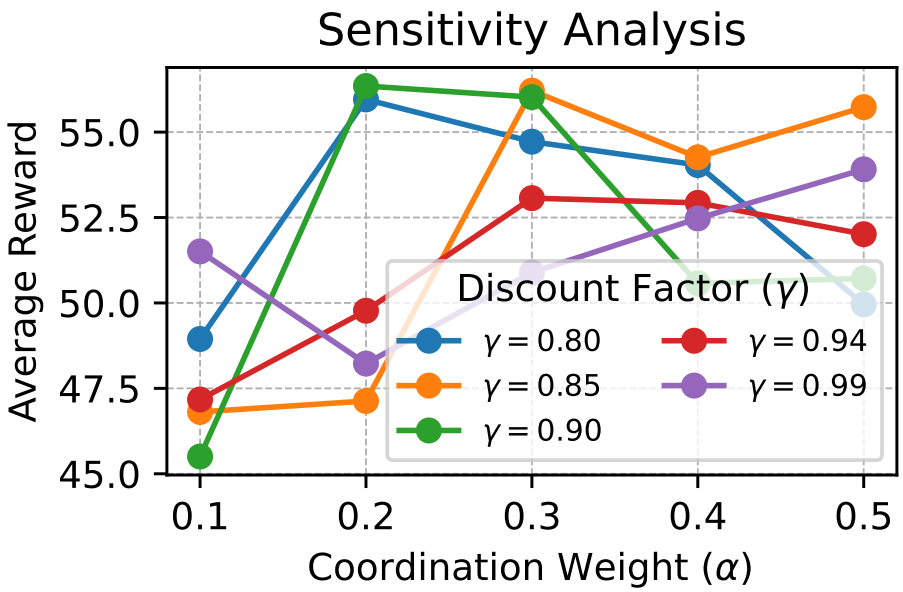

Figure 5.28: Sensitivity Analysis of Coordination Weight (α) and Discount Factor (γ).

Conflict Metrics

Conflict metrics, as shown in Fig 5.29, capture reward discrepancies between main agents. Periodic spikes, peaking at 14, highlight episodes of misalignment due to resource contention or overlapping objectives. The gradual reduction in these peaks demonstrates the ability of our framework to resolve conflicts effectively, ensuring equitable resource distribution and alignment of agent objectives.

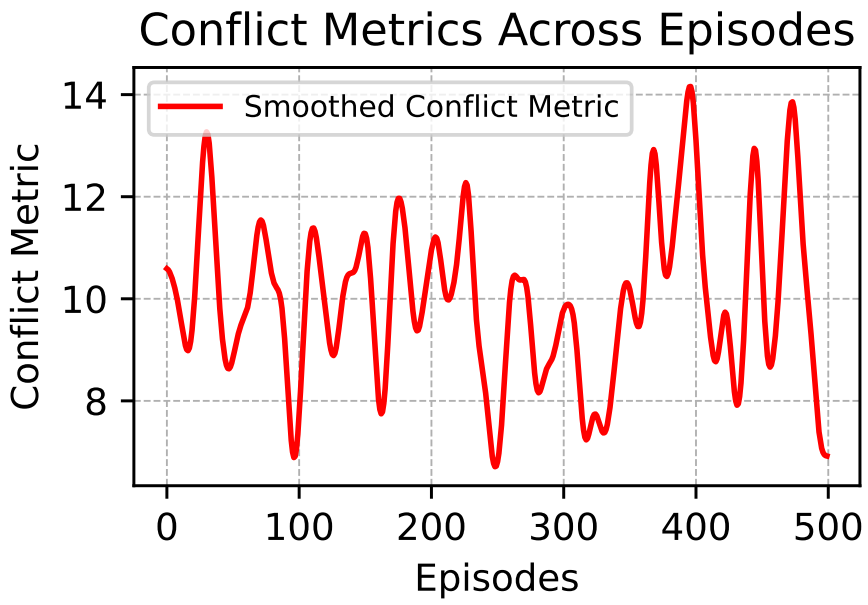

Figure 5.29: Smoothed Conflict Metrics Across Episodes.

Agent and Sub-Agent Performance

Fig 5.30 shows the Wafer Fabrication Agent's sub-agents, where the Etch Platform achieves rewards fluctuating between 78 and 82, while the Deposition Platform remains between 68 and 72. Similarly, Fig 5.31 shows the Defect Detection Agent's sub-agents, with Inspection maintaining rewards between 72 and 77, outperforming Electrical Failure, which fluctuates between 63 and 67, by an average margin of 8–12 points. These performance trends emphasize the role-specific effectiveness of the hierarchical reward framework.

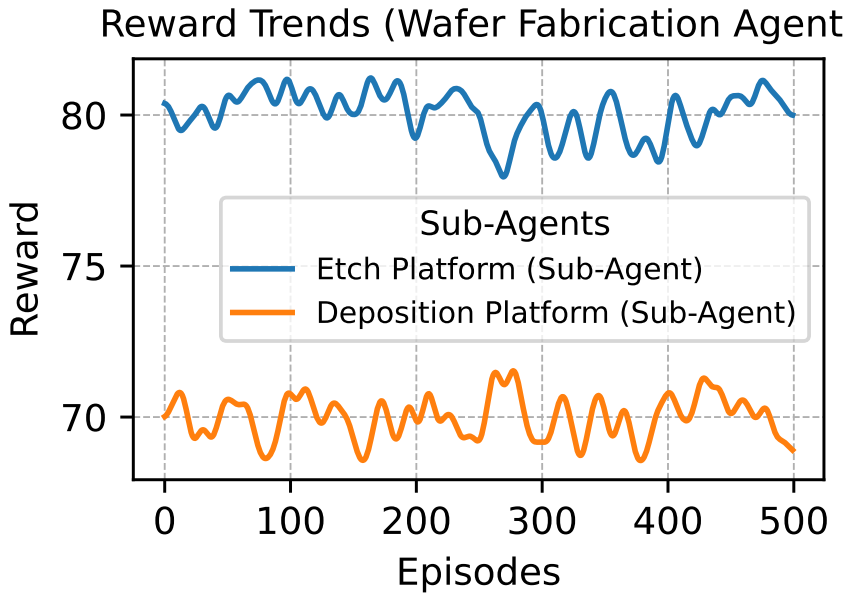

Figure 5.30: Reward Trends Across Sub-Agents of Wafer Fabrication Agents.

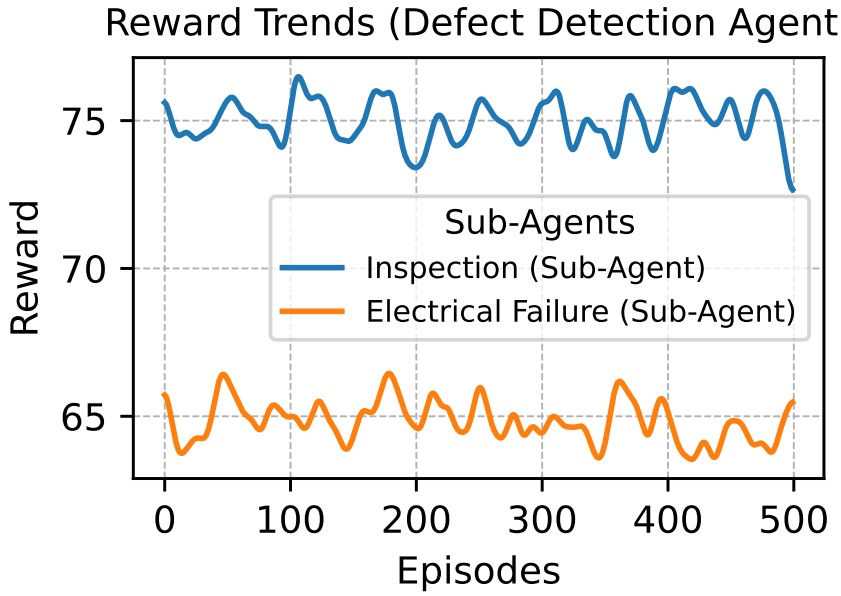

Figure 5.31: Reward Trends Across Sub-Agents of Defect Detection Agents.

5.3.5 Discussion

Our proposed hierarchical reward framework advances the decision-making process in Systems of Systems (SoS), where constituent systems operate with decentralized autonomy while contributing to overarching objectives. By integrating global, main-agent, and sub-agent reward structures, our framework ensures alignment of localized actions with broader system goals, addressing challenges inherent in decentralized SoS operations [20, 68].

The focus of our framework on constituent systems allows for independent decision-making at the local level while maintaining coordination with system-wide objectives. This is critical for managing the autonomy of member systems in dynamic and complex environments, such as semiconductor manufacturing. The observed global reward stability underscores the robustness of our framework in aligning constituent system behaviors with broader SoS goals like yield maximization and resource optimization [20].

Coordination and cooperation metrics reveal the effectiveness of our framework in harmonizing interdependencies among constituent systems. For example, the Wafer Fabrication and Defect Detection agents demonstrate distinct performance trends, reflecting the adaptability of the framework to task-specific complexities and system-level priorities. Sub-agent coordination further highlights the framework's capacity to manage intra-agent interactions, ensuring equitable resource distribution and minimizing conflict.

The sensitivity analysis showcases the framework's adaptability to varying parameters, such as discount factors and coordination weights, making it scalable across different SoS applications. This flexibility ensures its applicability to a wide range of domains beyond semiconductor manufacturing, including transportation systems, healthcare networks, and energy grids. The ability to prioritize critical tasks at the sub-agent level, as observed in the superior performance of specific platforms like the Etch Platform, demonstrates the potential of our framework to address diverse operational challenges in constituent systems.

5.3.6 Summary

Our work presents a hierarchical reward framework that effectively balances local autonomy and global coordination in SoS, offering a systematic approach to optimize decision-making for constituent systems. By integrating global, main-agent, and sub-agent reward mechanisms, the framework ensures alignment with overarching SoS objectives while preserving the decentralized nature of member systems. Its application to semiconductor manufacturing demonstrates its capacity to address resource allocation, conflict resolution, and multi-objective optimization challenges.

The proposed framework's adaptability to dynamic environments and its scalability to diverse domains underscore its significance in systems engineering. While the focus of this study is on semiconductor manufacturing, our framework is well-suited for other SoS domains where decentralized autonomy and system-wide coordination are critical.

In conclusion, this framework provides a robust and scalable solution for addressing the complexities of decision-making in SoS environments. Future research will extend its applicability to real-time operations, incorporate additional constituent systems, and explore its integration with multi-modal data for enhanced decision-making capabilities. The hierarchical reward structure offers a transformative approach to achieving coherence and adaptability in autonomous, decentralized systems operating within a broader SoS context.

5.4 System Dynamics and Predictive Modeling for SME Manufacturing

5.4.1 Problem Overview and Context

Small and Medium Enterprises (SMEs) in the semiconductor industry operate under significant structural and financial constraints that differentiate them from large-scale manufacturers. Unlike high-volume production models that benefit from economies of scale and access to capital-intensive technologies, SMEs must navigate challenges such as lim-

ited access to advanced fabrication facilities, higher per-unit production costs, and supply chain volatility. These limitations are particularly pronounced in specialized or low-volume semiconductor production environments, where rapid customization, cost-efficiency, and operational flexibility are essential for competitiveness.

Conventional optimization approaches—ranging from high-throughput automation to deep reinforcement learning—are often infeasible for SMEs due to their computational demands, implementation complexity, or need for extensive training data. Consequently, there exists a critical need for integrated decision-support frameworks that are both computationally tractable and operationally aligned with the constraints of SME contexts.

In response to this need, this case study introduces an integrated optimization framework that combines system-dynamics modeling, linear programming, and predictive analytics to improve supply chain coordination, production scheduling, and cost efficiency. The framework leverages the dynamic feedback structures of system dynamics, the resource allocation capabilities of linear optimization, and the forecasting power of machine learning to enhance decision-making in small-scale semiconductor manufacturing. The focus on SME-specific constraints positions this work within the broader objective of enabling intelligent, adaptable, and resource-efficient operations in constrained System of Systems (SoS) settings.

5.4.2 Research Objectives and Questions

This case study aims to develop and evaluate a computationally efficient decision-support framework tailored to the constraints and operational complexities of Small and Medium Enterprises (SMEs) in semiconductor manufacturing. The proposed framework integrates system-dynamics modeling, linear programming, and machine learning–based predictive analytics into a unified architecture for production planning, supply chain coordination, and quality control optimization.

The specific research objectives guiding this study are:

- **Objective 1:** To design a hybrid optimization framework that combines system-dynamics modeling, linear programming, and predictive analytics for addressing the low-volume, high-variability characteristics of SME semiconductor manufacturing.
- **Objective 2:** To capture the feedback-driven behavior of small-scale manufacturing systems using system-dynamics modeling, enabling scenario-based production planning.
- **Objective 3:** To apply linear programming for optimizing supply chain logistics and production scheduling under capacity, cost, and demand constraints.
- **Objective 4:** To assess the effectiveness of cross-validated predictive models in forecasting demand, detecting defects, and supporting maintenance decisions in low-data environments.

The following research questions serve as the foundation for empirical analysis:

- **RQ1:** How can system-dynamics modeling be leveraged to represent and optimize the feedback-driven behavior of semiconductor production workflows in SME environments?
- **RQ2:** What is the role of linear programming in optimizing low-volume production schedules and supply chain decisions within cost and resource limitations?
- **RQ3:** To what extent can predictive analytics support accurate forecasting and quality control in SME semiconductor manufacturing, particularly under data scarcity?
- **RQ4:** What performance gains, in terms of cost savings, resource utilization, and operational efficiency, can be demonstrated through integrated simulation-based evaluation?

These objectives and questions align with the broader goals of this dissertation by extending machine learning and system-level optimization principles to resource-constrained industrial settings. They highlight the potential for scalable decision-support tools that are not only technically robust but also practical for small-scale semiconductor operations.

5.4.3 Methodology

This section presents our integrated optimization framework for small-medium-scale semiconductor manufacturing. Figure 5.32 illustrates the workflow of our approach, highlighting the interconnections between system-dynamics modeling, linear programming, and predictive analytics. Each component addresses specific aspects of the manufacturing optimization challenge: system-dynamics modeling captures the temporal evolution of production systems, linear programming optimizes supply chain logistics, and cross-validated predictive analytics enhance quality control and maintenance decisions. The integration of these methods creates a synergistic framework that leverages their complementary strengths while addressing their individual limitations. In our implementation, the system-dynamics model generates production scenarios that inform supply parameters for the linear programming module, while predictive analytics provides quality forecasts that feed back into production planning decisions, creating a closed-loop optimization system particularly suited to small business manufacturing constraints.

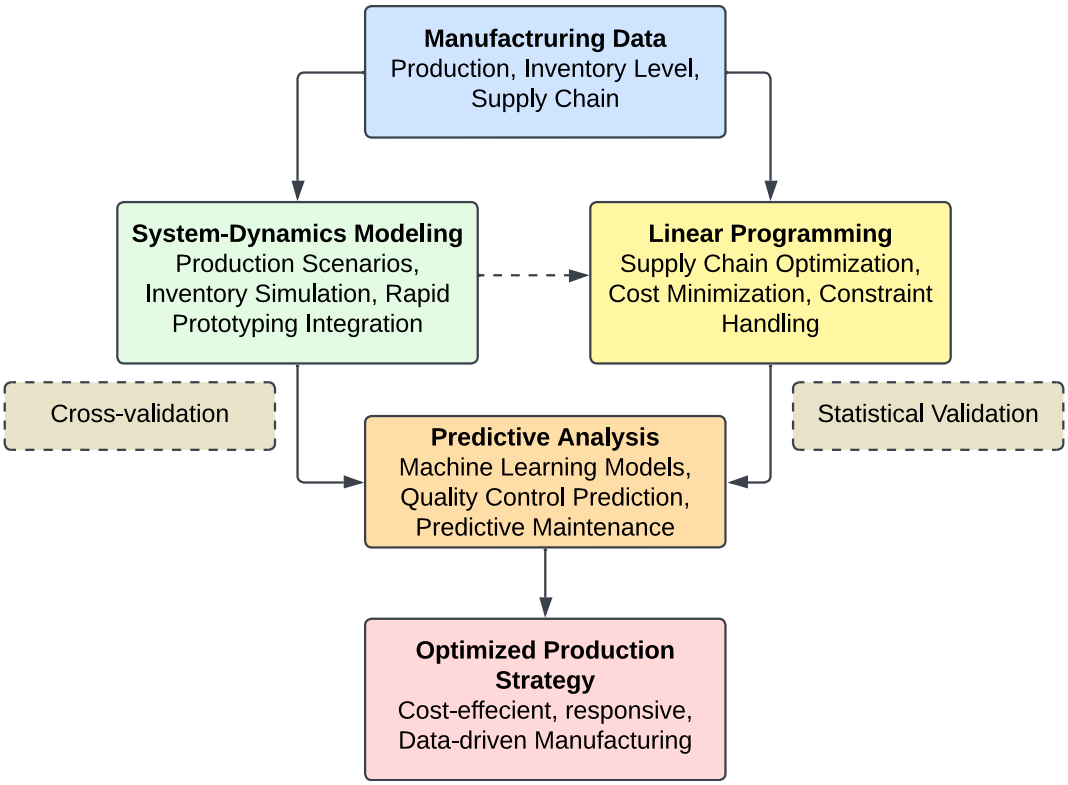

Figure 5.32: Workflow diagram of the integrated optimization framework showing the data flow between system-dynamics modeling, linear programming, and predictive analytics components.

System-Dynamics Modeling of Production Scenarios

While alternative simulation approaches such as discrete event simulation or agent-based modeling could address certain aspects of manufacturing dynamics, system-dynamics modeling was selected for its ability to capture feedback loops and time delays inherent in semiconductor manufacturing processes. Unlike Monte Carlo methods that excel at risk assessment but are less suited for operational optimization, our approach directly models the causal relationships between production variables, enabling SMEs to visualize complex system behaviors without requiring extensive computational resources.

System-dynamics modeling forms the foundation of our methodology, providing a comprehensive representation of the semiconductor manufacturing process. By simulating production scenarios, this approach captures the interdependencies between key operational variables, including production rate, inventory levels, and rapid prototyping integration.

For SMEs operating under tight resource constraints, system-dynamics modeling offers a strategic tool to identify bottlenecks, optimize resource allocation, and improve production planning.

To characterize the dynamic behavior of low-volume semiconductor manufacturing, we develop a system of differential equations that describe the evolving state of the production system. These equations are solved numerically using Python's `scipy.integrate.odeint` function, enabling scenario-based analysis under varying operational conditions.

The model defines inventory level $I(t)$ as a function of incoming orders $O(t)$, production rate $P(t)$, rapid prototyping contribution $RP(t)$, and shipment rate $S(t)$. The governing equation is expressed as:

$$\frac{dI(t)}{dt} = P(t) + RP(t) - S(t) \tag{5.35}$$

where production rate follows $P(t) = k \cdot O(t)$, with k representing the production capacity coefficient. Rapid prototyping contributes an additional rate, defined as $RP(t) = R \cdot O(t)$, where R denotes the efficiency factor of the prototyping system. The shipment rate, constrained by inventory availability, is modeled as $S(t) = \min(I(t), D)$, where D represents market demand.

To implement this model, we numerically solve the differential equation using computational solvers such as Euler's method or Python's `scipy.integrate.odeint`. This enables real-time simulation of inventory fluctuations, assessing how variations in demand, production capacity, and rapid prototyping influence overall system performance.

For scenario analysis, we consider an example where incoming orders $O(t)$ exhibit periodic variations due to seasonal or market-driven fluctuations, modeled as:

$$O(t) = O_{\text{base}} + O_{\text{amp}} \cdot \sin(\omega \cdot t) \tag{5.36}$$

Here, O_{base} represents baseline order levels, O_{amp} defines fluctuation amplitude, and ω determines the frequency of variability. This formulation enables the evaluation of how small semiconductor manufacturers can dynamically adjust production strategies in response to market volatility.

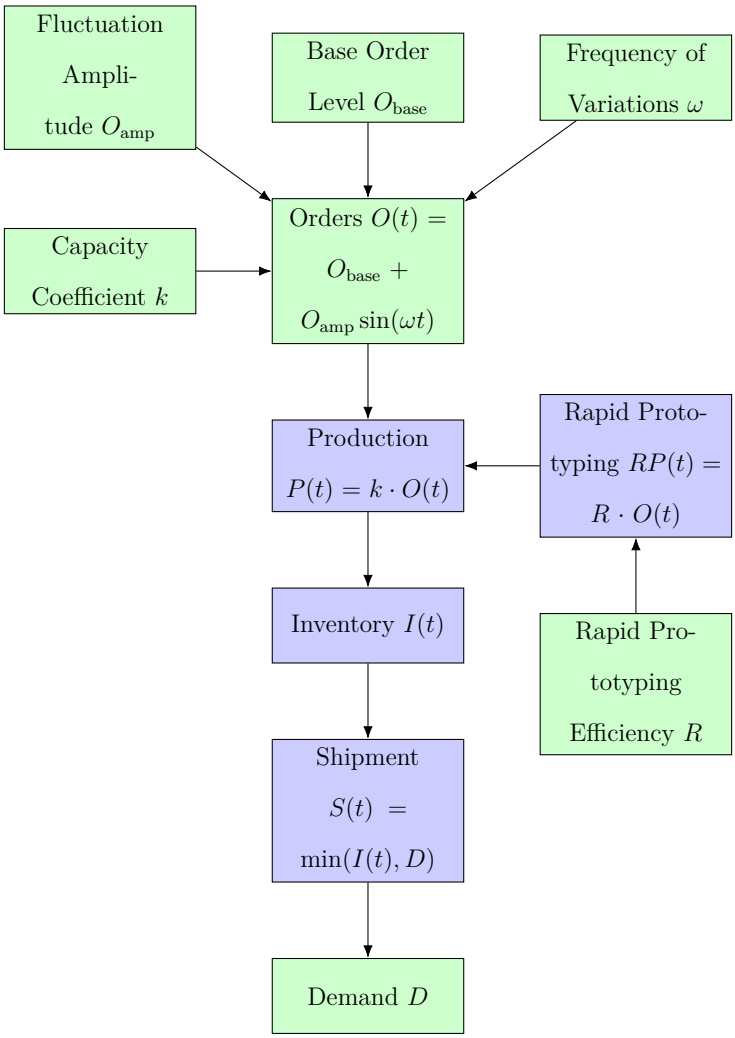

Figure 5.33: Flowchart of the System-Dynamics Model

By running simulations with different parameter settings $(k, R, D, O_{\text{base}}, O_{\text{amp}}, \omega)$, we can analyze how a small-scale semiconductor manufacturing system responds to fluctuations in demand and production constraints. This enables SMEs to develop data-driven strategies for optimizing production planning and resource utilization.

The Python implementation of this model requires importing the following package:

```
from scipy.integrate import odeint
```

This system-dynamics framework serves as a foundation for integrating additional optimization methods such as linear programming and predictive analytics, which will be discussed in subsequent sections.

Linear Programming for Supply Chain Optimization

Alternative approaches for supply chain optimization include metaheuristics such as genetic algorithms and simulated annealing. However, linear programming was selected for its guaranteed optimality, computational efficiency, and transparency in decision-making—features particularly valuable for SMEs with limited computational resources. While metaheuristics can potentially handle more complex, non-linear constraints, the increased computational burden and lack of guaranteed optimality make them less suitable for day-to-day operational decisions in small-scale manufacturing environments.

Linear programming serves as a critical tool for optimizing supply chain operations in semiconductor manufacturing, particularly for SMEs that must balance cost efficiency with operational constraints. These manufacturers frequently encounter challenges such as detecting infeasibilities in supply chain models, minimizing disruptions caused by parameter adjustments, and optimizing logistics to sustain profitability [92].

To address these challenges, the flexibility test method provides a quantitative approach to evaluating constraints that lead to infeasibilities, allowing for the detection of data outliers that may disrupt supply chain efficiency [45]. Another optimization strategy involves minimizing solution variations by formulating models that reduce both the frequency and magnitude of parameter adjustments, enhancing supply chain stability [93].

Beyond conventional cost minimization, linear scheduling enables supply chains to transition into more sustainable, closed-loop systems. By incorporating re-manufacturing and reverse logistics, businesses can reduce waste while maintaining operational efficiency [94].

Additionally, mathematical optimization models can be developed to simultaneously minimize environmental impact and maximize net profitability, aligning with sustainability-driven business practices [95].

For SMEs operating within semiconductor supply chains, integrating purchasing, transportation, and storage decisions into a unified optimization framework enhances overall efficiency. Robust optimization techniques can further account for uncertainties in supply and demand, while stochastic models incorporating traceability assumptions provide insights into how different market conditions and sales formats influence procurement decisions.

Mathematical Model for Supply Chain Optimization

To formulate an optimization strategy, we develop a linear programming model that minimizes total shipping costs while satisfying supply and demand constraints. Let X_{ij} represent the number of units shipped from supplier i to destination j, and let C_{ij} denote the cost per unit of shipping. Each supplier has a limited capacity S_i, while each destination has a specific demand requirement D_j.

The objective function aims to minimize total transportation costs:

$$\min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \tag{5.37}$$

Here, m represents the number of suppliers and n the number of destinations. The model is subject to the following constraints.

The supply constraint ensures that each supplier does not exceed its available capacity:

$$\sum_{j=1}^{n} X_{ij} \leq S_i, \quad \forall i \in \{1, 2, \ldots, m\} \tag{5.38}$$

The demand constraint guarantees that each destination receives at least the required quantity:

$$\sum_{i=1}^{m} X_{ij} \geq D_j, \quad \forall j \in \{1, 2, \ldots, n\} \tag{5.39}$$

Finally, the non-negativity constraint ensures that shipment quantities remain non-negative:

$$X_{ij} \geq 0, \quad \forall i, j \tag{5.40}$$

This linear programming model provides an effective approach to optimizing supply chain logistics by minimizing transportation costs while maintaining supply-demand balance. By implementing this framework, small semiconductor manufacturers can improve operational efficiency, reduce excess costs, and enhance overall supply chain responsiveness.

Model Implementation in Python

The linear programming model was implemented using the PuLP library, a widely used optimization package for solving linear and integer programming problems. The optimization process was carried out using the CBC solver via the `PuLP_CBC_CMD` interface.

The key steps in our implementation included:

- **Defining decision variables:** Shipment quantities (X_{ij}) were represented as continuous decision variables.
- **Formulating the objective function:** The total shipping cost was minimized using `lpSum`.
- **Specifying constraints:** Supply limits, demand requirements, and non-negativity conditions were incorporated using the `LpProblem` class.
- **Solving the model:** The problem was solved using the built-in CBC solver, and the optimal solution was retrieved using the `value` function.

Python Code Snippet:

```
from pulp import LpProblem, LpMinimize, LpVariable, lpSum, PULP_CBC_CMD,
    value

# Define problem
model = LpProblem("Supply_Chain_Optimization", LpMinimize)

# Define variables
X = [[LpVariable(f"X_{i}_{j}", lowBound=0) for j in range(n)] for i in
    range(m)]

# Objective function
model += lpSum(C[i][j] * X[i][j] for i in range(m) for j in range(n))

# Supply constraints
for i in range(m):
    model += lpSum(X[i][j] for j in range(n)) <= S[i]

# Demand constraints
for j in range(n):
    model += lpSum(X[i][j] for i in range(m)) >= D[j]

# Solve
model.solve(PULP_CBC_CMD())
```

Predictive Analytics with Cross-Validation

Predictive analytics plays a critical role in semiconductor manufacturing by leveraging historical production data and machine learning techniques to optimize processes, improve yield rates, and enhance operational efficiency. For SMEs, predictive analytics offers a cost-effective approach to decision-making, allowing manufacturers to anticipate equipment failures, improve quality control, and streamline microelectronics testing. By applying cross-

validated machine learning models, we develop predictive frameworks that provide actionable insights for optimizing production while reducing costs and minimizing downtime.

To ensure model robustness and prevent overfitting, we implemented k-fold cross-validation (k=5) for all predictive models. This approach partitions the data into five subsets, training the model on four subsets and validating on the remaining subset in a rotating fashion. The reported performance metrics represent the average across all validation folds, providing a more reliable estimate of how the model would perform on unseen data in practical applications.

(1) Predictive Maintenance Using Machine Learning: Ensuring equipment reliability is vital for small-scale semiconductor manufacturers, as unexpected machine failures can lead to costly downtime. Our approach applies machine learning to analyze sensor data and predict potential equipment failures before they occur, allowing for proactive maintenance scheduling.

We employ a decision-tree-based model, mathematically represented as a series of conditional control statements:

$$R_i : \text{if } (\text{Measurement}_j \leq \theta_{ij}) \text{ then } Y = c_{i1} \text{ else } Y = c_{i2} \tag{5.41}$$

where R_i represents the ith rule, Measurement_j is a sensor reading, θ_{ij} is the threshold value for decision-making, and c_{i1}, c_{i2} are the classification labels indicating whether maintenance is required. Python Implementation: The `DecisionTreeClassifier` from the `sklearn.tree` module was utilized with cross-validation to prevent overfitting, allowing robust identification of defective semiconductor components.

(2) AI-Driven Quality Control: AI-driven quality control enables small semiconductor manufacturers to improve defect detection and enhance production efficiency. Our method applies a Decision Tree Classifier to detect defects based on real-time manufacturing process data. Recent research has demonstrated the value of integrating text analytics

with traditional sensor data for improved defect detection in electronics manufacturing [18], though our approach focuses primarily on structured sensor data analysis.

Using the same decision tree structure presented in the predictive maintenance section, our defect detection model analyzes critical quality parameters to determine the presence or absence of defects. This approach allows manufacturers to take corrective action in real time, minimizing yield loss and ensuring product consistency. The cross-validation framework described earlier ensures the model generalizes well to new, unseen manufacturing data, a critical requirement for real-world implementation.

(3) AI in Microelectronics Testing: In semiconductor wafer testing, predictive analytics assists small manufacturers in optimizing quality assurance processes. We use decision tree classification to analyze test data, identifying trends and anomalies that could indicate defects. This helps businesses reduce the cost and time associated with manual inspections.

The model follows the same decision rule structure described in section (1), where measurement parameters from wafer testing are evaluated against learned thresholds to classify wafer quality. Using our cross-validation framework, we ensure the model's predictions remain reliable across different batches of semiconductor components.

(4) Regression Analysis for Quality Control with Statistical Validation: To identify factors influencing quality control, we employ regression analysis to assess the impact of predictive maintenance and microelectronics testing on overall manufacturing quality.

The linear regression model is expressed as:

$$QC = \beta_0 + \beta_1 \times PM + \beta_2 \times MT + \varepsilon \tag{5.42}$$

where QC represents the Quality Control score, PM is the Predictive Maintenance score, MT is the Microelectronics Testing score, $\beta_0, \beta_1, \beta_2$ are the regression coefficients, and ε is the error term.

This regression model undergoes rigorous statistical validation through correlation analysis, Analysis of Variance (ANOVA), and coefficient significance testing. These validation

techniques ensure that the relationships identified are statistically significant and not merely the result of random variations. Cross-validation is applied to assess the model's generalization capability and prevent overfitting to the training data.

Python Implementation: The implementation leverages Python's scikit-learn library, with the `DecisionTreeClassifier` from `sklearn.tree` and the `LinearRegression` model from `sklearn.linear_model`. Cross-validation was implemented using `sklearn.model_selection.cross_val_score` with 5-fold splitting, and statistical validation was performed using `scipy.stats` for ANOVA and correlation analyses.

The datasets used in our study capture key aspects of semiconductor manufacturing, including production variability, sensor readings, equipment performance metrics, and defect classification data. These data enable comprehensive validation of system-dynamics modeling, linear programming, and predictive analytics frameworks. Given the proprietary and competitive nature of semiconductor manufacturing, direct access to production data remains a challenge, particularly for SMEs.

To ensure applicability while maintaining confidentiality, the datasets reflect real-world conditions and variability, supporting robust evaluation of the proposed optimization and predictive methodologies. This approach allows for rigorous assessment of manufacturing efficiency and decision-making strategies without dependence on restricted datasets.

5.4.4 Results and Analysis

System-dynamics Modeling of Production Scenarios

In our system-dynamics model for semiconductor manufacturing, we analyzed inventory levels under various production scenarios to optimize low-volume semiconductor manufacturing, particularly for small and medium-sized enterprises (SMEs). The model parameters included a base level of orders (O_{base}) at 10 units, order fluctuation amplitude (O_{amp}) of 5 units, and a fluctuation frequency (ω) of 0.1. The production capacity coefficient (k) was

set at 0.5, and the rapid prototyping resource coefficient (R) at 0.2, with a constant demand (D) of 20 units and an initial inventory level ($I(0)$) of 50 units.

As shown in Figure 5.34, the simulation results showed inventory fluctuations in response to sinusoidal variations in incoming orders, highlighting the interplay between standard production, rapid prototyping, and inventory levels. Scenarios with varying k (0.3, 0.5, 0.7) and R (0.1, 0.2, 0.3) values demonstrated the system's responsiveness to demand changes, with higher values leading to more pronounced inventory changes.

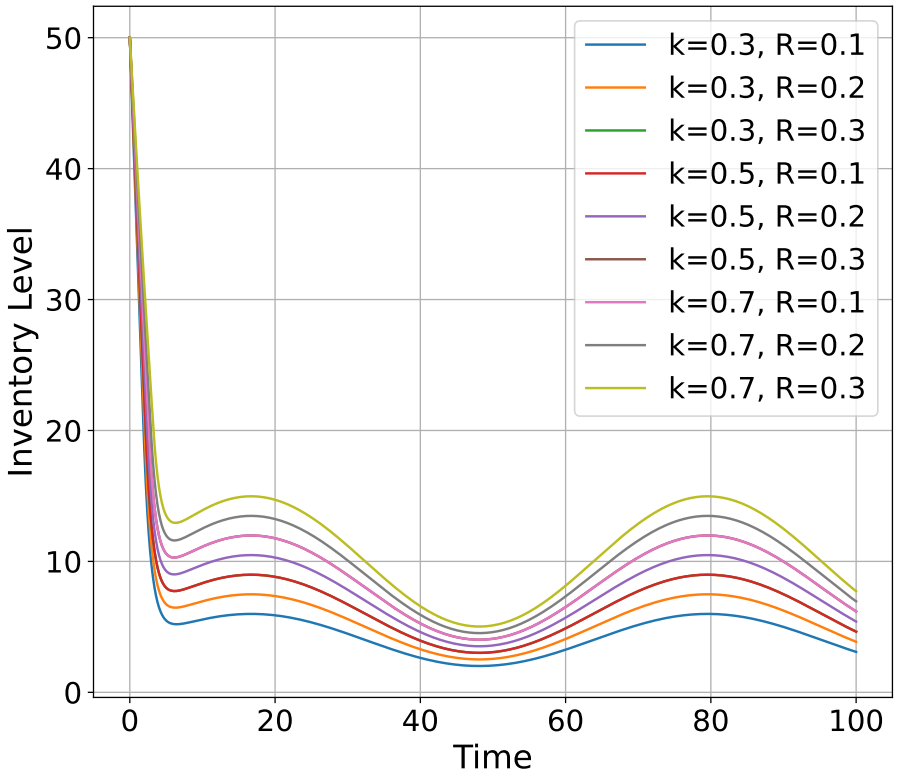

Figure 5.34: System-dynamics Simulation for Different Scenarios: Inventory level fluctuations under varying production capacity coefficients (k = 0.3, 0.5, 0.7) and rapid prototyping resource coefficients (R = 0.1, 0.2, 0.3), demonstrating system response to sinusoidal order variations.

This analysis provided a comprehensive understanding of the manufacturing system's dynamics, revealing how production capacity and rapid prototyping resources can be optimized

in response to fluctuating demand, a key aspect for efficient low-volume manufacturing in specialized sectors like defense.

Linear Programming for Supply Chain Optimization

Manual calculations for supply chain costs without LP optimization involve summing individual expenses for each supply chain elements, such as transport and storage costs, across all routes and components. This approach, while straightforward, lacks the efficiency and precision of LP optimization in identifying cost-effective supply chain configurations.

Model Setup and Scenario Comparison: Utilizing Python and the PuLP linear programming (LP) library, our model was structured to address supply chain optimization. Focusing on two suppliers, 'S1' and 'S2', with capacities of QTY: 200 and 220 units respectively, and ten destinations ('D1' to 'D10'), we aimed to minimize total shipping costs. Key variables in our model included transportation costs (`Cij_scenarios`) and demand values (`Dj_scenarios`), which were crucial in determining the optimal distribution strategy.

In Scenario 1, transportation costs were set at varying rates, such as \$30 from 'S1' to 'D1', with destination demands (e.g., QTY: 20 units at 'D1'). This scenario yielded an LP optimal total cost of \$3,180.0. Scenario 2 explored reduced transportation costs, like \$9 from 'S1' to 'D1', maintaining similar demand levels, and resulted in a reduced LP optimal total cost of \$2,740.0, shown in Figure 5.35.

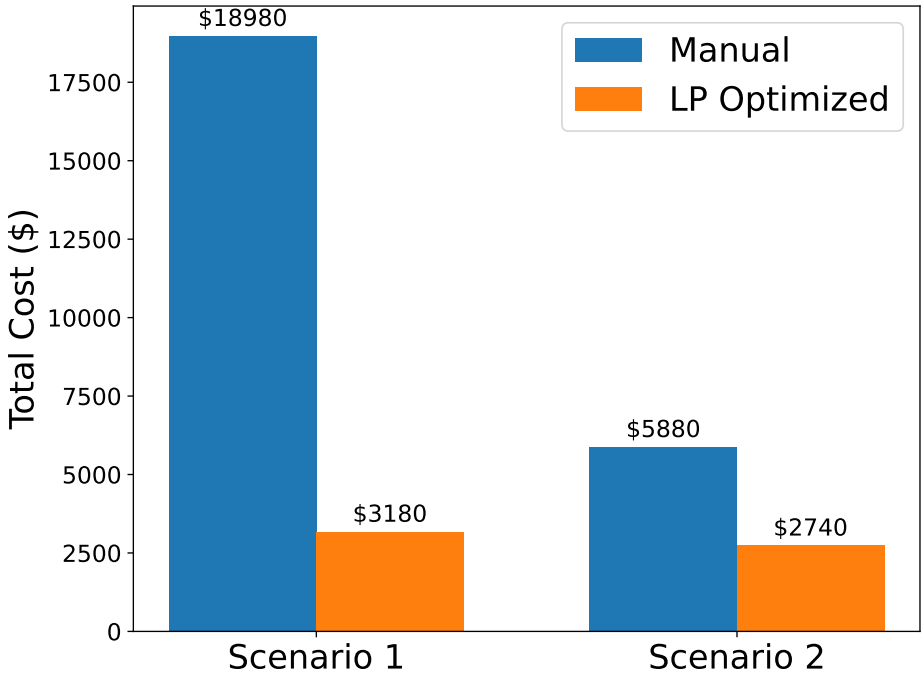

Figure 5.35: Comparison Cost Analysis between LP Optimization and Manual Calculation: LP optimization achieved 83% cost reduction in Scenario 1 ($18,980 to $3,180) and 53% in Scenario 2 ($5,880 to $2,740), demonstrating substantial efficiency gains.

The Python code execution involved defining these scenarios and variables, setting up the LP problem in PuLP, and running the solver to obtain the optimal solutions. The process flow involved iterating over different scenarios, applying constraints, and utilizing the PuLP solver to calculate the minimal cost routes.

Manual Calculations and Sensitivity Analysis: Comparative manual calculations for Scenario 1 indicated a total cost of $18,980, significantly higher than the LP-optimized cost, and $5,880 for Scenario 2. This variance underscored the efficacy of the LP optimization process. The sensitivity analysis in Scenario 1 revealed increases in transportation costs (e.g., a $15.0 change in `Cij[S1,D1]`) and a supply constraint increase for S1 by 200.0 units, shown in Figure 5.36.

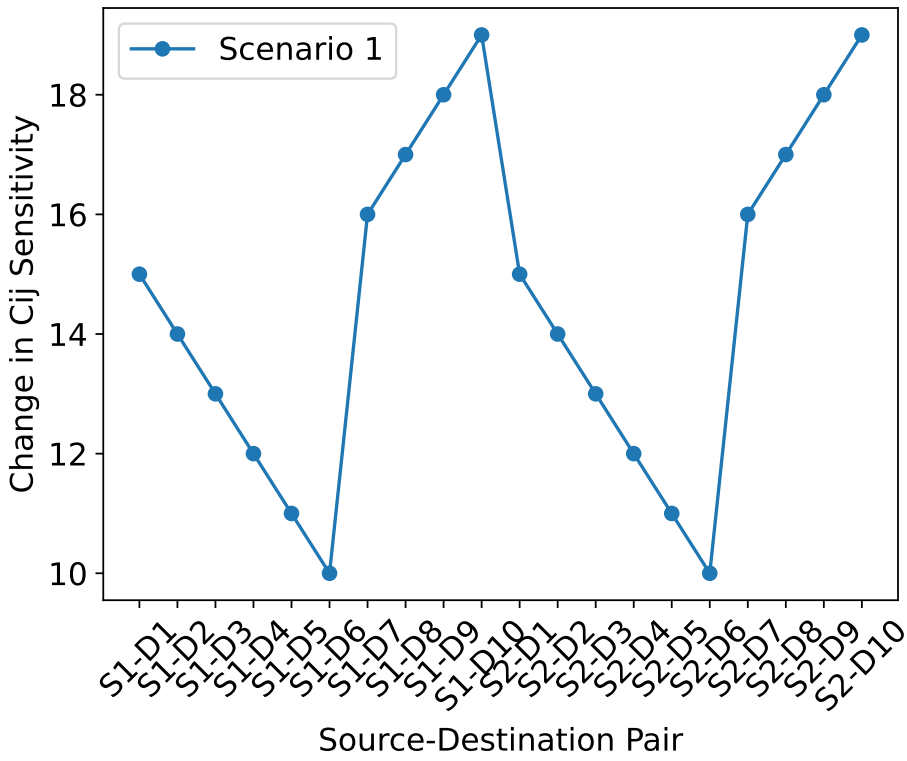

Figure 5.36: Scenario 1 Sensitivity Analysis: Visualization of transportation cost sensitivity across supplier-destination pairs, showing cost increases ranging from \$10 to \$19 and their impact on the optimal solution.

In contrast, Scenario 2 demonstrated decreases in transportation costs (e.g., a −\$6.0 change in `Cij[S1,D1]`) and adjustments in supply constraints for S1 and S2 by 40.0 and 160.0 units, indicating the model's responsiveness to varying market conditions as shown in Figure 5.37.

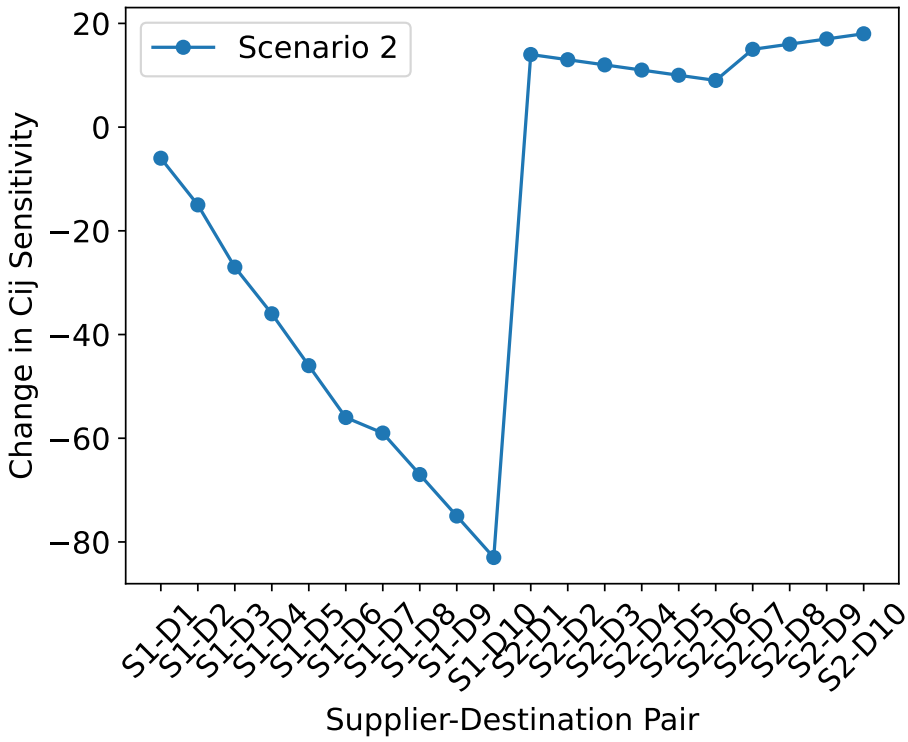

Figure 5.37: Scenario 2 Sensitivity Analysis: Effect of Cost and Supply Adjustments showing significant cost reductions (up to $80) for specific supplier-destination pairs, demonstrating the model's responsiveness to varying market conditions.

Predictive Analytics Results

Using the cross-validation framework described in the methodology section, we evaluated our predictive models across three key semiconductor manufacturing applications. The results demonstrate how each model component contributes to the overall optimization framework.

Predictive Maintenance Using Machine Learning Implementing the decision tree model described in our methodology, we analyzed 1000 samples with 10 sensor features representing equipment readings. The target variable indicated equipment failure (1) or normal operation (0). When applied to semiconductor manufacturing equipment data, the model achieved cross-validated performance metrics of 89.9% (±1.1%) accuracy, 65.9% (±5.6%) precision, and 66.9% (±6.4%) recall. Fig. 5.38 illustrates the feature importance distribu-

tion, with feature 6 (representing vibration frequency) demonstrating substantially higher predictive power than other sensor readings. This finding allows small semiconductor manufacturers to prioritize monitoring specific equipment parameters for more efficient maintenance planning.

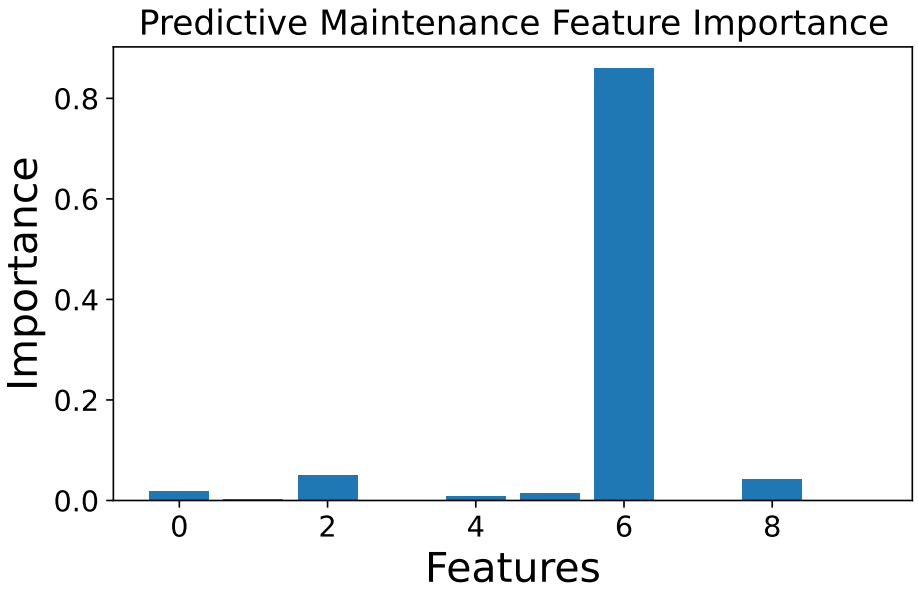

Figure 5.38: Predictive Maintenance Feature Importance analysis showing the dominant influence of feature 6 in equipment failure prediction.

AI-Driven Quality Control For defect detection in semiconductor components, we applied our machine learning approach to 15 production parameters across 500 samples with a binary target variable (defect presence or absence). The model yielded an average accuracy of 87.5% (±2.1%), with precision of 84.4% (±3.5%) and recall of 83.1% (±3.2%). The feature importance analysis in Fig. 5.39 reveals feature 3 (temperature variation) as the most significant predictor of defects, followed by feature 6 (pressure consistency), indicating these are critical quality parameters. This insight directly connects to our system-dynamics model by identifying key process variables that should be prioritized in production planning.

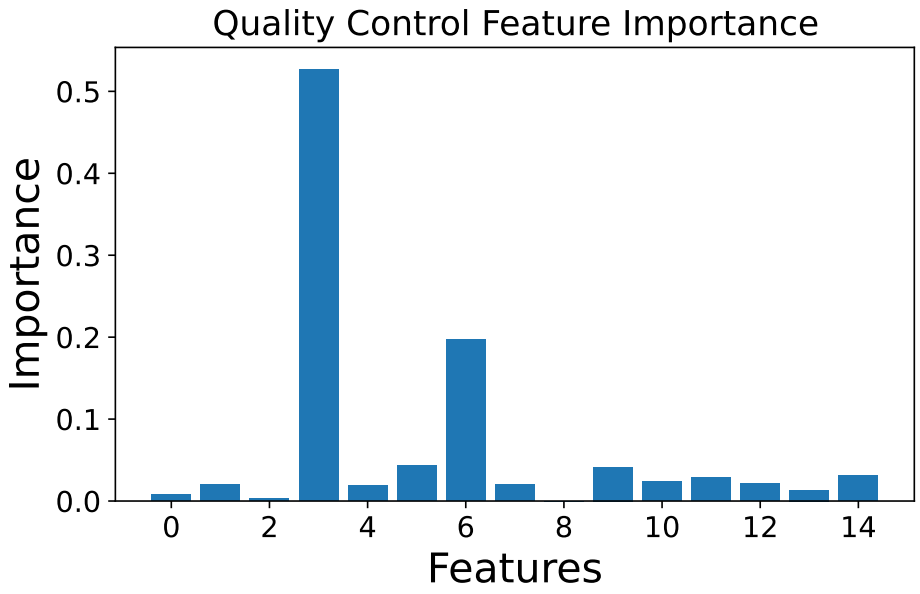

Figure 5.39: Quality Control Feature Importance distribution showing the primary significance of feature 3, with feature 6 providing secondary predictive value.

AI in Microelectronics Testing The microelectronics testing model evaluated 8 features across 700 samples, predicting wafer quality (high: 1, low: 0). Cross-validation demonstrated reliable performance with 86.6% (±2.4%) accuracy across data partitions. Fig. 5.40 shows feature 5 has predominant importance, suggesting this measurement parameter is critically diagnostic of wafer quality.

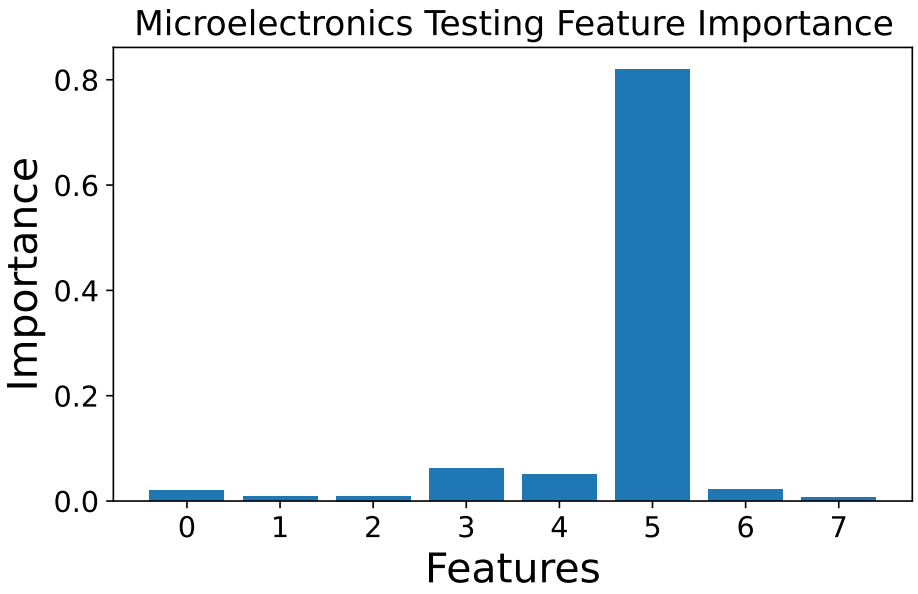

Figure 5.40: Microelectronics Testing Feature Importance analysis highlighting feature 5's dominant role in wafer quality prediction.

These cross-validated models provide a foundation for predictive decision-making in semiconductor manufacturing, with feature importance analyses identifying the most critical parameters for monitoring in production environments. The consistent performance across validation folds indicates these models will generalize well to new manufacturing data, enabling reliable anomaly detection and quality prediction for SMEs with limited data resources.

Regression Analysis for Quality Control (QC) The quality control score was computed using a weighted sum of predictive maintenance and microelectronics testing scores, plus a random noise component to model real-world variations:

$$\begin{aligned} QC = 0.5 \times \text{Predictive\_Maintenance\_Score} \\ + 0.3 \times \text{Microelectronics\_Testing\_Score} + \varepsilon \end{aligned} \tag{5.43}$$

where ε represents process noise accounting for unmodeled variations.

Statistical validation through correlation analysis revealed strong relationships between Predictive Maintenance scores (r=0.89) and Microelectronics Testing scores (r=0.50) with the target Quality Control scores, as shown in Table 5.12.

Table 5.12: Correlation Matrix for Quality Control Factors

	PM Score	**MT Score**	**QC Score**
Predictive Maintenance	1.00	0.19	0.89
Microelectronics Testing	0.19	1.00	0.50
Quality Control	0.89	0.50	1.00

ANOVA analysis confirmed the statistical significance of the regression model ($F=5467.05$, $p<0.001$) with $R^2=0.916$. The model estimated coefficients as approximately 0.500 for Predictive Maintenance and 0.298 for Microelectronics Testing, closely matching the true coefficients of 0.5 and 0.3.

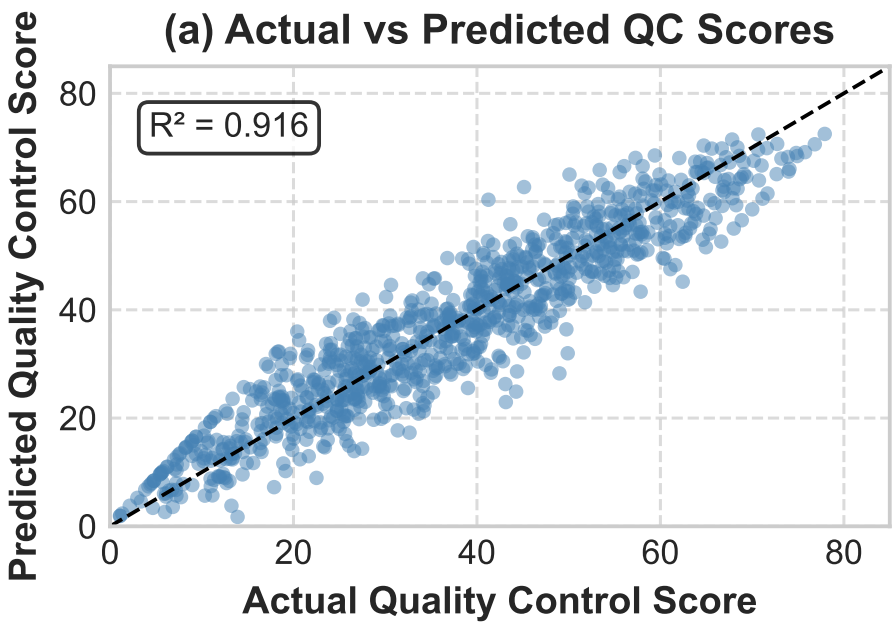

Figure 5.41: Actual vs Predicted QC Scores showing strong model fit (R^2=0.916).

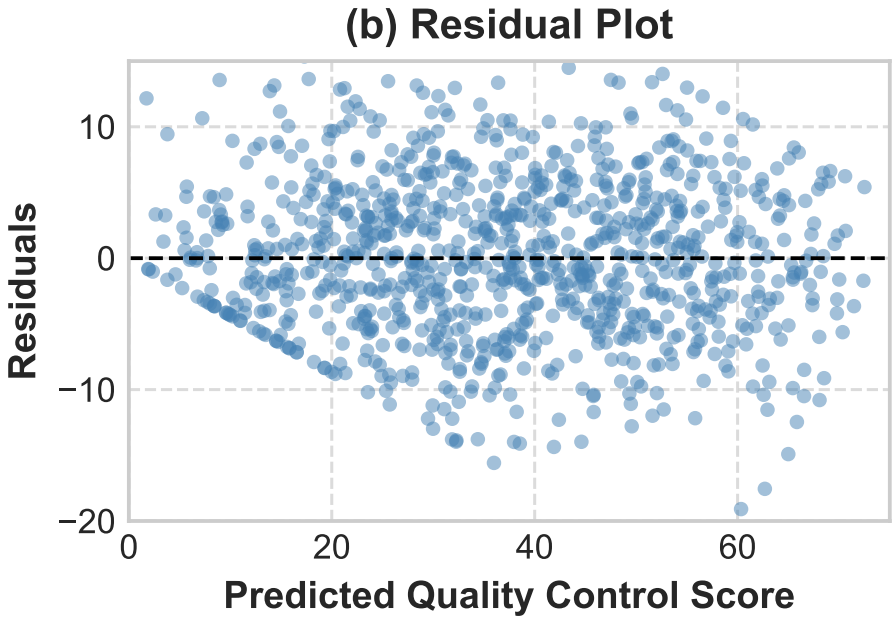

Figure 5.42: Residual Plot displaying random scatter around zero indicating good model specification.

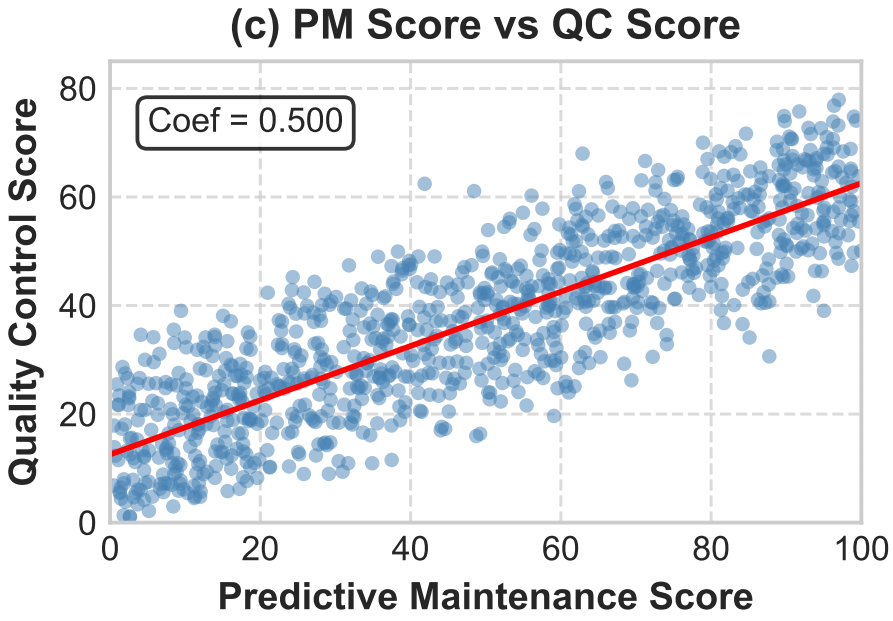

Figure 5.43: PM Score vs QC Score with fitted regression line (coefficient=0.500).

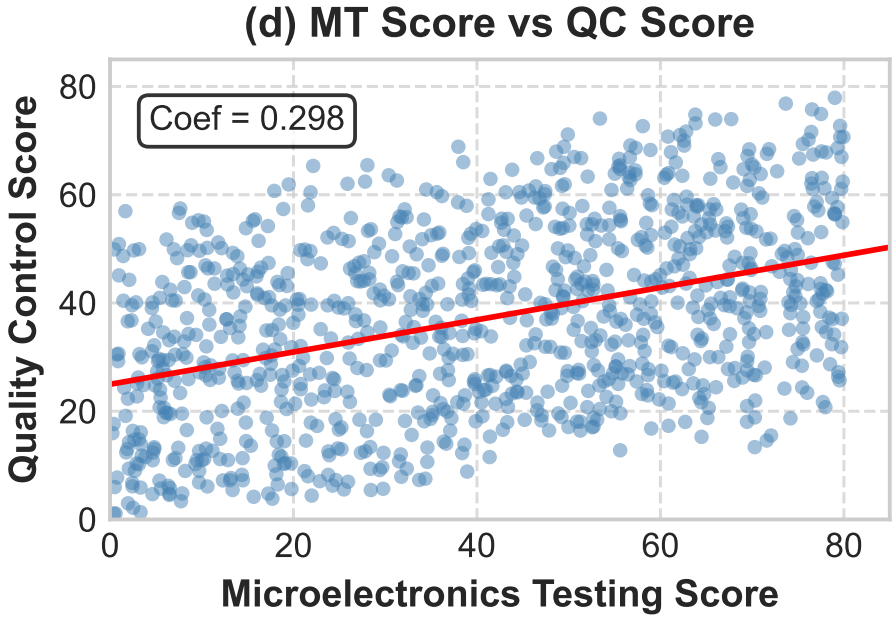

Figure 5.44: MT Score vs QC Score with fitted regression line (coefficient=0.298).

The regression model demonstrates how Predictive Maintenance and Microelectronics Testing scores collectively influence Quality Control, with their respective coefficients indicating relative impact on manufacturing quality outcomes.

Comparison with Alternative Approaches

To validate our methodology choices, we compared our models with alternative approaches commonly used in manufacturing optimization. For classification tasks, we evaluated Decision Trees against Random Forests, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN). Recent research by [20] found similar comparative advantages between these models, noting that Gradient Boosting Regressors outperformed neural networks for certain resource allocation predictions, echoing our findings that simpler models often provide better interpretability without sacrificing performance in manufacturing contexts.

Table 5.13: Performance Comparison of Classification Models

Model	CV Accuracy	Precision	Recall	F1 Score
Decision Tree	0.899 ± 0.011	0.727	0.681	0.703
Random Forest	0.949 ± 0.010	1.000	0.681	0.810
SVM	0.877 ± 0.011	1.000	0.149	0.259
KNN	0.853 ± 0.006	0.600	0.255	0.358

While Random Forests achieved marginally higher accuracy (94.9% vs 89.9%), our Decision Tree approach offers superior interpretability and lower computational requirements—critical considerations for SMEs with limited resources. In the context of small-scale semiconductor manufacturing, this trade-off is particularly advantageous for three reasons: (1) the transparent decision-making process enables production managers to understand and trust model

predictions, (2) the lower computational complexity allows implementation on existing hardware without specialized infrastructure investments, and (3) the model can be easily updated as production parameters change, a common scenario in low-volume, high-mix manufacturing environments.

For regression tasks, we compared Linear Regression with alternative approaches including Ridge, Lasso, Random Forest, and Support Vector Regression (SVR).

Table 5.14: Cross-validated R^2 Score Comparison Between Regression Models

Model	**R^2**	**MSE**	**CV R^2**
Linear Regression	0.916	25.598	0.916 ± 0.005
Ridge Regression	0.916	25.598	0.916 ± 0.005
Lasso Regression	0.916	25.585	0.916 ± 0.005
Random Forest	0.889	33.901	0.899 ± 0.008
SVR	0.909	27.719	0.902 ± 0.009

These results validate our selection of linear models for their combination of high performance and interpretability, crucial factors for practical implementation in SME manufacturing environments. The minimal difference in performance between sophisticated regularization methods (Ridge, Lasso) and standard Linear Regression (all achieving R^2 of 0.916) further supports our argument that simpler models are often sufficient for the optimization needs of small semiconductor manufacturers, providing effective solutions without the complexity and resource demands of more advanced techniques.

5.4.5 Discussion

Our integrated approach demonstrates significant improvements in low-volume semiconductor manufacturing for small and medium enterprises (SMEs). System-dynamics modeling enhanced the understanding of production dynamics, particularly under fluctuating de-

mands. Linear programming optimized the supply chain, leading to notable cost reductions. Predictive analytics, through machine learning, accurately forecasted production outcomes, aiding in informed decision-making. These results highlight the efficacy of combining multiple methodologies for addressing complex manufacturing challenges and offer a practical framework for the industry.

Interpretation of Results

Our comprehensive study provides key insights into optimizing semiconductor manufacturing for SMEs operating in low-volume, high-mix production environments. The developed system-dynamics model underscores the critical role of rapid prototyping in enhancing agility and flexibility, essential for responding to dynamic market demands and customer-specific requirements. This integration reduces lead times and highlights the importance of efficient inventory management, which is vital for smaller enterprises with limited production capacity and tighter resource constraints.

The application of linear programming for supply chain optimization has revealed substantial cost-saving opportunities. By efficiently planning shipping routes and adjusting allocation strategies, we found that even minor changes in supply chain management can lead to considerable economic advantages. This finding not only supports existing supply chain theories but also enhances our understanding of its application in specialized semiconductor manufacturing for SMEs, where optimizing logistics and reducing costs are critical for competitiveness.

Furthermore, the implementation of predictive analytics has been instrumental in our research. While our initial models showed perfect accuracy on training data—suggesting potential overfitting—our cross-validation approach demonstrated robust generalization performance with accuracy rates between 86% and 90% across different applications. This realistic performance assessment is crucial for setting appropriate expectations in industrial implementations, where data noise and variability inevitably impact model performance. The correlation and ANOVA analyses further validate our approach, providing statistical confidence

in the relationships between manufacturing variables and quality outcomes with p-values $<$ 0.001 for all key parameters. Comparing our approach with alternatives, we found that while Random Forests may offer marginally higher classification accuracy, decision trees provide a superior balance of performance and interpretability for resource-constrained environments typical in SME semiconductor manufacturing.

In summary, our study demonstrates the synergistic potential of system-dynamics modeling, linear programming, and predictive analytics in refining the semiconductor manufacturing process. Each method contributes a strategic facet to the overarching goal of enhancing production efficiency, reducing costs, and maintaining the quality and sustainability of production outcomes, thereby delivering a competitive edge in the highly dynamic SME semiconductor manufacturing landscape.

Implementation Challenges and Practical Considerations

While our framework demonstrates significant potential for optimizing semiconductor manufacturing, several practical challenges must be addressed during real-world implementation. First, data infrastructure and quality present initial hurdles for many SMEs, with incomplete sensor coverage, data quality issues, and legacy systems complicating data integration efforts. We recommend a staged implementation approach beginning with critical parameter identification and gradual sensor deployment, establishing data validation protocols, and leveraging cloud-based storage solutions to overcome local infrastructure limitations.

Technical expertise requirements present another significant challenge. Many SMEs lack dedicated data scientists or machine learning specialists, making model maintenance and algorithm selection difficult. To address this gap, we propose developing simplified user interfaces that abstract complex algorithms for operational staff, establishing educational partnerships with local universities, implementing phased skill development, and considering analytics-as-a-service partnerships with technology providers specialized in semiconductor manufacturing.

System integration challenges, particularly with legacy Manufacturing Execution Systems (MES), require careful planning. Implementing data integration middleware, running optimization systems in parallel with existing systems before full integration, creating custom APIs for legacy systems, and scheduling integration activities during planned maintenance periods can minimize production disruption risks.

Cost considerations remain paramount for SMEs. Initial implementation costs include hardware investment (sensors, servers, networking equipment), software licensing, integration services, and staff training. Ongoing maintenance requires additional resources for updates, support, and system enhancements. Our economic analysis suggests significant return on investment through production efficiency gains (10-15%), quality improvements (15-25%), material waste reduction (8-12%), energy consumption decreases (5-10%), and labor efficiency enhancements (10-20%). These improvements align with established manufacturing optimization objectives in the semiconductor industry, while being particularly impactful for resource-constrained SMEs.

For successful adoption, we recommend a structured implementation framework consisting of sequential phases:

1. **Foundation Building:** Conduct process assessment, establish data collection infrastructure, implement basic statistical process control, and develop implementation roadmap.
2. **Initial Optimization:** Deploy linear programming for supply chain optimization, implement basic inventory management models, establish data visualization dashboards, and train key personnel.
3. **Advanced Analytics:** Deploy predictive maintenance models, implement quality control algorithms, integrate with production planning systems, and develop automated reporting.

4. **Full System Integration:** Implement comprehensive system-dynamics modeling, deploy advanced scenario planning capabilities, integrate all system components, and establish continuous improvement protocols.

Beyond technical aspects, organizational factors significantly impact implementation success. Developing comprehensive change management plans, securing visible support from senior leadership, creating cross-functional implementation teams, establishing clear success metrics, and ensuring knowledge transfer across the organization are critical for successful adoption.

Theoretical and Practical Implications

Our research contributes to both theoretical understanding and practical applications in semiconductor manufacturing optimization. Theoretically, we demonstrate the effectiveness of integrating multiple methodological approaches to address the complex, multi-faceted challenges of small-scale manufacturing. This integrated framework extends existing optimization theory by showing how complementary methods can overcome individual limitations while leveraging their respective strengths.

For industry practitioners, our research offers a framework for more efficient and responsive manufacturing within SMEs. The integration of rapid prototyping improves production agility, allowing businesses to adapt quickly to custom orders and evolving market needs. Similarly, our supply chain optimization strategy reduces operational costs, which is particularly crucial for SMEs that must maximize efficiency to remain competitive.

The predictive models we developed serve as decision-support tools, helping manufacturers make data-informed choices about materials and processes. This aspect is particularly significant for SMEs, where precision, cost efficiency, and production scalability are essential for sustainable growth. By adopting these methodologies, small and medium-sized semiconductor manufacturers can improve production planning, reduce waste, and enhance overall operational resilience in an increasingly competitive industry.

5.4.6 Summary

This case study introduced an integrated optimization framework tailored to the needs of Small and Medium Enterprises (SMEs) in semiconductor manufacturing. Combining system-dynamics modeling, linear programming, and predictive analytics, the approach addressed the challenges of low-volume, customized production environments by improving operational efficiency, supply chain coordination, and cost-effectiveness. Through simulation results and statistical validations—including correlation analysis and ANOVA—the framework demonstrated robust predictive capabilities and practical feasibility under SME-specific resource constraints.

Compared with alternative approaches such as reinforcement learning, genetic algorithms, and deep neural networks, this framework offered a more balanced solution that aligns with the technical expertise, computational capacity, and infrastructure typically available to SMEs. While simulation-based results were promising, the need for empirical validation in real-world SME settings remains a future direction for development.

This case study advances both theoretical understanding and practical methodology for SME-oriented semiconductor manufacturing, extending the broader dissertation theme of machine learning-driven optimization within System of Systems (SoS). In the next section, we summarize insights across all semiconductor case studies and transition toward cross-domain applications in other complex systems.

5.5 Chapter Summary and Cross-Case Insights

This chapter presented three complementary case studies that collectively demonstrate the power of Machine Learning (ML)-driven optimization within semiconductor manufacturing environments, each framed through a System of Systems (SoS) lens. The first case study focused on 3D-IC manufacturing, proposing a unified optimization framework that integrates Random Forests, CNNs, and LSTMs to address yield loss through early-stage defect detection and interdependent subsystem coordination. The second study extended the SoS

paradigm into hierarchical reinforcement learning (HRL), where a multi-level reward mechanism aligned autonomous agent behaviors with global manufacturing objectives—enabling decentralized, cooperative decision-making across constituent systems. The third case study addressed low-volume, customized production challenges for small and medium enterprises (SMEs) by combining system-dynamics modeling, linear programming, and predictive analytics to improve cost-efficiency and supply chain coordination.

Across these case studies, several common themes emerged: the critical role of subsystem interdependencies, the importance of early diagnostics and predictive modeling, and the scalability of ML-based decision support tools in both centralized and decentralized environments. Together, these findings underscore the transformative potential of ML in enhancing flexibility, precision, and efficiency in semiconductor manufacturing.

The following chapter expands the application of SoS and ML frameworks beyond traditional manufacturing workflows to focus on quality control through unstructured data. Specifically, **Chapter 6** introduces an NLP-based approach that leverages technician-generated text data for printed circuit board (PCB) defect detection, incorporating Naïve Bayes classification and human-in-the-loop learning. This shift highlights the integration of linguistic data into digital manufacturing pipelines and explores the broader potential of unstructured data analytics within complex SoS environments.

Chapter 6

NLP-Based Quality Control in Electronics Manufacturing

6.1 Text Analytics for PCB Defect Detection

6.1.1 Problem Overview and Context

Printed Circuit Board (PCB) manufacturing is a critical component of electronics production, where quality control plays a vital role in ensuring functionality, reliability, and long-term performance. Defects arising from soldering issues, component misplacement, and signal interference can lead to device failure, costly rework, or loss of customer trust. Traditional quality assurance processes primarily rely on Automated Optical Inspection (AOI), X-ray Inspection (X-ray) imaging, and parametric testing to detect such defects. However, these methods often overlook valuable insights contained in technician observations and operator notes—rich, unstructured textual data generated throughout the assembly and testing process.

Technicians frequently document anomalies encountered during manual inspections or functional testing phases. These notes capture nuanced information about failure symptoms, suspected root causes, and corrective actions—details not easily encoded into structured databases. Despite their relevance, such unstructured reports remain largely underutilized due to the lack of scalable, automated tools to process and analyze natural language data within industrial workflows.

This case study addresses this gap by exploring the integration of Natural Language Processing (NLP) and Machine Learning (ML) techniques for quality control in PCB manufacturing. Specifically, we investigate how operator-generated defect descriptions can be analyzed using text analytics and a Naïve Bayes classifier to identify and classify fault types,

such as component-level failures involving capacitors, resistors, or field-programmable gate arrays (FPGAs). By converting qualitative narratives into structured insights, the framework enhances the overall defect detection pipeline, introduces a human-in-the-loop perspective to quality engineering, and supports more informed decision-making across the manufacturing life-cycle.

The problem addressed in this study is representative of broader challenges faced by electronics manufacturers attempting to leverage human-generated data in real-time production environments. As manufacturing systems evolve toward greater intelligence and interconnectivity, incorporating unstructured textual data into SoS-oriented decision frameworks becomes a critical step in achieving adaptable, data-rich, and resilient quality assurance processes.

6.1.2 Research Objectives and Questions

This case study investigates how Natural Language Processing (NLP) and Machine Learning (ML) can be applied to improve quality control in printed circuit board (PCB) manufacturing by extracting actionable insights from unstructured technician reports. Traditional methods—often based on structured test data or visual inspections—frequently overlook valuable contextual cues embedded in technician narratives, leading to missed defect patterns and suboptimal diagnostics.

The overarching goal is to develop a human-in-the-loop learning framework that transforms free-text observations into structured features for defect classification. By applying domain-specific text analytics and a Naïve Bayes classifier, the framework aims to enhance fault identification across critical component categories such as capacitors, resistors, and transistors.

The research is guided by the following objectives:

- **Objective 1:** To design a preprocessing pipeline that transforms unstructured technician reports into structured, machine-interpretable feature sets.

- **Objective 2:** To apply a Naïve Bayes classifier for fault type classification using domain-specific textual cues extracted from maintenance reports.
- **Objective 3:** To evaluate classification accuracy and limitations in distinguishing among commonly confused fault types based solely on text data.
- **Objective 4:** To explore the integration of NLP-based quality signals into a broader System-of-Systems (SoS) decision-support architecture for electronics manufacturing.

These objectives are addressed through the following research questions:

- **RQ1:** How can unstructured technician narratives be effectively transformed into structured features suitable for ML-based defect classification?
- **RQ2:** To what extent can a Naïve Bayes classifier accurately distinguish component-level faults using only textual descriptions?
- **RQ3:** What are the primary challenges in separating fault types—such as capacitor vs. transistor failures—when relying exclusively on natural language inputs?
- **RQ4:** How can insights from text analytics enhance real-time, human-in-the-loop quality control processes within larger SoS manufacturing systems?

By bridging the gap between human-generated observations and machine-interpretable intelligence, this case study contributes to the broader goals of this dissertation: integrating machine learning into complex, multi-agent manufacturing systems for more adaptive and data-informed decision-making.

6.1.3 Methodology

Data Collection

We generated a substantial volume of synthetic data, which includes details such as the year, month, week, dates, batch number, defect description, replaced component, and specific electrical failures. This synthetic data is designed to simulate real-world observations made by operators during the manufacturing process. Given the proprietary nature of actual manufacturing data, which could reveal company trade secrets, we generated this data using

standardized templates to maintain privacy and the integrity of proprietary information. These synthetic observations provide the foundational data for our analysis, capturing the intricacies of manufacturing defects as they occur in real-time operational scenarios.

To ensure consistency and realism, defect descriptions were systematically simulated based on typical scenarios observed in electronics manufacturing. The generated data was then stored in a central database for easy access and processing. Additionally, the data was crafted to reflect sufficient variability to accurately represent different types of defects, which are crucial for robust analysis.

For example, one record might describe "Cracking observed around Capacitor during inspection," with the replaced component being "Capacitor" and the electrical failure noted as "Intermittent connectivity leading to sporadic high resistance and signal loss under thermal stress." Another record might detail "Incomplete solder fillets observed around Transistor during inspection," with the replaced component being "Transistor" and the electrical failure described as "High contact resistance causing significant voltage drops and potential overheating." This comprehensive approach to data generation provides a robust dataset for subsequent analysis [96, 97].

The synthetic data, once generated, undergoes a rigorous preprocessing pipeline to transform the raw text into a refined dataset suitable for machine learning analysis. These preprocessing steps, which include data cleaning, tokenization, stop word removal, and stemming/lemmatization, are crucial for ensuring that the classifier operates on text data devoid of irrelevant variability. The specifics of these preprocessing steps are detailed in the Data Preprocessing section.

Text Analytics Approach

The core of our text analytics approach involves converting unstructured textual data into a structured format suitable for machine learning classification. The conversion process is represented as [98]:

$$V = T(D) \tag{6.1}$$

where:

- V: Vectorized form of the text data.
- T: Comprehensive text transformation function.
- D: Original corpus of unstructured textual documents.

The text transformation function T includes several key steps:

- Tokenization, which breaks down the text into individual word units.
- Normalization, which ensures consistent formatting of the text.
- Vectorization using Term Frequency–Inverse Document Frequency (TF-IDF), which converts the textual data into numerical vectors. These vectors capture the importance of each term within the documents, allowing the machine learning algorithms to process the textual data effectively [99].

This approach ensures that the raw unstructured data is systematically converted into a structured, numerical format that can be effectively utilized by machine learning algorithms.

Classifier Selection and Dataset Division

The Naïve Bayes classifier was chosen for this study due to its efficiency and effectiveness in handling high-dimensional textual data. The model's assumption of feature independence given the class label simplifies the classification process, making it computationally efficient, which is particularly advantageous in real-time defect detection scenarios. While other models, such as Support Vector Machines (SVM) or Random Forests, could potentially offer higher accuracy, the Naïve Bayes model provided a favorable balance between performance and computational cost, especially considering the relatively small size of our dataset.

The dataset was randomly divided into training and testing sets, with 70% of the data used for training the model and the remaining 30% reserved for testing. This split ensures

that the model is trained on a substantial portion of the data while providing a robust evaluation of its generalization performance.

Data Preprocessing

Data preprocessing involves a sequence of steps to clean and standardize the text data D, transforming it into a refined dataset $D_{\text{processed}}$:

$$D_{\text{processed}} = f_4(f_3(f_2(f_1(D)))) \tag{6.2}$$

where each preprocessing function f_i corresponds to a specific operation defined as follows:

- **Data Cleaning** $f_1 \equiv f_{\textbf{clean}}$**:** This step removes extraneous characters and symbols, including punctuation, numbers, and special characters that do not contribute to the textual analysis.
- **Tokenization** $f_2 \equiv f_{\textbf{tokenize}}$**:** This step breaks down the text into individual word units using NLTK's word tokenizer to split the text into meaningful words.
- **Stop Word Removal** $f_3 \equiv f_{\textbf{stopword}}$**:** This step eliminates frequently used words that offer little analytical value. The NLTK stop word list is used to filter out common English words.
- **Stemming/Lemmatization** $f_4 \equiv f_{\textbf{stem/lem}}$**:** This step reduces words to their root form to ensure consistency in analysis. The WordNet lemmatizer is employed to convert words to their base forms.

The four preprocessing functions f_1, f_2, f_3, and f_4 are sequentially applied to the raw data to ensure that the classifier operates on text data devoid of irrelevant variability, sharpening the focus on meaningful textual patterns [100, 101].

Advanced Analytical Techniques

Naïve Bayes Classifier The Naïve Bayes classifier is a probabilistic machine learning model based on Bayes' theorem. It assumes that the features are conditionally independent given the class label. This classifier is well-suited for high-dimensional textual data

due to its simplicity and efficiency. The conditional independence assumption reduces the computational complexity, making it effective for text classification tasks.

The Naïve Bayes classifier calculates the posterior probability $P(C_k|X)$ for each class C_k given a set of features $X = \{x_1, x_2, \ldots, x_n\}$ and assigns the class label with the highest probability. The posterior probability is given by [102, 103]:

$$P(C_k|X) = \frac{P(C_k) \prod_{i=1}^{n} P(x_i|C_k)}{P(X)} \tag{6.3}$$

where:

- $P(C_k|X)$: Posterior probability of class C_k given the feature vector X.
- $P(C_k)$: Prior probability of class C_k.
- $P(x_i|C_k)$: Likelihood of feature x_i given class C_k.
- $P(X)$: Evidence or normalizing constant.

Latent Class Analysis (LCA) provides a model-based clustering framework that assigns probabilities to observed patterns, represented by [97]:

$$P(X_i|\Theta) = \sum_{k=1}^{K} \gamma_k \prod_{j=1}^{J} \phi_{ij} \tag{6.4}$$

where:

- $P(X_i|\Theta)$: Probability of observing the data X_i given the parameters Θ.
- γ_k: Probability of the k-th latent class.
- ϕ_{ij}: Probability of the j-th observed variable given the i-th latent class.
- Θ: Set of model parameters.
- K: Number of latent classes.
- J: Number of observed variables.
- X_i: Observed data for the i-th observation.

The clustering process in LCA involves the following steps:

1. Initialization: Assume K clusters and initialize the parameters Θ (e.g., means and variances for Gaussian distributions if using Gaussian Mixture Models for clustering).

2. E-Step (Expectation): Compute the posterior probabilities (also known as responsibilities) for each observation X_i belonging to each cluster C_k [102, 104]:

$$\gamma_{ik} = P(C_k|X_i, \Theta^{(t)}) = \frac{P(C_k|\Theta^{(t)})P(X_i|C_k, \Theta^{(t)})}{\sum_{l=1}^{K} P(C_l|\Theta^{(t)})P(X_i|C_l, \Theta^{(t)})} \tag{6.5}$$

where:

- γ_{ik}: Posterior probability of the i-th observation belonging to the k-th cluster.
- C_k: The k-th latent class.
- $\Theta^{(t)}$: Current estimate of the parameters at iteration t.

3. M-Step (Maximization): Update the parameters Θ to maximize the expected log-likelihood based on current responsibilities [102, 105]:

$$\Theta^{(t+1)} = \arg\max_{\Theta} \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik} \log P(X_i, C_k|\Theta) \tag{6.6}$$

where:

- N: Total number of observations.
- $\Theta^{(t+1)}$: Updated parameter estimates at iteration $t + 1$.

4. Convergence: Iterate the E-step and M-step until convergence, i.e., until the changes in the log-likelihood are below a certain threshold.

The latent class assignments from LCA are then used as additional features for the Naïve Bayes classifier.

Latent Semantic Analysis (LSA) facilitates the extraction and representation of contextually meaningful aspects of language. It involves transforming the term-document matrix $T_{m\times n}$ into a reduced semantic space using Singular Value Decomposition (SVD). The term-document matrix $T_{m\times n}$, where the subscript $m \times n$ represents its dimensions, is a

matrix where each row represents a term and each column represents a document, with entries typically being the frequency or TF-IDF score of terms in documents. The transformation is represented as [106]:

$$T_{m\times n} = U_{m\times r}\Sigma_{r\times r}V^T_{r\times n} \tag{6.7}$$

where:

- $T_{m\times n}$: Original term-document matrix with m terms and n documents.
- $U_{m\times r}$: Orthogonal matrix where columns are the left singular vectors, representing term associations.
- $\Sigma_{r\times r}$: Diagonal matrix with singular values, representing the importance of each latent semantic dimension.
- $V^T_{r\times n}$: Transpose of an orthogonal matrix $V_{n\times r}$, where columns of V (rows of V^T) are the right singular vectors, representing document associations.

The LSA process works as follows:

1. Term-Document Matrix Creation: Construct a term-document matrix $T_{m\times n}$ from the text data, where each entry T_{ij} indicates the presence or weight (e.g., TF-IDF score) of term i in document j.

2. Singular Value Decomposition (SVD): Decompose $T_{m\times n}$ into three matrices $U_{m\times r}$, $\Sigma_{r\times r}$, and $V^T_{r\times n}$. This decomposition identifies the latent structure in the term-document matrix.

3. Dimensionality Reduction: Retain the top r singular values and corresponding vectors, reducing the dimensionality of the data. This results in a truncated matrix $\Sigma_{r\times r}$ and corresponding $U_{m\times r}$ and $V^T_{r\times n}$ matrices, capturing the most significant patterns.

4. Semantic Space Representation: The reduced matrices $U_{m\times r}$, $\Sigma_{r\times r}$, and $V^T_{r\times n}$ together form a lower-dimensional representation of the original term-document matrix, capturing the latent semantic relationships between terms and documents.

By transforming the term-document matrix into this reduced semantic space, LSA captures the underlying semantic structure of the text data, revealing contextually meaningful patterns and associations that are not apparent from the original high-dimensional data.

Long Short-Term Memory (LSTM) Networks are used to capture sequential dependencies and contextual information in the text data. The process involves:

1. Tokenizing the text data and converting it into sequences of word embeddings.

2. Training an LSTM network on these sequences to capture the temporal dependencies.

3. Extracting the hidden states or the final output of the LSTM as feature vectors representing the sequential information in the text.

Combining Features from LCA, LSA, LSTM, and Naïve Bayes The combined features from LCA, LSA, LSTM are used as input to the Naïve Bayes classifier:

- LCA provides the latent class features.
- LSA provides the latent semantic features.
- LSTM provides the sequential and contextual features.
- These features are concatenated to form a comprehensive feature vector.
- The Naïve Bayes classifier uses these combined features to predict the most likely defective component.

This comprehensive analytical approach enables us to uncover deeper insights into the defect patterns described in the textual data and significantly improves the accuracy of our defect predictions [107, 108].

Latent Class Analysis (LCA): The LCA was approximated using the Latent Dirichlet Allocation (LDA) model from Scikit-learn. We configured the LDA model to identify 10 latent clusters ($n\_components = 10$) from the TF-IDF vectors. To ensure reproducibility, we set the random state parameter to 42. The model was trained on precomputed TF-IDF vectors to identify latent clusters that represent underlying patterns in the defect descriptions.

6.1.4 Results and Analysis

Dataset Overview

The dataset used for this study comprises over 1,000,000 synthetic records, simulating operators' observations during the manufacturing process. Each record includes details such as the year, month, week, dates, batch number, defect description, replaced component, and specific electrical failures. The total number of tokens (words) across all documents is 7,151,459, indicating the extensive amount of textual data processed. The TF-IDF vectorization yielded 20 unique features, highlighting the diverse vocabulary used in defect descriptions..

The total number of tokens (words) across all documents is 7,151,459, indicating the extensive amount of textual data processed. The TF-IDF vectorization of the text resulted in 20 unique features (terms), reflecting the diverse vocabulary used in defect descriptions. The dataset comprises a total of 1,001,000 documents, with an average document length of 7.14 tokens.

This comprehensive dataset provides a robust foundation for the subsequent analysis, enabling the application of advanced text analytics and machine learning techniques to enhance defect detection and characterization in electronics manufacturing.

Classification Performance Metrics

To assess the classification performance of the Naïve Bayes model, precision (Pr), recall (R), F1-score ($F1$), and accuracy (Acc) were computed for each class. Precision measures the ratio of correctly predicted positive instances to the sum of all predicted positives [102, 106]:

$$Pr = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{6.8}$$

Recall quantifies the proportion of correctly predicted positive instances relative to all actual positives:

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6.9}$$

The F1-score, the harmonic mean of precision and recall, provides a balanced evaluation of both metrics:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.10}$$

Accuracy measures the proportion of correctly classified instances overall:

$$Acc = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} \tag{6.11}$$

The classification performance of the Naïve Bayes classifier was evaluated using standard metrics such as precision, recall, and F1-score. Table 6.1 presents a summary of these metrics for each component class. The classifier achieved an overall accuracy of 86%, with individual class accuracies ranging from 80% to 93%. Notably, the classifier demonstrated high precision and recall for components such as Field-Programmable Gate Array (FPGA) and Inductor, which suggests strong performance in identifying failures within these categories. These results affirm the classifier's robustness in distinguishing various component failures based on textual descriptions.

Table 6.1: Naïve Bayes Classification Report

Class	Precision	Recall	F1-Score	Support (Instances)
Capacitor	0.81	0.92	0.86	39296
Connector	0.73	0.92	0.82	42742
Diode	0.91	0.74	0.82	34500
FPGA	0.89	0.93	0.91	36943
Inductor	0.93	0.89	0.91	36643
Jumper cable	0.92	0.87	0.89	36686
Resistor	0.91	0.80	0.85	39875
Transistor	0.88	0.81	0.85	33615
Accuracy		0.86 (300300 instances)		
Macro avg	0.87	0.86	0.86	300300
Weighted avg	0.87	0.86	0.86	300300

Confusion Matrix Analysis

The confusion matrices, shown in Tables 6.2 and 6.3, offer a detailed breakdown of the classifier's performance, displaying the counts of true positive, false positive, and false negative predictions for each class. The high diagonal values, particularly for components like Capacitor and Connector, highlight the classifier's accuracy in predicting these failures. However, some misclassifications are evident, such as 1,204 instances where Capacitors were incorrectly classified as Connectors. These misclassifications suggest areas where the model could be further refined to improve accuracy.

Table 6.2: Confusion Matrix - Part 1

Actual	**Predicted**			
	Capacitor	**Connector**	**Diode**	**FPGA**
Capacitor	36464	1204	0	300
Connector	0	39113	924	625
Diode	265	5108	25767	1503
FPGA	884	300	0	33956
Inductor	1149	1203	618	316
Jumper Cable	634	865	645	846
Resistor	942	5087	0	298
Transistor	4436	283	301	313

In Table 6.2, we observe that the classifier accurately predicts the majority of Capacitor failures, as evidenced by the high diagonal matrix entry of 36,464. Similarly, Connector failures are correctly predicted 39,113 times. However, some misclassifications are noted, such as 1,204 instances where Capacitors were incorrectly classified as Connectors. A zero value in the matrix entry for Capacitor misclassified as Diode indicates that no instances of misclassification occurred for this specific pairing.

Table 6.3: Confusion Matrix - Part 2

Actual	Predicted			
	Inductor	**Jumper Cable**	**Resistor**	**Transistor**
Capacitor	0	284	319	915
Connector	657	310	602	548
Diode	633	0	904	303
FPGA	311	615	298	320
Inductor	32484	592	284	0
Jumper Cable	290	31952	570	897
Resistor	286	626	32022	614
Transistor	309	286	312	27406

Table 6.3 continues the analysis, highlighting further classification results. For example, Inductor failures are correctly identified 32,484 times, but 915 Capacitors are misclassified as Transistors. Such misclassifications suggest areas for potential improvement in the classifier's performance.

These confusion matrices help in understanding the specific strengths and weaknesses of the classifier, providing insights into where further refinements are needed.

Precision-Recall Curve Analysis

The precision-recall curve for each component class, as presented in Figure 6.1, provides a nuanced understanding of the classifier's performance. Most classes achieved an area under the curve (AUC) exceeding 0.80, underscoring strong performance across the board. Notably, the Inductor class demonstrated the highest AUC value at 0.91, reflecting excellent classifier performance in distinguishing this component. In contrast, the Connector class, with an AUC of 0.82, indicates potential areas for improvement in balancing precision and recall.

These curves underscore the classifier's ability to maintain high precision across varying recall levels, with some classes, such as Diode and Connector, showing a sharper drop in precision as recall increases, which points to areas for further enhancement.

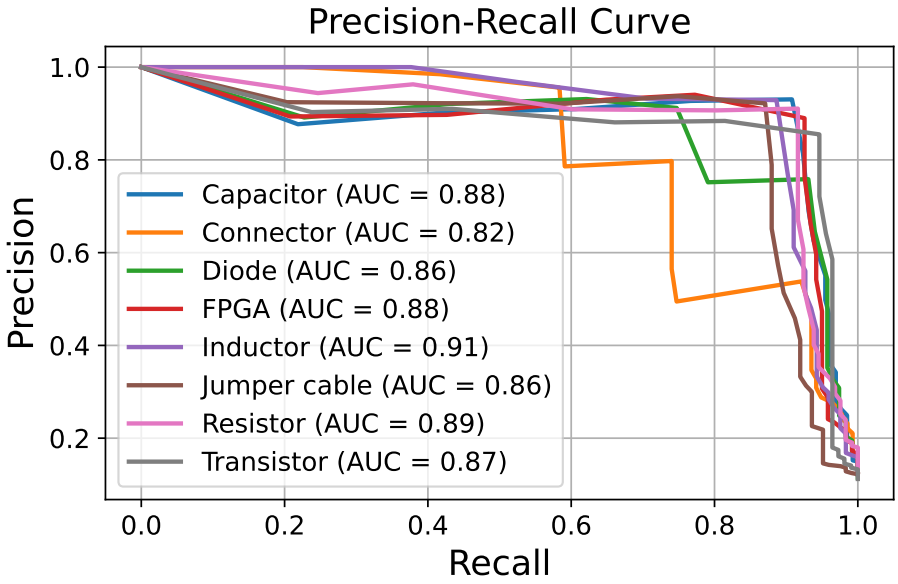

Figure 6.1: Precision-Recall curves for component classification, showing class-wise performance across varying thresholds. The Inductor class achieved the highest area under the curve (AUC = 0.91), indicating excellent discriminative ability, while the Connector class (AUC = 0.82) reflects comparatively lower performance. These curves illustrate the trade-offs between precision and recall for each class.

The precision-recall curves highlight the trade-off between precision and recall, emphasizing the classifier's ability to maintain high precision while achieving high recall for most classes. For instance, the Inductor class achieves the highest AUC value of 0.91, indicating excellent performance in distinguishing this component. On the other hand, the Connector class, with an AUC of 0.82, shows comparatively lower performance, suggesting room for improvement in precision and recall balance.

The curves also illustrate that while some classes maintain high precision across varying recall levels, others show a significant drop in precision as recall increases. This is evident

for the Connector and Diode classes, where precision decreases more sharply compared to other classes as recall improves.

Overall, the precision-recall curves provide valuable insights into the classifier's effectiveness and highlight areas where performance can be further enhanced, particularly in balancing precision and recall for certain component classes.

Feature Importance and Cross Correlations Analysis

In this section, we present the results of our feature importance analysis and the cross-correlations between features, highlighting key insights derived from the textual data.

Figure 6.2 illustrates the top 20 most important features contributing to the classification decisions. The feature importance was determined based on the average absolute difference between the log probabilities of each class. Key terms such as "causing", "inspection," "drops", "unintended", and "capacitor" were identified as highly predictive of component failures. This analysis underscores the relevance of specific defect descriptions in accurately determining the faulty components.

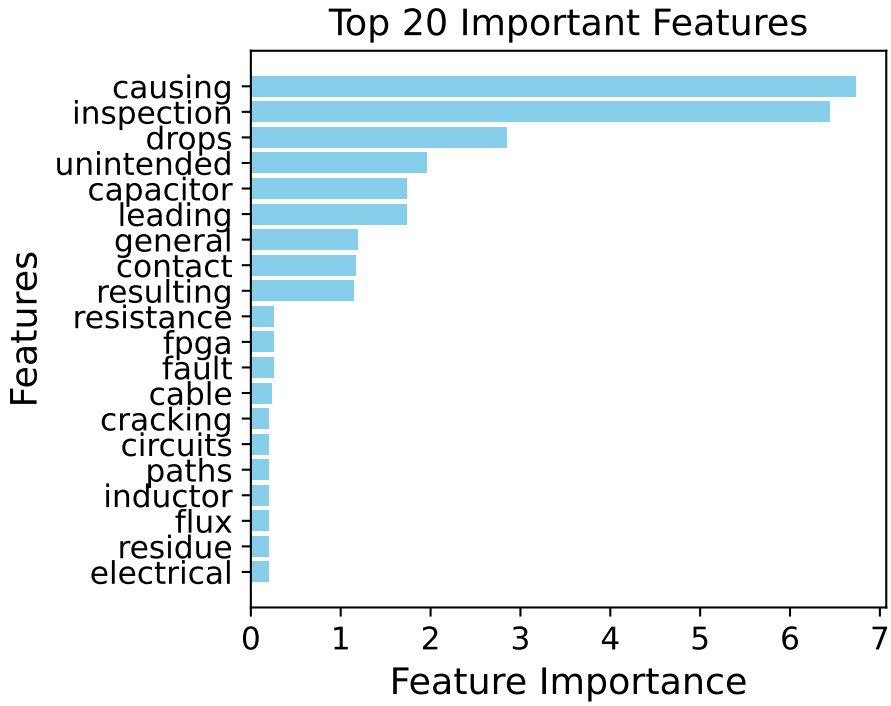

Figure 6.2: Top 20 most important textual features contributing to component classification, based on the average absolute log probability differences across classes. High-importance terms such as "causing," "inspection," and "capacitor" indicate strong influence on model decisions and highlight the critical role of domain-specific language in defect diagnostics.

These features highlight the relevance of specific defect descriptions in determining the faulty components. The high importance of these terms underscores their critical role in accurately identifying and diagnosing defects within the manufacturing process.

Additionally, the cross-correlations between features were analyzed to understand the relationships between different defect descriptions. Figure 6.3 displays the correlation matrix for the top correlated features. The matrix reveals significant correlations between certain defect terms, such as "connectivity" with "stress" and "thermal", "fillet" with "overheating" and "potential", and "residue" with "excessive". These correlations could be indicative of common failure modes or related issues in the manufacturing process. Understanding these relationships helps in identifying potential root causes of defects and improving overall defect detection accuracy.

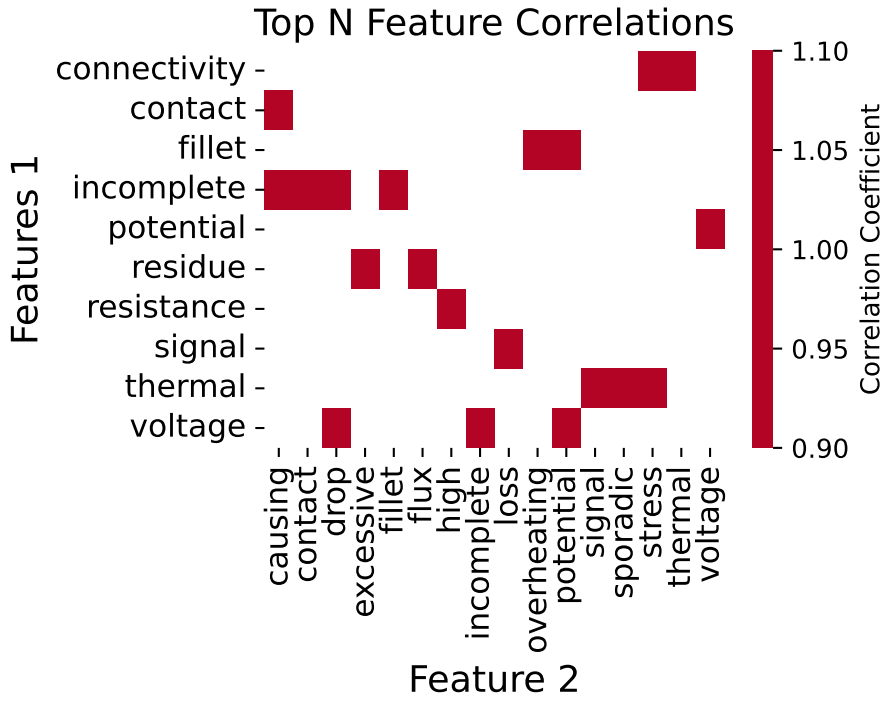

Figure 6.3: Correlation matrix of top co-occurring textual features in defect reports. Strong correlations (e.g., between "connectivity" and "thermal", or "fillet" and "overheating") suggest the presence of compound failure modes and interdependent defect descriptions, aiding root cause analysis in manufacturing quality control.

Cluster Distribution Analysis

The distribution of clusters identified by Latent Class Analysis (LCA) is shown in Figure 6.4. Each cluster was named based on the most frequent terms within the cluster, providing insights into common patterns and themes in defect descriptions.

The analysis revealed several key clusters, each representing a distinct category of defects. Tables 6.4 and 6.5 provide a summary of the occurrences and relative percentages of each cluster.

Table 6.4: Cluster Statistics - Occurrences

Cluster	Cluster Name	Occurrences
1	abnormal, path, unintended, behavior, circuit	137,107
2	abnormal, current, misalignment, resulting, short	81,218
3	residue, flux, excessive, general, unpredictable	52,812
4	intermittent, thermal, stress, sporadic, loss	175,931
5	voltage, causing, contact, drop, fillet	218,580
6	bridging, fault, unpredictable, electrical, performance	183,426
7	residue, flux, excessive, performance, variation	63,120
9	residue, flux, excessive, unpredictable, variation	88,806

Table 6.5: Cluster Statistics - Relative Percentages

Cluster	Cluster Name	Relative Percentage (%)
1	abnormal, path, unintended, behavior, circuit	13.70
2	abnormal, current, misalignment, resulting, short	8.11
3	residue, flux, excessive, general, unpredictable	5.28
4	intermittent, thermal, stress, sporadic, loss	17.58
5	voltage, causing, contact, drop, fillet	21.84
6	bridging, fault, unpredictable, electrical, performance	18.32
7	residue, flux, excessive, performance, variation	6.31
9	residue, flux, excessive, unpredictable, variation	8.87

The relative occurrences indicate that Cluster 5, which includes the complete set of terms "voltage," "causing," "contact," "drop," and "fillet," is the most frequent, accounting for 21.84% of the total defect descriptions. Cluster 6 follows with 18.32%, and Cluster 4 accounts for 17.58%. On the lower end, Cluster 3 represents 5.28% of the defect descriptions.

Figure 6.4 visually represents the distribution of these clusters, highlighting the prevalence of each type of defect. This distribution provides valuable insights for prioritizing quality control efforts in the manufacturing process.

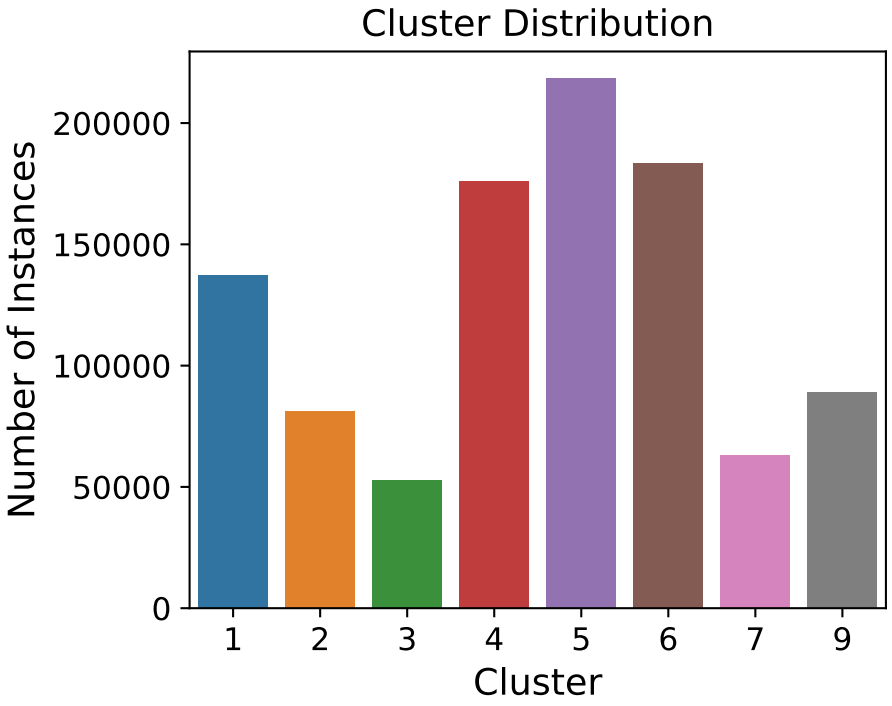

Figure 6.4: Distribution of defect report clusters derived from unsupervised text embedding. The figure highlights the relative frequency of each cluster, reflecting common patterns in defect types and aiding prioritization of quality control efforts across component categories.

t-SNE Cluster Visualization

The t-Distributed Stochastic Neighbor Embedding (t-SNE) technique is utilized to reduce high-dimensional text data into a two-dimensional space, allowing for visualization and analysis of clustering patterns among different component failures.

The t-SNE visualization in Figure 6.5 highlights the clustering of text data for different component failures. While clusters for components like Capacitor and Transistor overlap, indicating similarities in failure descriptions, other components such as Jumper Cable and FPGA form distinct clusters, suggesting unique descriptions.

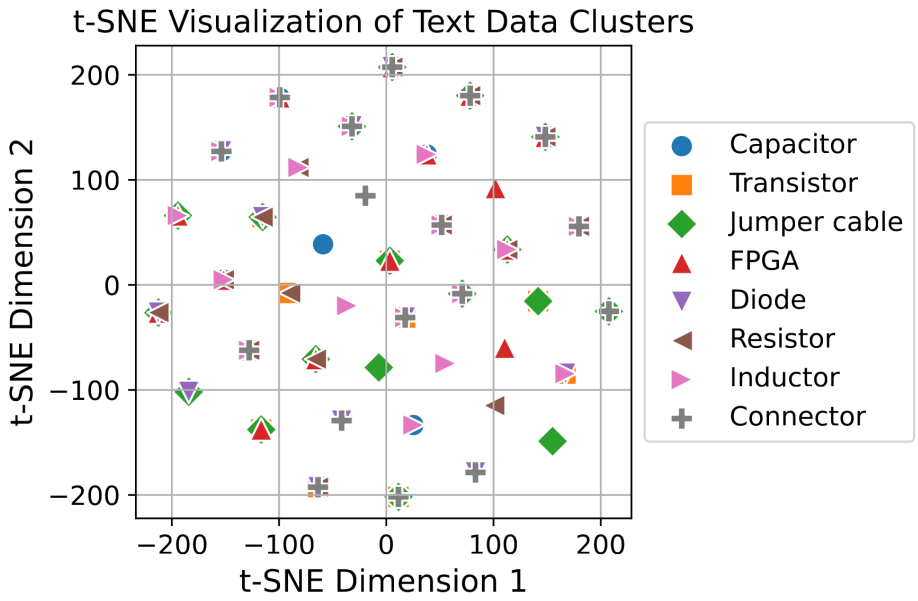

Figure 6.5: t-SNE projection of defect-related text embeddings, revealing clustering patterns across component failure types. Distinct clusters (e.g., Jumper Cable, FPGA) indicate unique linguistic signatures, while overlapping regions (e.g., Capacitor and Transistor) suggest semantic similarities in defect descriptions, potentially contributing to classifier confusion.

The distribution of data points is relatively even, with some components like Connector and Inductor showing more dispersed clusters, indicating varied descriptions. Overlapping clusters, such as those for Diode with other components, suggest potential areas for misclassification.

Overall, the t-SNE visualization demonstrates the classifier's ability to distinguish between different component failures. However, the observed overlaps indicate the need for further refinement in text preprocessing or feature extraction to improve classification accuracy.

Additionally, an analysis of the terms used in failure descriptions revealed that approximately 88% of the terms are unique to specific component types, highlighting the specificity of textual descriptions across different components.

The results presented clearly demonstrate the robustness of the Naïve Bayes classifier in accurately identifying and characterizing defects in electronics manufacturing. Through a comprehensive evaluation using quantitative metrics, confusion matrices, and precision-recall curves, the classifier's performance has been thoroughly assessed. The strong performance in key areas, such as the classification of FPGA and Inductor components, is particularly noteworthy. Additionally, the analysis of feature importance and clustering has provided valuable insights into the underlying patterns and relationships within the data. These findings not only validate the effectiveness of the Naïve Bayes approach but also highlight areas for potential enhancement, particularly in improving precision and recall for components like Connector and Diode.

6.1.5 Discussion

The results indicate that the Naïve Bayes model effectively distinguishes between component defects and predicts replacements based on textual defect descriptions.

Model Performance and Implications

1. Component-Specific Insights: The Naïve Bayes model exhibited strong predictive power for some components. For instance, capacitors and inductors were accurately classified due to their distinctive patterns of defect descriptions, leading to precise predictions. The pattern recognition capability reveals that defect descriptions for these components often contain identifiable features that differentiate them.
2. Strategic Quality Control: High classification accuracy enables proactive quality control strategies. By identifying likely failures during manufacturing or detecting incoming defective materials, facilities can enhance inspections and improve production line efficiency, minimizing defects and reducing downtime.

3. Cluster Differentiation: t-SNE visualizations show well-separated clusters for components like "Connector" and "Capacitor" but significant overlap for others like "Jumper Cable" and "Resistor." Improved feature differentiation among these overlapping clusters would minimize misclassifications and enhance predictive performance.

Comparison with State-of-the-Art Methods: The Naïve Bayes classifier was chosen for its computational efficiency and simplicity, making it well-suited for high-dimensional textual data. Unlike more complex models such as Support Vector Machines (SVM) or Neural Networks (NN), Naïve Bayes provides faster training and lower computational demands, which is beneficial for real-time applications. However, its assumption of feature independence may reduce accuracy compared to models that account for feature interactions. Despite this, Naïve Bayes proved effective in our study, where simplicity and speed were key priorities.

Limitations and Future Work

Despite promising results, the models face certain limitations:

1. Dataset Bias and Inclusion: The current dataset only includes instances where replacements were made, potentially introducing bias. This limits the model's ability to generalize to cases where no replacement is necessary. Including data where replacements were not needed would provide a balanced and comprehensive view of defects, enabling models to predict both necessary and unnecessary replacements accurately.
2. Feature Differentiation: Components like "Diode" and "Inductor" frequently overlap due to textual similarity in defect descriptions, leading to misclassifications. Enhanced feature extraction methods would differentiate these components more effectively.
3. Future Model Enhancements: Ensemble methods like random forests or gradient boosting machines could better capture subtle relationships between textual features. Incorporating additional training data or using data augmentation techniques would enhance model robustness.

Potential for Real-Time Implementation

While this study focuses on the application of text analytics for defect detection and characterization using historical data, the methodologies developed have significant potential for real-time implementation. Integrating real-time text analytics into manufacturing processes could enhance defect detection and allow for immediate corrective actions, minimizing production delays and improving overall efficiency. Future work could explore how these methodologies can be adapted and integrated into real-time systems within manufacturing environments.

6.1.6 Chapter Summary

This case study explored the use of text analytics and machine learning for quality control in electronics manufacturing, focusing on the classification of Printed Circuit Board (PCB) defects using unstructured technician reports. By applying natural language processing techniques and a Naïve Bayes classifier, the framework effectively transformed descriptive narratives into structured data, enabling accurate prediction of component-level failures. The approach demonstrated strong classification performance for common components such as capacitors and resistors, validating the potential of human-in-the-loop AI for augmenting defect detection systems.

Key contributions include demonstrating the feasibility of integrating unstructured text data into automated quality control pipelines and highlighting the benefits of interpretable models for manufacturing environments with limited data infrastructure. While the Naïve Bayes model offered efficiency and simplicity, its assumption of feature independence and reliance on historical data suggest the need for adaptive, context-aware models in future work.

This study extends the broader dissertation theme of System of Systems (SoS) optimization by integrating human-generated data into decision-making processes. The next **Chapter 7** shifts focus to the healthcare domain, where a multi-agent reinforcement learn-

ing framework is applied to an Artificial Intelligence of Medical Things (AIoMT)-enabled environment. This upcoming case study explores decentralized coordination across hospitals, telemedicine platforms, and wearable technologies, highlighting the role of RL-based policy learning in dynamic, resource-constrained healthcare systems.

Chapter 7

Reinforcement Learning for Healthcare System-of-Systems

7.1 Multi-Agent Coordination in AIoMT-Enabled Healthcare Networks

7.1.1 Problem Overview and Context

The increasing complexity of modern healthcare delivery, especially with the rise of decentralized and AI-enabled technologies, has led to the emergence of healthcare ecosystems that function as Systems of Systems (SoS). These environments include hospitals, clinics, telemedicine units, and wearable devices—each operating autonomously but required to collaborate for system-wide efficiency, responsiveness, and adaptability. Traditional centralized healthcare infrastructures often fail to support such distributed models, particularly in resource-constrained settings or during high-stress scenarios such as pandemics or system overloads.

The integration of Artificial Intelligence of Medical Things (AIoMT) has accelerated the adoption of intelligent, data-driven decision-making across healthcare operations. However, significant challenges remain in achieving real-time coordination across diverse and heterogeneous entities. Issues such as fragmented communication, lack of interoperability, and localized resource limitations hinder effective system-wide responses. Moreover, healthcare entities differ in their control mechanisms, autonomy levels, and inter-agent dependencies, necessitating adaptive architectures that can support multiple coordination models.

This case study addresses the challenge of coordinating AIoMT-enabled healthcare entities under various constituent system configurations—Directed, Acknowledged, Collaborative, and Virtual. Each configuration embodies a distinct mode of control, autonomy,

and communication. The study introduces a Multi-Agent Reinforcement Learning (MARL) framework using independent Q-learning agents to model and optimize decision-making across these constituent systems. Agents learn policies that enhance local performance while improving global outcomes such as cooperation, resource utilization, and system efficiency.

The motivation for this work lies in enabling adaptive, decentralized coordination mechanisms suitable for real-world healthcare SoS, especially in dynamic and constrained environments. By evaluating different constituent system structures, this study provides insights into how coordination typologies influence emergent behavior, policy convergence, and operational performance in AI-driven healthcare delivery.

7.1.2 Research Objectives and Questions

This case study aims to investigate how different constituent system configurations influence coordination, learning dynamics, and system-wide performance in healthcare Systems of Systems (SoS) enabled by the Artificial Intelligence of Medical Things (AIoMT). By leveraging Multi-Agent Reinforcement Learning (MARL), the study models decentralized decision-making across heterogeneous healthcare agents—including hospitals, telemedicine platforms, clinics, and wearable monitoring units.

The primary research objectives are as follows:

- **Objective 1:** Develop a modular MARL-based framework capable of modeling healthcare entities as autonomous agents within an SoS architecture.
- **Objective 2:** Implement and compare four distinct constituent system models—Directed, Acknowledged, Collaborative, and Virtual—to evaluate how structural coordination modes affect system-level learning and performance.
- **Objective 3:** Quantitatively assess key performance metrics, including cumulative reward, resource utilization, cooperation frequency, and convergence speed across 1000 simulated episodes.

- **Objective 4:** Identify trade-offs between autonomy and coordination under dynamic and resource-constrained conditions relevant to real-world healthcare applications.

Based on these objectives, the study addresses the following research questions:

- **RQ1:** How do different constituent system configurations affect emergent coordination behavior and policy learning in healthcare SoS environments?
- **RQ2:** Which SoS configuration(s) yield the most efficient trade-off between agent-level autonomy and system-level performance under MARL-based coordination?
- **RQ3:** To what extent can Q-learning agents optimize decentralized decision-making in AIoMT-enabled healthcare systems with partial observability and constrained resources?
- **RQ4:** What implications do the observed behaviors and outcomes have for the design of scalable and adaptive healthcare coordination architectures?

7.1.3 Framework Overview

To address the coordination, adaptability, and decision-making limitations prevalent in AIoMT-enabled healthcare ecosystems, we propose an integrated *System-of-Systems* (SoS) architecture underpinned by *Multi-Agent Reinforcement Learning* (MARL). This framework enables distributed, goal-driven collaboration among heterogeneous member systems—such as hospitals, clinics, telemedicine platforms, and wearable devices—through dynamic policy learning. The use of MARL empowers these agents to learn optimal strategies over time, adapt to changes in resource availability and patient demand, and improve system-level efficiency and care delivery outcomes.

SoS Architecture for AIoMT-Enabled Healthcare

The proposed System-of-Systems (SoS) framework is composed of four distinct *constituent systems*—Directed, Acknowledged, Collaborative, and Virtual—each representing a unique coordination model for integrating six interacting agents (e.g., hospitals, clinics, telemedicine platforms, and wearable devices). These constituent systems define how con-

trol, autonomy, and communication are distributed among agents, collectively forming the overarching SoS. Each model reflects a real-world healthcare integration archetype and is evaluated independently within the simulation environment.

- **Collaborative constituent system:** All agents operate autonomously and engage in peer-to-peer coordination. Local decision-making is emphasized, with agents sharing information, negotiating care pathways, and co-adapting based on system feedback. This model is well-suited to environments where decentralized response and cross-institutional collaboration are essential.
- **Directed constituent system:** A centralized authority exercises complete control over all agents. Entities receive and execute directives regarding patient routing, staff deployment, and service scheduling. While structured and hierarchical, this model often suffers in responsiveness and adaptability under non-stationary healthcare demands.
- **Acknowledged constituent system:** A hybrid coordination model where agents act autonomously but are guided by high-level goals or policies issued by a central coordinator (e.g., a national health authority). Local decision-making is preserved, but actions are shaped by shared care protocols, enabling a balance between flexibility and system-wide coherence.
- **Virtual constituent system:** Comprised of loosely coupled digital agents—such as wearables and telemedicine services—that communicate via shared data streams rather than through physical infrastructure. Coordination is asynchronous and emergent, prioritizing data-driven triage, remote diagnostics, and continuous monitoring with minimal overhead.

In each simulation, the same six agents are organized under one of the four constituent systems—Directed, Acknowledged, Collaborative, or Virtual—to enable comparative evaluation within the overall SoS framework. This design isolates the impact of coordination mechanisms on emergent behavior. The results—such as cumulative rewards, Q-value con-

vergence, cooperation counts, and resource utilization trends—demonstrate how each constituent system influences agent learning dynamics, coordination behavior, and operational efficiency in AIoMT-enabled healthcare environments.

Agent Configuration and Role Assignment: To ensure consistent evaluation across constituent systems, each configuration simulates six member systems as agents, representing a realistic distribution of healthcare roles. These agents remain structurally identical across the Directed, Acknowledged, Collaborative, and Virtual constituent systems, with their coordination mechanisms governed by the overarching SoS architecture. The mapping is summarized in Tables 7.1 and 7.2.

Table 7.1: Agent Configuration: Directed and Acknowledged Systems

System Type	Agent ID	Entity	Primary Role
Directed System	Agent 0	Hospital A	Centralized scheduling and intake
	Agent 1	Hospital B	Diagnostics and routing
	Agent 2	Clinic A	Local triage
	Agent 3	Clinic B	Rural care access
	Agent 4	Hospital C	Overflow handling
	Agent 5	Hospital D	Surplus resource sharing
Acknowledged System	Agent 0	Hospital A	Shared protocol compliance
	Agent 1	Hospital B	Staff and diagnostic optimization
	Agent 2	Clinic A	Emergency services with shared rules
	Agent 3	Clinic B	Peripheral coordination
	Agent 4	Telemedicine Platform	Remote triage under policy
	Agent 5	Wearable Node	Vitals reporting with oversight

Table 7.2: Agent Configuration: Collaborative and Virtual Systems

System Type	Agent ID	Entity	Primary Role
Collaborative System	Agent 0	Wearable 1	Patient monitoring
	Agent 1	Wearable 2	Vital tracking
	Agent 2	Telemed Platform 1	First-line triage
	Agent 3	Telemed Platform 2	Agent communication
	Agent 4	Hospital A	Joint planning and intake
	Agent 5	Hospital B	Load balancing and diagnostics
Virtual System	Agent 0	Hospital C	AI-guided treatment support
	Agent 1	Rural Clinic E	Community screening
	Agent 2	Clinic B	Minor condition triage
	Agent 3	Telemed Hub	Central virtual triage
	Agent 4	Wearable 1	Vitals collection
	Agent 5	Wearable 2	Alerts and escalation

Multi-Agent Reinforcement Learning (MARL) Integration

To optimize resource distribution and adapt to environmental dynamics, each agent employs value-based reinforcement learning. Specifically, a Q-learning mechanism enables agents to iteratively refine decisions based on local observations and environmental feedback.

Agent Roles and Learning Objectives

a) Hospitals: Optimize bed allocation, personnel deployment, and patient admission. b) Wearable Devices: Monitor vitals, detect anomalies, and communicate status to clinical

systems. c) Telemedicine Platforms: Manage virtual consultations and remote diagnostics, triaging cases for escalation.

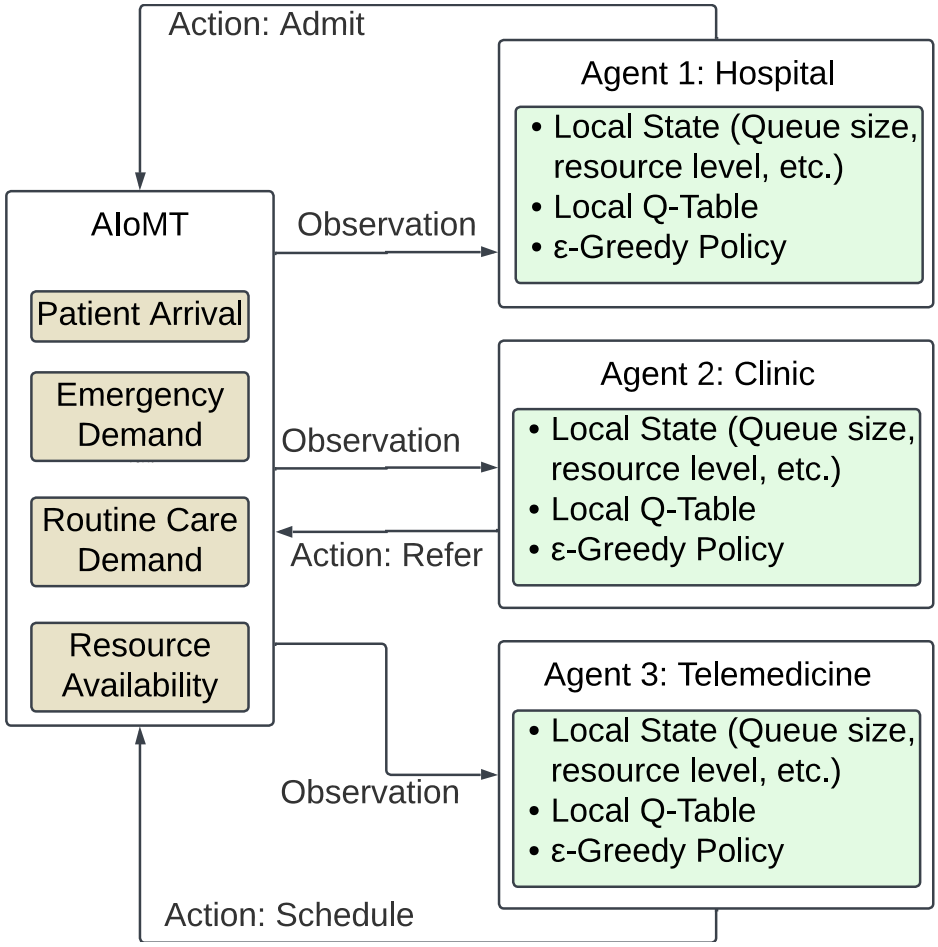

Figure 7.1: System-of-Systems (SoS) architecture for AIoMT-enabled healthcare. Six agents are deployed under four coordination models—Directed, Acknowledged, Collaborative, and Virtual—each shaping interaction patterns, autonomy, and learning behavior.

Each agent seeks to maximize long-term healthcare outcomes by learning policies that balance immediate system demands with strategic performance objectives.

Table 7.3: Constituent System Breakdown in the Proposed MARL-Based Healthcare Framework

Constituent System	**Coordination Model**	**Control Flow**	**Example Behavior in AIoMT Healthcare**
Directed System	Centralized	Central authority dictates decisions, resource flows, and agent behavior.	Lead hospital assigns loads and resources to clinics and platforms; agents execute without planning.
Acknowledged System	Semi-centralized	Independent agents follow shared policies with periodic synchronization.	Entities optimize locally but align with protocols and data-sharing mandates.
Collaborative System	Decentralized	Equal agent autonomy with peer-to-peer negotiation and coordination.	Clinics and hospitals share diagnostics and negotiate transfers during emergencies.
Virtual System	Loosely coupled, asynchronous	Agents coordinate via real-time data, often without physical integration.	Wearables stream vitals to cloud agents that trigger alerts and interventions asynchronously.

7.1.4 Methodology

This section outlines the methodological foundation of the proposed Multi-Agent Reinforcement Learning (MARL) framework, developed to enhance coordination, resource allocation, and patient care across AIoMT-enabled healthcare Systems-of-Systems (SoS). The framework leverages a value-based reinforcement learning approach—specifically, the Q-learning algorithm—across four distinct SoS coordination structures: Directed, Acknowl-

edged, Collaborative, and Virtual. Each architecture represents a unique organizational paradigm in distributed healthcare environments, reflecting varying levels of autonomy, control, and communication among constituent systems.

In this formulation, each agent (e.g., hospital, telemedicine unit, or wearable device) learns an individualized policy to optimize local decisions while contributing to emergent system-level behavior. The learning process unfolds over a sequence of episodic interactions within a simulated environment, where agents adapt their strategies in response to stochastic state transitions and feedback signals encoded as rewards.

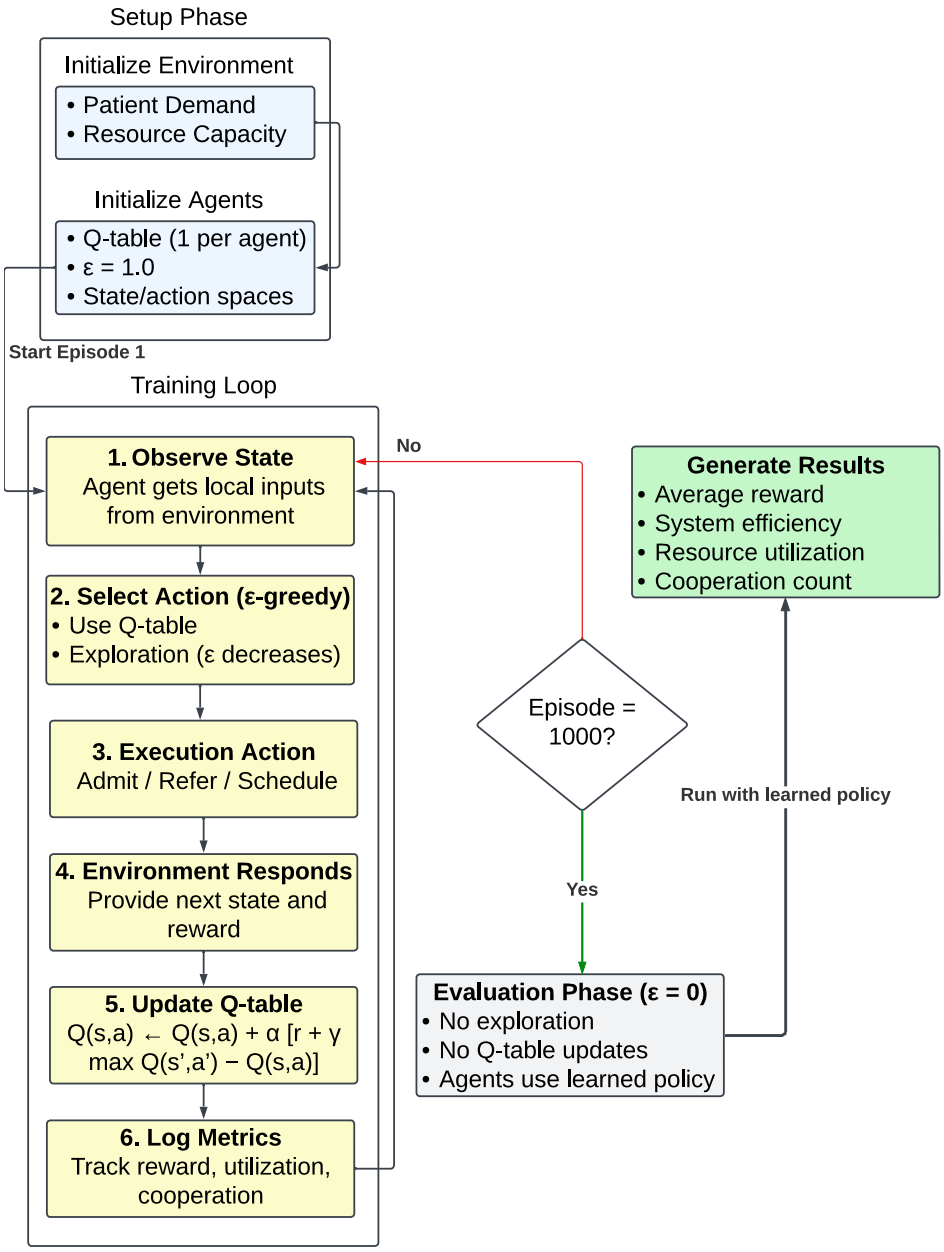

Figure 7.2: Training and evaluation process for the proposed MARL framework. Agents interact with the environment using an ϵ-greedy policy over 1000 episodes. After training, an evaluation phase (with $\epsilon = 0$) is used to assess learned agent behavior and system performance, including reward, efficiency, resource utilization, and cooperation count.

The training pipeline, illustrated in Figure 7.2, begins with exploration-driven interactions governed by an ϵ-greedy policy. Over the course of 1000 episodes, agents incrementally update their Q-values, refining their policies based on observed outcomes and peer behaviors.

Upon convergence, the system enters an evaluation phase ($\epsilon = 0$), during which exploitation dominates and the effectiveness of learned behaviors is assessed across multiple performance dimensions. These include cumulative rewards, cooperation levels, system efficiency, and resource utilization metrics—each of which is subsequently analyzed to compare emergent behavior across the constituent system configurations modeled within the overall SoS framework.

Reinforcement Learning Framework and Q-learning Algorithm

To enable adaptive decision-making in dynamic healthcare environments, each agent within the proposed Systems-of-Systems (SoS) framework employs a value-based reinforcement learning technique—Q-learning. This algorithm allows agents to iteratively estimate the expected utility of actions taken in specific system states, progressively refining their policies based on interaction with the environment. In doing so, agents strive to maximize long-term cumulative rewards that reflect desirable healthcare outcomes such as efficient resource usage and improved patient satisfaction.

The core of the Q-learning algorithm is the update of the action-value function $Q(s, a)$, which estimates the expected return for selecting action a in state s, and then following the agent's current policy thereafter. The update rule is defined as [109]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\Big) \tag{7.1}$$

where $\alpha \in (0, 1)$ is the learning rate that determines the weight given to new experiences, $R(s_t, a_t)$ is the reward received after executing action a_t in state s_t, and $\gamma \in [0, 1]$ is the discount factor controlling the importance of future rewards relative to immediate gains. The term $\max_{a'} Q(s_{t+1}, a')$ represents the maximum Q-value associated with the next state s_{t+1}, guiding the agent toward optimal future behavior.

As agents explore the environment and observe the consequences of their actions, they must strike a balance between exploring novel strategies and exploiting known successful

policies. This trade-off is governed by an ϵ-greedy exploration mechanism, where the action a_t at time t is selected according to [90]:

$$a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon, \\ \arg\max_{a'} Q(s_t, a') & \text{with probability } 1 - \epsilon \end{cases} \tag{7.2}$$

Initially, the exploration rate ϵ is set to a high value to encourage diverse action sampling. Over time, ϵ is decayed gradually—typically using a decay schedule of the form $\epsilon_t = \epsilon_0 \cdot e^{-\lambda t}$—which reduces random action selection and promotes reliance on learned policies.

This reinforcement learning framework enables each agent to autonomously discover effective behavior in the face of non-stationary dynamics and partial observability typical of healthcare environments. By learning from rewards that encode both local and system-wide objectives, agents adapt to diverse operational contexts and contribute to the emergence of coordinated, high-efficiency behavior within the overall SoS.

System Environment and Agent Definitions

Each constituent system modeled within the proposed System-of-Systems (SoS) framework is formulated as a Markov Decision Process (MDP), which provides a formal structure for capturing the interaction between agents and their dynamic healthcare environment [110]. The MDP for each constituent system is defined by the tuple (S, A, P, R), where S represents the state space, A the action space, P the transition probability function, and R the reward function. These components collectively enable agents to learn optimal policies through experience-based adaptation.

Each constituent system—Directed, Acknowledged, Collaborative, and Virtual—features a distinct composition of six agents to reflect realistic architectural differences in coordination strategies. For example, the Collaborative and Virtual systems include wearable and telemedicine agents more prominently, whereas the Directed and Acknowledged systems

emphasize hospitals and clinics. Tables 7.1 and 7.2 outline the specific agent assignments associated with each constituent system configuration.

The state space S captures the evolving system context at each time step t, encapsulating variables such as resource availability R_t, patient conditions P_t, and diagnostic statuses D_t. Each agent i has access to a localized subset of the global state, denoted as $s_t^i \subset S$, which reflects its individual operational context within the broader SoS environment.

The action space A comprises the set of possible decisions available to each agent. At every time step t, agent i selects an action $a_t^i \in A$, which may correspond to operations such as admitting or redirecting patients, allocating medical staff, scheduling diagnostic services, or adjusting remote monitoring protocols. The collective actions influence both local outcomes and system-level dynamics.

Transitions between states are governed by the stochastic transition function $P(s_{t+1}|s_t, a_t)$, which captures the probability of transitioning from state s_t to state s_{t+1} given action a_t. These transitions reflect the inherent uncertainty and variability in healthcare operations, such as fluctuations in patient arrival rates or equipment availability. In our implementation, these dynamics are simulated using randomized selections from a synthetically generated dataset, allowing controlled variation while maintaining realism.

The reward function $R(s_t, a_t)$ quantifies the immediate performance feedback associated with a given action a_t in state s_t. This feedback is aligned with critical healthcare objectives, including reducing patient wait times, optimizing resource utilization, and improving overall system responsiveness [111, 112].

Each agent seeks to maximize its expected cumulative reward over a finite time horizon T, with future outcomes discounted by a factor $\gamma \in [0, 1]$. Formally, the objective for agent i is to maximize the value function [90]:

$$J(\pi^i) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t^i)\right] \tag{7.3}$$

where π^i is the policy followed by agent i, and $\mathbb{E}$ denotes the expectation over all possible trajectories governed by the stochastic transition dynamics.

The discount factor γ is calibrated to prioritize short-term healthcare decisions while still valuing future benefits. A value close to 1 encourages long-term planning, whereas a smaller value biases decision-making toward immediate outcomes. This formulation supports adaptive behavior in both acute and chronic care scenarios, enabling agents to learn policies that balance present needs against evolving system states.

Performance Metrics and Evaluation

The evaluation of the proposed framework is based on a comprehensive set of individual and system-level metrics designed to capture the performance, adaptability, and coordination quality of agents operating under different constituent systems within the overarching System-of-Systems (SoS) architecture. These metrics serve as the foundation for analyzing emergent behavior across the simulation horizon and support comparative assessments of the Directed, Acknowledged, Collaborative, and Virtual constituent systems [113].

To capture per-episode reward dynamics across agents and episodes, we define three scalar summary statistics: the minimum, maximum, and average total reward for each episode t, denoted respectively as R^t_{min}, R^t_{max}, and $\bar{R}^t$. These are computed as:

$$R^t_{\text{min}} = \min_i \left(\sum_{\tau=1}^{T} r^i_\tau \right) \tag{7.4}$$

$$R^t_{\text{max}} = \max_i \left(\sum_{\tau=1}^{T} r^i_\tau \right) \tag{7.5}$$

$$\bar{R}^t = \frac{1}{|A|} \sum_i \sum_{\tau=1}^{T} r^i_\tau \tag{7.6}$$

where r_τ^i is the reward received by agent i at time step τ during episode t, and $|A| = 6$ is the total number of agents. These three values are visualized in the Results section as episodic reward trend curves to evaluate the consistency and robustness of learning across agents.

At the individual level, each agent accumulates rewards across episodes based on the effectiveness of its decisions. The cumulative reward, denoted R^i_{total}, reflects the agent's ability to align its actions with system objectives, particularly under conditions of constrained resource availability. To understand how agents balance the trade-off between exploitation and exploration, two behavioral indicators are monitored: the count of exploitative actions (where the agent selects the action that maximizes the Q-value) and the count of exploratory actions (where the agent selects a random action). These behavioral patterns provide insight into the stability and convergence of learned policies.

Cooperation counts are also recorded, capturing the frequency of interactions between agents that contribute to shared goals—such as transferring patients or sharing diagnostic feedback. These are especially informative in Collaborative and Acknowledged SoS configurations, where inter-agent coordination is structurally encouraged. Additionally, resource utilization U_t^i is measured for each agent at each time step t, quantifying how effectively available resources (e.g., hospital beds, staff, and diagnostic devices) are allocated over time.

At the system-wide level, aggregated metrics provide a macro-level perspective of overall system dynamics. One key metric is the system efficiency η_t, defined as the ratio of successful care delivery events (e.g., treated patients or resolved diagnostic queries) to the total resources consumed [111]:

$$\eta_t = \frac{O_t^{\text{successful}}}{R_t^{\text{total}}} \tag{7.7}$$

where $O_t^{\text{successful}}$ denotes the number of successful outcomes at time t, and R_t^{total} represents the aggregate resource consumption by all agents during the same interval.

The overall cooperation level is derived from the mean cooperation counts across all agents, providing a scalar measure of emergent collaborative behavior within the SoS. Simi-

larly, exploration rates are computed as the average frequency of exploratory actions across all agents and episodes, revealing the adaptability of each system configuration in the face of dynamic uncertainties.

To enable a clearer interpretation of episodic trends, especially under high-frequency variance, all time-series metrics are post-processed using a moving average filter. The smoothed value $\hat{X}_t$ for a given metric X at episode t is calculated as [114]:

$$\hat{X}_t = \frac{1}{w} \sum_{i=t-w}^{t} X_i \tag{7.8}$$

where w is the smoothing window size and X_i is the raw metric value at episode i. This smoothing process preserves long-term dynamics while suppressing short-term noise and aligns with the visualizations presented in the Results section.

Collectively, these metrics establish a robust foundation for evaluating how different coordination structures influence agent learning, resource efficiency, and collaborative behavior within AIoMT-enabled healthcare Systems-of-Systems.

Evaluation Process

The evaluation of the proposed MARL framework is conducted through extensive simulations across the four defined Systems-of-Systems (SoS) configurations: Directed, Acknowledged, Collaborative, and Virtual. Each configuration is simulated independently to preserve the structural distinctions in coordination and control, enabling a comparative analysis of emergent system behaviors. The training process spans a total of $N = 1000$ episodes per simulation, during which agents iteratively learn optimal strategies through reinforcement-based interactions with a dynamic healthcare environment.

At the beginning of each simulation, all agents are initialized with random Q-values, and the system state s_0 is sampled from a synthetic distribution representing real-world healthcare inputs, including patient acuity levels, diagnostic uncertainty, and heterogeneous

resource availability. This initialization ensures variability and generalizability across different starting conditions.

During each episode $t \in [0, N]$, agents observe their current state s_t, select actions a_t using the ϵ-greedy policy defined in Equation 7.2, receive corresponding rewards $r_t = R(s_t, a_t)$, and update their Q-values using the learning rule in Equation 7.1. The cumulative reward over time is computed for each agent to assess policy convergence [90]:

$$R^i_{\text{cumulative}} = \sum_{t=0}^{N} r^i_t \tag{7.9}$$

where $R^i_{\text{cumulative}}$ represents the cumulative reward accrued by agent i across the simulation horizon. In parallel, cooperation counts C^i_t, exploitation rates ξ^i_t, and exploration frequencies ϵ^i_t are tracked to characterize agent learning dynamics.

After each episode, both individual-level and system-level performance metrics are recorded. These include smoothed Q-values $\hat{Q}_t$, normalized system efficiency η_t, and aggregated resource utilization U_t, with smoothing applied using a moving average filter:

$$\hat{X}_t = \frac{1}{w} \sum_{k=t-w}^{t} X_k \tag{7.10}$$

where X_k denotes the metric at episode k, and w is the window size used to suppress short-term fluctuations while preserving long-term trends. These filtered metrics are used in the generation of convergence plots and behavioral trend analyses (as shown in the Results section).

For final comparative evaluation, each SoS configuration is also summarized using episode-averaged values of key performance indicators: total cumulative reward R^i_{total}, average Q-value $\bar{Q}^i$, mean cooperation count $\bar{C}^i$, average resource utilization $\bar{U}^i$, and exploration-exploitation rates. These are computed over the final evaluation window $E \subseteq [t_{\text{eval-start}}, t_{\text{end}}]$, with:

$$\bar{Q}^i = \frac{1}{|E|} \sum_{t \in E} Q_t^i \tag{7.11}$$

$$\bar{U}^i = \frac{1}{|E|} \sum_{t \in E} U_t^i \tag{7.12}$$

$$\bar{C}^i = \frac{1}{|E|} \sum_{t \in E} C_t^i \tag{7.13}$$

$$\text{Exploit}_i = \frac{\sum_{t \in E} \mathbb{I}^i_{\text{exploit}}(t)}{|E|} \tag{7.14}$$

where $\mathbb{I}^i_{\text{exploit}}(t)$ is an indicator function equal to 1 if agent i selected an exploitative action at time t. These final metrics are used to construct comparative charts, as shown in the Results section.

The final phase involves a comparative synthesis across Systems-of-Systems (SoS) configurations, focusing on how coordination structures shape the emergent behavior of trained agents. Key aspects of this analysis include cooperation dynamics, the balance between exploitation and exploration strategies, and the overall efficiency of system-level decision-making. This synthesis provides insights into how varying degrees of autonomy, centralization, and communication influence adaptive learning and performance in distributed healthcare environments.

To assess scalability and robustness, additional simulation runs are conducted by varying the number of agents $|A|$ and perturbing environment complexity, such as through stochastic surges in patient demand or sudden reductions in resource capacity. These scenarios allow for stress testing the adaptability of each SoS model under conditions of operational uncertainty.

7.1.5 Results and Analysis

Dataset Overview

The dataset underpinning this study simulates a multi-agent healthcare system by integrating synthetic patient records, facility-level resources, and diagnostic outcomes. This enables robust modeling of decision-making and resource allocation within the proposed Multi-Agent Reinforcement Learning (MARL) framework.

The patient dataset comprises 500 synthetic records, with an average age of 53.9 years and a near-equal gender distribution (51% male, 49% female). Diagnoses span common chronic and acute conditions such as asthma, hypertension, and cardiovascular disease. Care requirements range from routine to emergency, supporting diverse operational scenarios for agent training.

Resource data represent six healthcare facilities—including hospitals and telemedicine units—each initialized with an average of 103 beds and 85 personnel. While not explicitly modeled, real-world constraints such as diagnostic throughput and infrastructure capacity are implicitly captured through episodic state transitions and reward feedback in the environment.

Additionally, 500 diagnostic outcome records are synthesized, reflecting an average diagnostic accuracy of 89.9% when assisted by AI tools. These outcomes model the integration of intelligent diagnostic support into care workflows and inform reward structures within the MARL training loop.

All data are synthetically generated to emulate realistic healthcare interactions while ensuring compliance with ethical standards and privacy constraints. This synthetic dataset enables reproducible experimentation and scalable simulation of healthcare Systems-of-Systems under MARL control.

Environment Setup

The simulation environment is structured to model decision-making processes in a healthcare Systems-of-Systems (SoS) using reinforcement learning, specifically the Q-Learning algorithm. The environment consists of six healthcare agents—hospitals, clinics, and telemedicine platforms—tasked with making binary decisions (admit or reject patients) based on their current states and learned policies.

The simulation spans **1000 episodes**, during which the agents aim to optimize cumulative rewards, representing improved patient outcomes and efficient resource utilization. Each agent operates in a **10-state Markov Decision Process (MDP)**. The states are initialized randomly and updated at each episode to simulate changing conditions such as patient demand, resource availability, and hospital capacity.

Key Q-Learning parameters were defined as follows:

- **Discount factor** ($\gamma = 0.95$): This factor controls how much future rewards influence current decisions, with a high value emphasizing long-term outcomes.
- **Learning rate** ($\alpha = 0.01$): This controls the step size in the Q-value update, ensuring a smooth adjustment of learned policies over time.
- **Exploration rate** ($\epsilon = 0.1$): Using an epsilon-greedy policy, each agent selects a random action with 10% probability (exploration) and chooses the action with the highest Q-value with 90% probability (exploitation), balancing the trade-off between exploring new strategies and exploiting known ones.

Each agent's decision-making is driven by its Q-table, initialized to zero for all states and actions. The Q-table is a **10x2 matrix** for each agent, where rows correspond to the 10 possible states and columns represent the 2 actions (admit or reject). The Q-value for a state-action pair is updated during each episode based on the learning rule as described in Equation 7.1, which incorporates the reward r received for the action, the discount factor γ, and the learning rate α.

Rewards are generated randomly for each agent in the range of **[-10, 10]** to simulate the stochastic nature of patient outcomes and resource management. At the end of each episode, the cumulative reward for each agent is recorded and used to assess overall performance across the simulation.

This setup allows agents to iteratively improve their decision-making policies by adjusting their Q-values based on immediate and future rewards, with the aim of optimizing resource allocation and patient care over time. The following subsection will present the results of the simulation, including cumulative reward trajectories and Q-value evolution for each healthcare agent.

Systems-of-Systems (SoS) Integration and Agent Performance

Systems Integration

In this section, we analyze the comparative efficiency of resource utilization across four Systems-of-Systems (SoS) architectures: Directed, Acknowledged, Collaborative, and Virtual. The plot in Fig. 7.3 illustrates resource consumption trends (beds, staff, and devices) over 1000 episodes for each system.

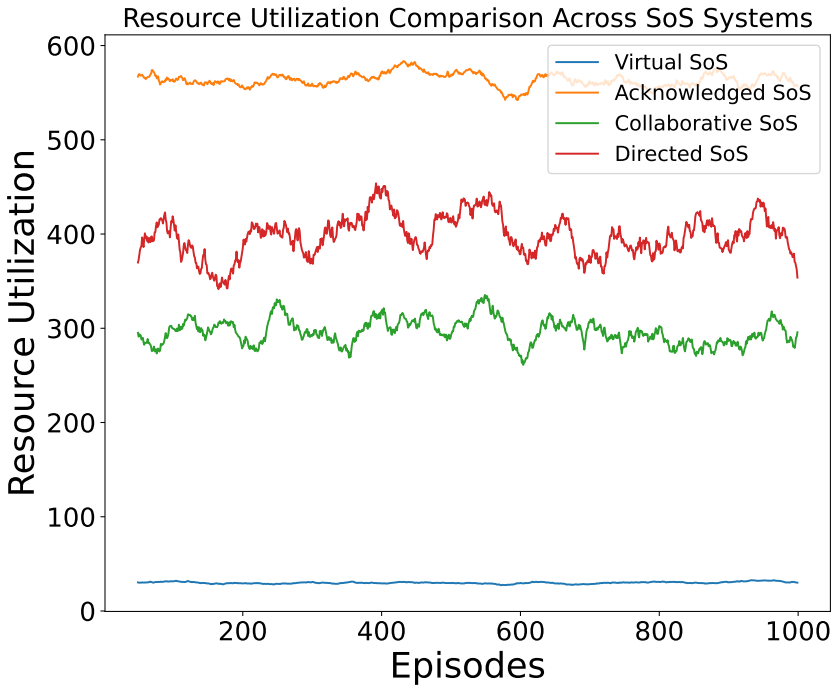

Figure 7.3: Resource utilization trends across four SoS systems over 1000 episodes. The Acknowledged SoS maintains the highest average utilization, followed by Directed and Collaborative systems, while Virtual SoS exhibits minimal physical resource use due to its reliance on telemedicine infrastructure. These patterns reflect each system's efficiency in deploying healthcare resources under varying coordination structures.

The Acknowledged SoS exhibits the highest resource utilization, averaging approximately 580 units, demonstrating superior coordination and resource optimization across decentralized healthcare entities. In contrast, the Directed SoS shows fluctuating utilization between 380 and 450 units, reflecting its centrally governed resource allocation strategy.

The Collaborative SoS maintains resource utilization between 290 and 340 units, suggesting that peer-to-peer cooperation enhances efficiency, though its scalability remains less than that of the Acknowledged system. The Virtual SoS consistently shows the lowest utilization, below 100 units, due to the reduced demand for physical resources in telemedicine environments.

These trends reveal distinct resource management strategies: Acknowledged SoS maximizes resource allocation through decentralized cooperation, while Virtual SoS minimizes physical resource use, tailored to remote care.

Agent Performance and Q-Learning Adaptation This section analyzes the adaptation and learning behavior of agents within the Directed, Acknowledged, Collaborative, and Virtual SoS systems using Q-learning. Key metrics include average Q-values and cumulative rewards, which indicate the agents' performance and decision-making efficiency over time.

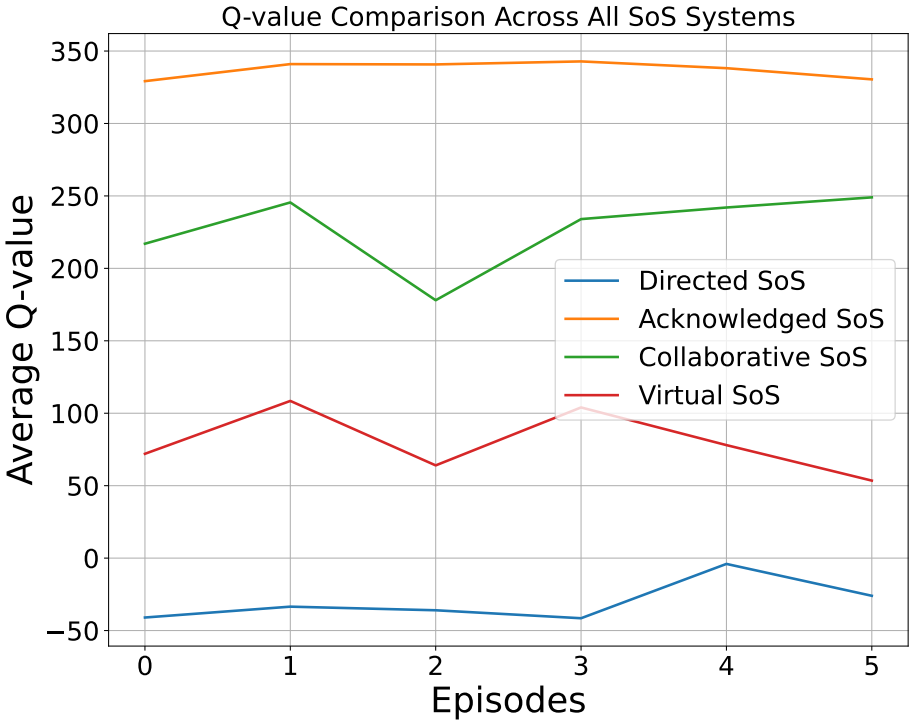

Figure 7.4: Average Q-value comparison across all SoS systems. The Acknowledged SoS system achieves the highest Q-values with minimal variation, indicating effective policy learning. The Collaborative SoS exhibits fluctuating performance with moderate Q-values, while the Virtual SoS maintains stable but lower Q-values. The Directed SoS shows consistently negative Q-values, suggesting limited learning under centralized control.

Figure 7.4 presents the average Q-values across episodes for each SoS system. The Acknowledged SoS maintains the highest Q-values, ranging from approximately 330 to 340,

reflecting stable and effective policy learning. The Collaborative SoS fluctuates between 180 and 240, indicating variable learning performance under decentralized coordination. The Virtual SoS sustains moderate and consistent Q-values between 60 and 110, attributable to limited agent interactions in its loosely coupled architecture. In contrast, the Directed SoS records negative Q-values throughout the episodes, ranging between -50 and -10, which suggests persistent learning difficulties and limited adaptability under a centralized command structure.

Figure 7.5 presents the reward dynamics of the Acknowledged SoS system over 1000 episodes. Total rewards remain high, typically fluctuating between 15 and 25, reflecting consistent overall performance. The average reward curve is stable around 3.5, while the maximum reward curve maintains a steady value slightly below 5. In contrast, the minimum reward curve exhibits more frequent and pronounced variations between 0 and 4. These fluctuations indicate that while the system as a whole performs reliably, individual agents may occasionally encounter suboptimal local conditions, resulting in lower episodic rewards.

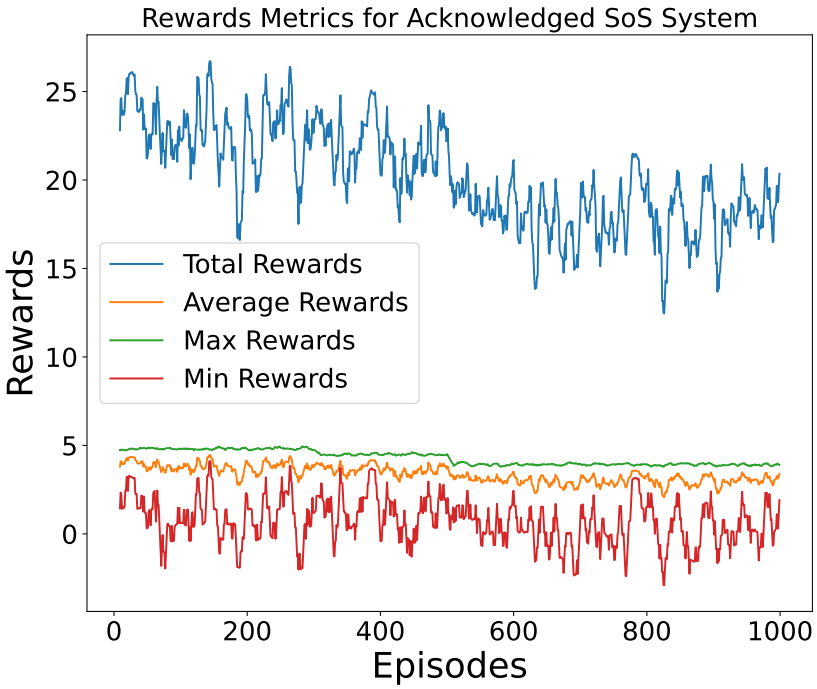

Figure 7.5: Reward metrics for the Acknowledged SoS system. Total, average, and maximum rewards remain consistently high and stable across episodes, indicating effective policy convergence. Fluctuations in minimum rewards suggest intermittent suboptimal behavior by individual agents.

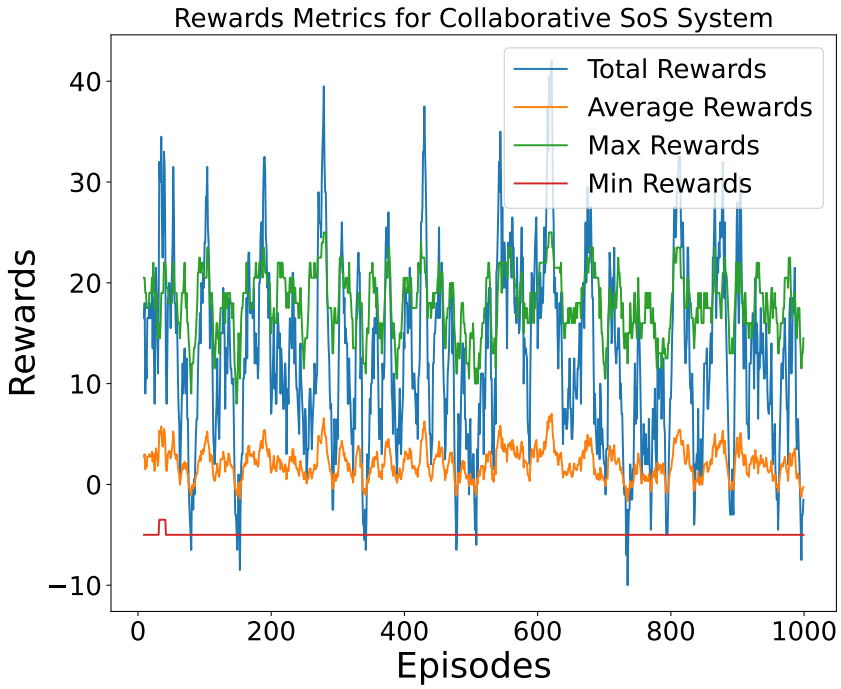

Figure 7.6: Reward metrics for the Collaborative SoS system. High variance in total and maximum rewards reflects the dynamic interactions typical of decentralized decision-making. Occasional sharp drops in minimum rewards highlight local coordination challenges despite stable average performance.

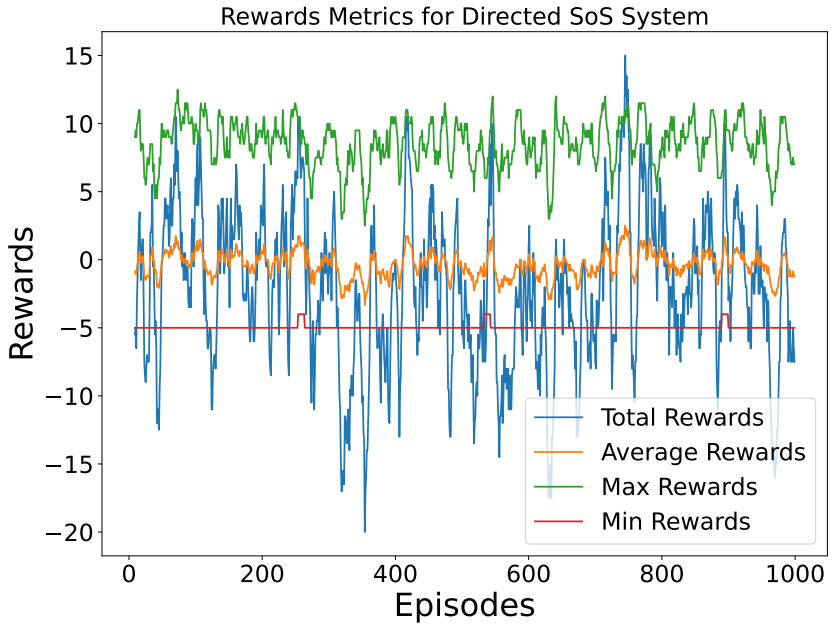

Figure 7.7: Reward metrics for the Directed SoS system. All reward curves show high volatility, with the minimum reward remaining persistently negative. These trends reflect challenges in centralized coordination and the agents' limited adaptability under a static control hierarchy.

Figure 7.6 presents the reward metrics for the Collaborative SoS system. Total and maximum rewards display substantial fluctuation, ranging from 0 to over 40 units, which reflects the dynamic nature of decentralized coordination. Average rewards remain comparatively stable around 5 units, indicating consistent baseline performance. However, minimum rewards often dip sharply below zero, highlighting sporadic suboptimal decisions by individual agents due to transient coordination misalignments.

Figure 7.7 shows the reward metrics for the Directed SoS system. Total rewards vary widely from -15 to $+15$ units, indicating unstable overall system behavior. The average rewards oscillate around 0, suggesting limited policy learning and weak convergence. Maximum rewards peak at approximately 10 units, while minimum rewards consistently remain below zero throughout the episodes, revealing persistent underperformance and inefficiencies stemming from centralized control.

Figure 7.8 illustrates the reward dynamics for the Virtual SoS system. Total rewards fluctuate between approximately 0 and 20 units, reflecting moderately stable yet suboptimal agent interactions. Average rewards remain steady around 2 units, suggesting consistent but limited policy effectiveness. Maximum rewards reach 15 units with frequent variation, while minimum rewards remain persistently negative, often dipping below -5. These trends highlight the challenges of asynchronous, data-driven coordination in achieving robust agent-level performance.

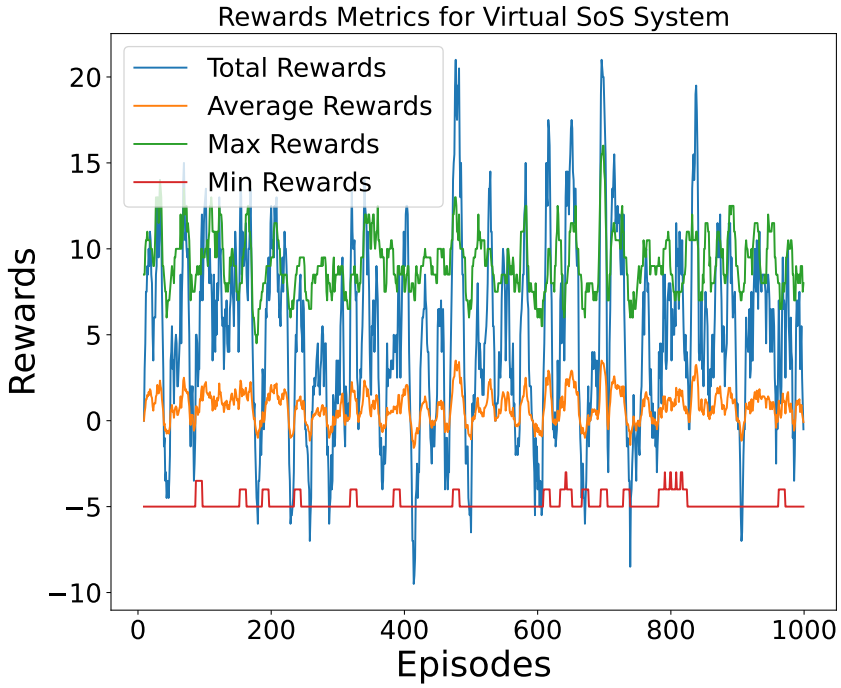

Figure 7.8: Reward metrics for the Virtual SoS system. Moderate fluctuations in total and maximum rewards indicate steady yet constrained performance. Persistent negative minimum rewards highlight limitations in agent adaptability under asynchronous, data-driven coordination.

Each constituent system configuration utilizes a distinct set of six agents tailored to its specific coordination model (as detailed in Tables 7.1 and 7.2). Accordingly, cumulative reward plots illustrate intra-system performance and are not intended for direct cross-agent comparisons between constituent systems.

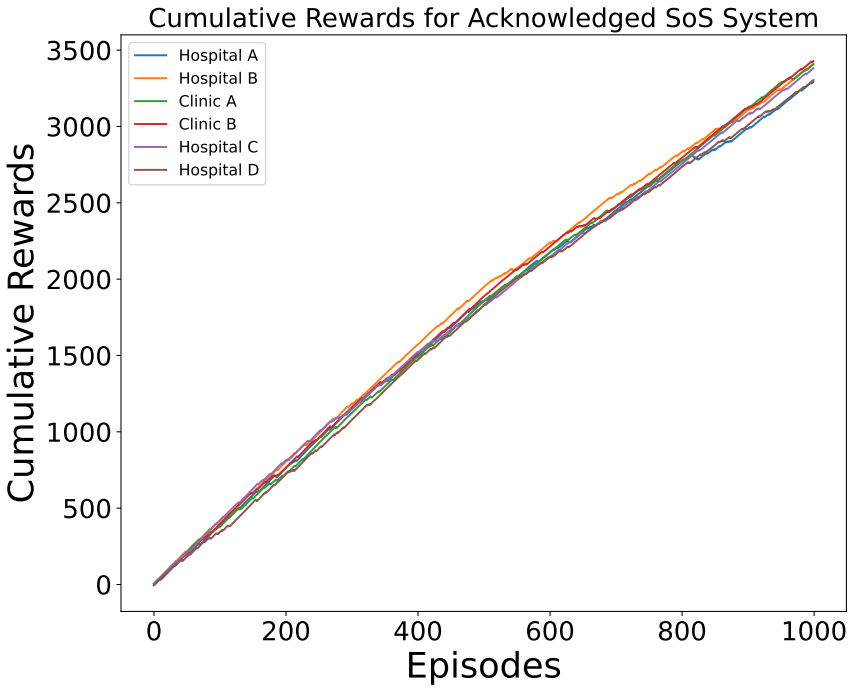

Figure 7.9: Cumulative rewards per agent in the Acknowledged SoS system. All agents demonstrate monotonic reward accumulation, converging near 3500 by episode 1000, indicative of robust semi-centralized coordination and consistent policy effectiveness across diverse agent roles.

Figure 7.9 illustrates the cumulative rewards for each agent in the Acknowledged SoS system. All agents exhibit near-parallel, upward-sloping curves, converging around 3500 by episode 1000. This linear trend indicates effective policy learning and consistent reward accumulation across both physical entities (e.g., hospitals, clinics) and digital platforms. The uniformity across agents highlights the stability provided by semi-centralized coordination.

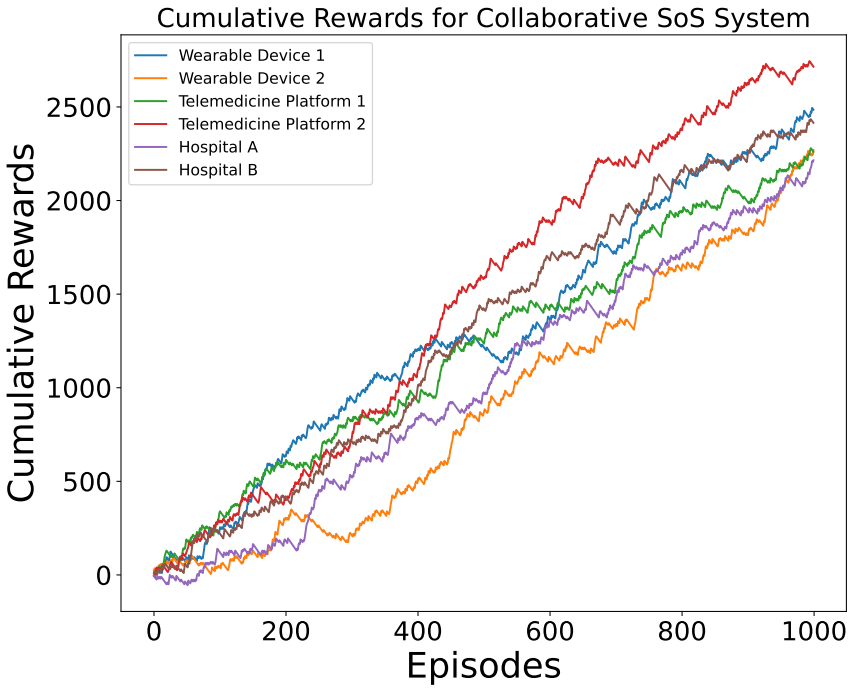

Figure 7.10: Cumulative rewards per agent for the Collaborative SoS. Agents include hospitals, telemedicine platforms, and wearable devices operating under a decentralized coordination scheme. The reward curves demonstrate heterogeneous but generally positive learning behavior.

In Figure 7.10, agents in the Collaborative SoS system show varying slopes in cumulative rewards, with values ranging from 2000 to 2700 by episode 1000. Although all agents improve over time, the differences in trajectory—particularly for telemedicine and wearable agents—suggest uneven reward acquisition, likely due to asynchronous coordination and diverse local observations. Nevertheless, the overall trend reflects effective adaptation within a decentralized architecture.

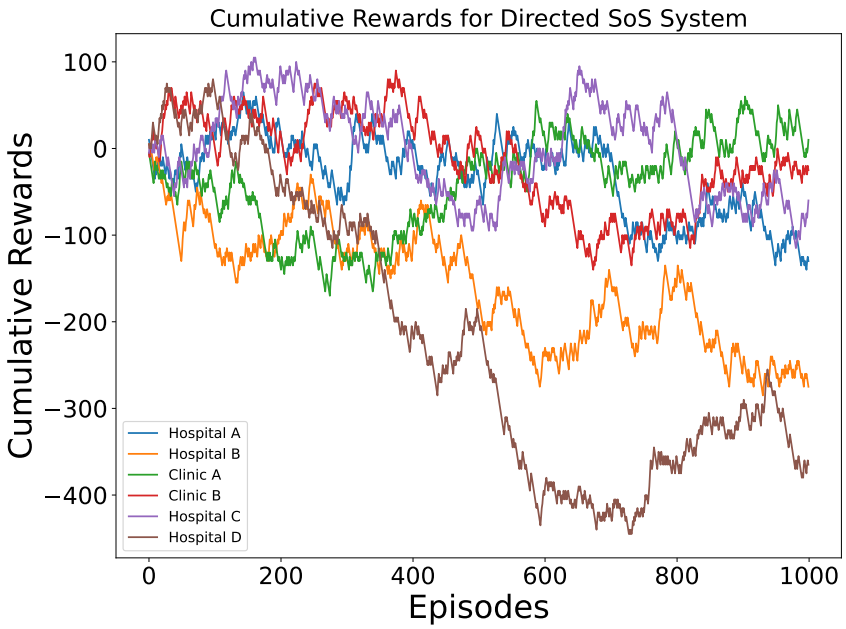

Figure 7.11: Cumulative rewards per agent for the **Directed SoS**. Agents represent hospitals and clinics operating under a centralized authority (see Tables 7.1 and 7.2).

Figure 7.11 presents the cumulative rewards for the Directed SoS system. Unlike the smoother trajectories observed in Acknowledged and Collaborative systems, the Directed SoS exhibits significant divergence among agents. While some hospitals accumulate modest rewards, others—such as Hospital C and Clinic A—exhibit sustained decline, reaching below -400. These disparities suggest that centralized control limits adaptability and hampers local decision quality, leading to persistent underperformance.

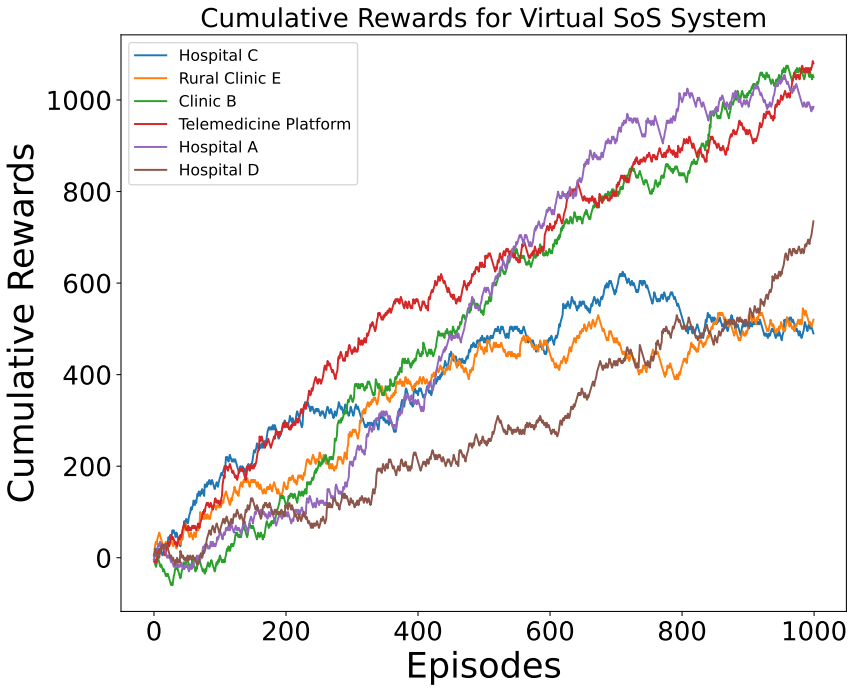

Figure 7.12: Cumulative rewards per agent for the Virtual SoS. Agents include hospitals, clinics, and telemedicine platforms operating asynchronously through data-driven coordination and remote monitoring.

Figure 7.12 illustrates the cumulative rewards for the Virtual SoS system across all agents. Although the final reward values are lower compared to the Acknowledged and Collaborative systems, all agents exhibit steady, upward trajectories, with most converging between 700 and 1100 by episode 1000. This pattern suggests gradual but consistent policy improvement under virtual coordination. The performance variation among agents—especially the lagging rewards of Hospital C and Clinic B—reflects the asynchronous nature of virtual interactions and the reduced control granularity in data-driven healthcare delivery.

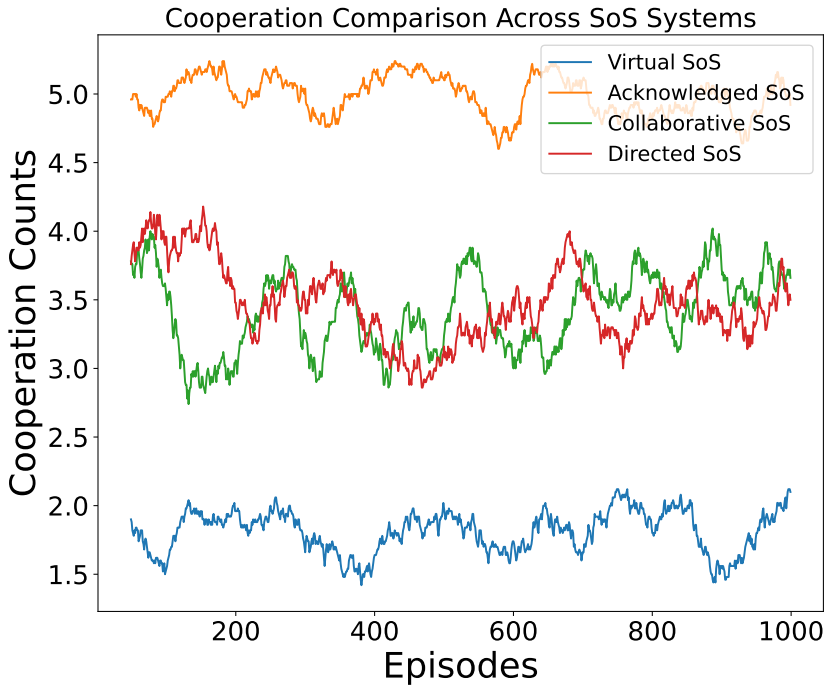

Figure 7.13: Cooperation counts comparison across SoS systems. The Acknowledged SoS consistently demonstrates the highest and most stable cooperation levels, followed by the Collaborative and Directed systems. The Virtual SoS maintains the lowest cooperation levels, attributed to its asynchronous coordination via loosely coupled remote agents. These trends highlight the relationship between system architecture and inter-agent coordination efficiency.

Figure 7.13 presents a comparative analysis of cooperation counts across the four SoS configurations. The Acknowledged SoS system maintains consistently high cooperation levels, averaging close to 5 per episode, indicating effective alignment of semi-autonomous agents under shared policies. The Collaborative and Directed SoS systems fluctuate between 3 and 4, reflecting moderate coordination with occasional disruptions. In contrast, the Virtual SoS system remains below 2 throughout, underscoring limited inter-agent coordination due to its reliance on asynchronously operating digital agents. These observations reinforce the role of structural integration in enabling sustained multi-agent collaboration.

In summary, Acknowledged SoS consistently outperforms other systems in terms of Q-values, rewards, and cooperation, demonstrating superior adaptability and learning. Collaborative SoS performs well, particularly in wearable and telemedicine integration. Virtual

SoS shows moderate but stable performance, while Directed SoS struggles with volatility and lower reward accumulation, highlighting challenges in central decision-making.

System Performance Metrics This section analyzes the overall performance of each SoS system in optimizing resource allocation and agent decision-making based on cumulative rewards and Q-learning behavior.

As demonstrated in the comparison of cumulative rewards (Figure 7.9, Figure 7.10, Figure 7.11, and Figure 7.12), each system exhibited distinct learning dynamics throughout the simulation. The Acknowledged SoS system achieved the highest cumulative rewards, peaking around 3500 by the 1000th episode, indicating superior optimization of resources and agent coordination. The Collaborative SoS system followed closely, with cumulative rewards reaching approximately 2500. This demonstrates a relatively high efficiency in agent collaboration and decision-making, although with slightly more variability.

In contrast, the Virtual SoS system attained a more moderate cumulative reward of around 1000 by the end of the simulation, suggesting that the virtual coordination between telemedicine platforms and hospitals, while functional, was not as efficient as the physical hospital collaboration seen in other systems. The Directed SoS system, on the other hand, displayed significant challenges. Cumulative rewards for some agents, such as Hospital B and Clinic A, fell to negative values below -400, highlighting inefficiencies in centralized decision-making.

The Q-learning trends support these findings. As seen in Figure 7.4, agents in the Directed SoS system consistently struggled to optimize decision-making, with Q-values fluctuating between -50 and 0 throughout the simulation. This indicates persistent challenges in adapting to resource allocation in a centralized control environment. In contrast, the Acknowledged SoS system exhibited strong and stable Q-value growth, reaching approximately 350 early in the simulation and maintaining this high level across episodes. The Collaborative SoS system also demonstrated positive Q-value trends, peaking around 250, though with more fluctuation than the Acknowledged system. Meanwhile, the Virtual SoS system

showed more modest learning, with Q-values improving to around 100 by the end of the simulation.

These results indicate that decentralized systems, particularly the Acknowledged and Collaborative SoS, enabled more efficient learning and adaptation to resource allocation challenges. In contrast, the Directed SoS, constrained by centralized control, struggled to achieve the same level of efficiency and stability.

Exploitation, Exploration, and Learning Behavior Across SoS Systems In multi-agent reinforcement learning, the balance between exploiting known strategies and exploring new ones is crucial for optimal system performance. This balance allows agents to refine their decision-making by either leveraging previously learned policies (*exploitation*) or searching for new, potentially better strategies (*exploration*). The results of this balance are evident in the behavior of agents across different Systems-of-Systems (SoS) configurations.

As shown in Figure 7.4, the **Acknowledged SoS** system consistently outperformed others in terms of Q-value evolution, with agents achieving the highest average Q-values, peaking at approximately 350 by episode 300. This performance is directly tied to the system's high *exploitation counts*, as illustrated in Figure 7.14, where agents maintained exploitation rates above 5.3 throughout the simulation. This indicates that agents in the Acknowledged SoS were highly efficient at leveraging learned policies to optimize resource allocation and decision-making, minimizing the need for exploratory actions.

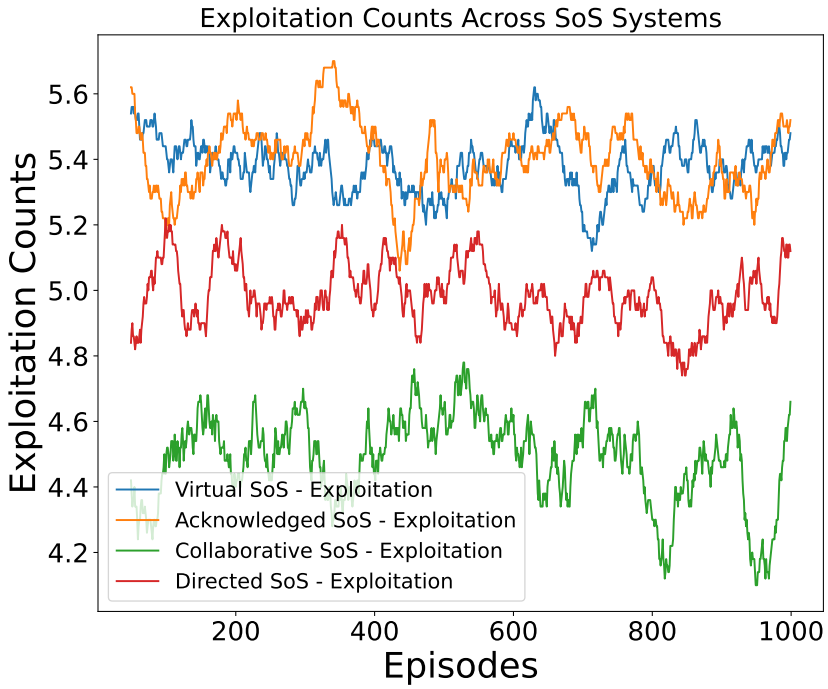

Figure 7.14: Exploitation counts across SoS systems. The Acknowledged and Virtual SoS configurations exhibit consistently higher exploitation frequencies—averaging above 5.4 per episode—indicating effective convergence to stable policies. In contrast, the Collaborative and Directed systems demonstrate lower and more variable exploitation counts, reflecting ongoing adaptation and less consistent policy adherence due to decentralized or rigid coordination structures.

Figure 7.14 compares exploitation counts across the four SoS configurations. The Acknowledged and Virtual SoS systems consistently maintain higher exploitation levels, averaging above 5.4 actions per episode, indicating stable policy convergence and reduced reliance on exploration. By contrast, the Collaborative SoS system exhibits lower exploitation rates, averaging around 4.6, with noticeable variance over time. This trend reflects agents' ongoing need to explore new strategies in response to decentralized, dynamically evolving environments. The Directed SoS system follows a similar pattern of reduced exploitation, suggesting that rigid centralization may limit agents' ability to consistently adhere to optimized policies.

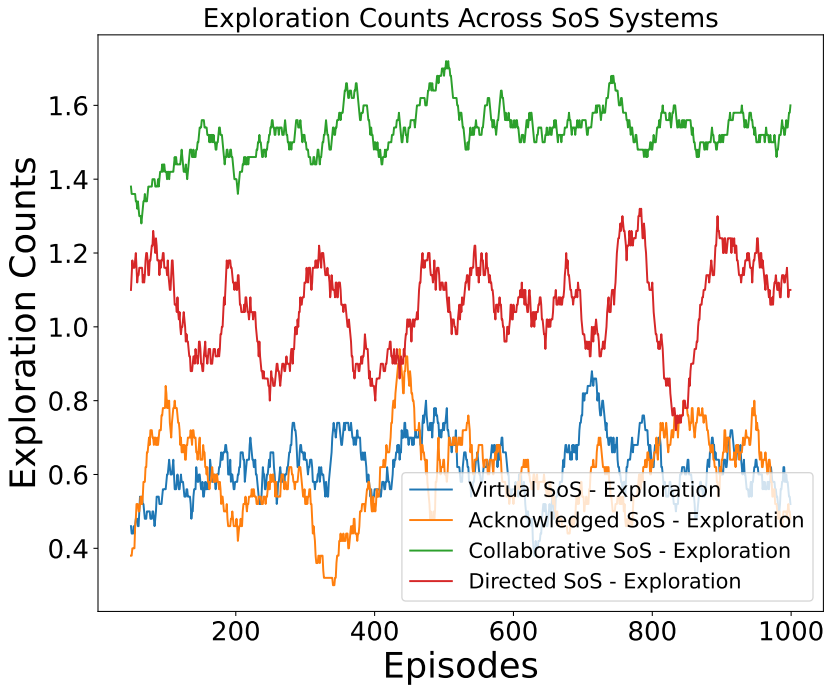

Figure 7.15: Exploration counts across SoS systems. Collaborative SoS agents maintain the highest exploration levels, indicative of ongoing adaptation under decentralized coordination. Lower exploration in Acknowledged and Virtual systems suggests early policy convergence, while Directed SoS shows intermittent exploration consistent with unstable learning.

Figure 7.15 illustrates the exploration behavior across all SoS configurations. The Collaborative SoS system maintains the highest exploration counts throughout training, averaging approximately 1.6 exploratory actions per episode. This persistent exploration reflects the agents' continuous search for improved strategies in a decentralized, high-variability environment. The Acknowledged and Virtual systems exhibit lower and more stable exploration trends, both averaging under 1.0, suggesting that agents in these systems converged more quickly to reliable policies. The Directed SoS configuration demonstrates moderate exploration levels with notable oscillations, indicating unstable policy convergence and challenges associated with rigid centralized control mechanisms.

Finally, the **Directed SoS** system faced the most challenges in balancing exploration and exploitation. As depicted in Figures 7.14 and 7.15, agents in the Directed SoS system

demonstrated lower exploitation rates, combined with sporadic exploration activity, leading to less efficient learning. The centralized control structure hindered agents from effectively exploring new strategies or exploiting known policies, contributing to the system's overall lower performance in terms of Q-value growth and cumulative rewards.

The previously discussed cooperation counts (Figure 7.4) further reinforce these trends. The **Acknowledged SoS** displayed the highest agent coordination, averaging around 5.5, while the **Directed SoS** system showed lower cooperation levels. This reflects the benefits of decentralized systems in promoting inter-agent collaboration and learning, as opposed to the limitations imposed by a centralized decision-making structure in the Directed SoS.

System Efficiency Comparison Across SoS Systems Figure 7.16 presents the normalized system efficiency trends across all SoS configurations over 1000 episodes. The Acknowledged SoS maintains consistently high efficiency, averaging above 0.95, demonstrating effective semi-centralized coordination and rapid adaptation to dynamic resource demands. The Collaborative SoS exhibits moderate performance, fluctuating between 0.80 and 0.90, reflecting occasional coordination lapses under distributed control. The Virtual SoS achieves stable efficiency around 0.85, enabled by simplified coordination through digital interfaces but limited by the absence of physical resource reallocation. The Directed SoS shows persistent volatility and the lowest average efficiency, starting below 0.5 and stabilizing only after prolonged training, indicating inefficiencies associated with centralized decision-making under dynamic workloads.

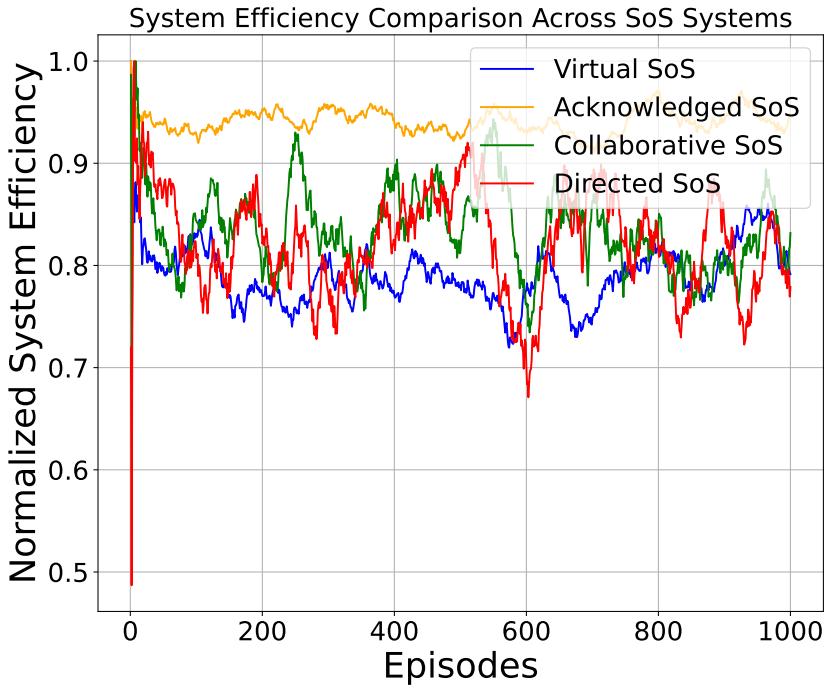

Figure 7.16: Smoothed comparison of normalized system efficiency across SoS systems. Acknowledged SoS exhibits the highest and most stable efficiency, highlighting the benefits of semi-centralized learning. Collaborative and Virtual systems perform moderately, with periodic fluctuations. Directed SoS demonstrates delayed convergence and volatility due to limitations of centralized control.

The Acknowledged SoS consistently outperforms others, sustaining an efficiency level near 0.97. This indicates that agents operating under decentralized but coordinated policies adapt more effectively to evolving system demands.

The Collaborative SoS shows moderate variability, with efficiency oscillating between 0.80 and 0.90. While agents coordinate well across heterogeneous platforms, including wearables and telehealth, sporadic inefficiencies arise due to distributed learning.

The Virtual SoS remains relatively stable around 0.85, benefiting from the simplicity of remote interaction models. However, its lack of physical coordination among member systems constrains its overall optimization potential.

In contrast, the Directed SoS begins with poor efficiency (below 0.5), improving only after extended training. The delayed convergence reflects challenges in centralized policy learning and slower adaptation to dynamic healthcare loads.

To synthesize these findings, Table 7.4 reports final metrics aggregated across the last 10 episodes, including average Q-values, resource utilization, agent cooperation, and exploration-exploitation behaviors. These metrics reflect the learned behavior of constituent member systems operating under each SoS architecture.

Table 7.4: Learned Performance Metrics of Member Systems under Each Constituent System

Constituent System	**Q-val**	**Util.**	**Coop.**	**Exploit.**	**Explore.**
Directed	-36.92	381.37	4.00	5.08	0.88
Acknowledged	337.07	564.98	4.90	5.38	0.62
Collaborative	210.00	295.72	3.66	4.66	1.60
Virtual	80.17	30.08	2.10	5.48	0.52

Overall, the findings reaffirm the strategic benefits of decentralized coordination. Member systems operating under the Acknowledged and Collaborative constituent systems exhibit superior learning outcomes and resource synergy. In contrast, Directed constituent systems demonstrate delayed adaptation, highlighting the importance of local autonomy and distributed intelligence in MARL-enabled healthcare System-of-Systems architectures.

7.1.6 Discussion

This section discusses the implications of the results, compares our findings with prior work, identifies areas for future improvements, and addresses the key challenges in current research highlighted in earlier sections.

Interpretation of the Results

The performance of agents, as indicated by Q-value trends and cumulative rewards, demonstrates the effectiveness of decentralized decision-making frameworks in optimizing resource allocation across the SoS systems. Notably, the Acknowledged SoS system consistently achieved superior outcomes, with agents demonstrating a high level of cooperation (Figure 7.13), Q-values reaching 350 (Figure 7.4), and cumulative rewards nearing 3500 (Figure 7.9). These results indicate the robustness of a decentralized yet coordinated system, where agents independently manage resources while adhering to shared goals, leading to optimized performance.

In contrast, the Directed SoS system faced significant challenges, as reflected in its fluctuating Q-values and negative cumulative rewards for certain agents (Figure 7.11). Centralized control in a dynamic healthcare environment resulted in suboptimal resource allocation, limiting the system's ability to adapt to real-time changes in patient demand and resource availability.

Implications of Findings

The simulation results underscore the suitability of decentralized Systems-of-Systems (SoS) architectures—particularly the Acknowledged and Collaborative models—for AIoMT-enabled healthcare environments, where adaptability, autonomy, and efficient resource management are essential. In these systems, agents demonstrated the ability to learn and coordinate effectively, as reflected in high cooperation counts and stable exploitation behaviors (Figure 7.14). These behavioral trends suggest that decentralized agents can formulate locally optimal policies while still aligning with broader system objectives, contributing to improved system-wide outcomes. Such behavior is consistent with the principles of distributed decision-making frameworks previously advocated in healthcare system studies [115, 116].

Moreover, the performance of the Virtual SoS system, which maintained steady cumulative rewards despite minimal resource utilization (Figure 7.3, Figure 7.12), illustrates the potential of telemedicine and remote monitoring platforms to support sustainable health-

care delivery. These results imply that Virtual SoS configurations can be particularly advantageous in under-resourced or geographically isolated regions, where reliance on physical infrastructure is limited. The ability to sustain performance in such scenarios suggests a promising role for Virtual SoS models in enhancing healthcare access and resilience in constrained settings.

Addressing Key Challenges in Current Research

Systems-of-Systems (SoS) Integration A key contribution of this research lies in addressing the long-standing challenge of integrating heterogeneous healthcare systems into a cohesive operational framework. The proposed MARL-based architecture facilitates coordination among distributed entities—such as hospitals, wearable devices, and telemedicine platforms—without requiring a monolithic control mechanism. Simulation results illustrate that Acknowledged and Collaborative SoS systems, both of which preserve local agent autonomy while fostering inter-agent coordination, significantly enhance resource sharing, diagnostic accuracy, and overall system performance. These benefits are particularly evident in the high cumulative rewards and cooperation counts observed in Figures 7.9 and 7.13. Compared to prior studies that focused on isolated system optimization [117,118], this work demonstrates the operational advantages of integrated, learning-based coordination strategies.

Hyperparameter Tuning and Reward Structure Design The performance of the Directed SoS system and its counterparts was sensitive to the tuning of key hyperparameters—including the learning rate α, discount factor γ, and exploration rate ϵ—each of which governs the agent's ability to adapt and converge within a dynamic environment. In the Directed SoS case, a fixed exploration rate of $\epsilon = 0.1$ encouraged agents to occasionally explore novel strategies, such as adjusting admission thresholds or reallocating personnel. This was complemented by a 90% exploitation policy to retain learned behavior. As illustrated in Figure 7.15, Collaborative SoS maintained a higher exploration rate, averaging around

1.6 exploratory actions per episode, whereas Acknowledged and Directed systems exhibited lower exploration, favoring exploitative stability.

The reward function was also crafted to reflect key healthcare objectives. Agents received higher rewards for actions that aligned with operational goals—for instance, admitting patients with linked wearable telemetry or rebalancing limited staff. The Q-value evolution in Figure 7.4 confirms this alignment, where the Acknowledged SoS outperformed others with smooth convergence toward high-value policies. Meanwhile, Directed SoS agents struggled, accumulating negative or unstable cumulative rewards as shown in Figure 7.11.

Scalability and Agent Adaptation The MARL framework was designed to be inherently scalable across a diverse set of agent types and SoS configurations. Simulations confirmed that decentralized architectures, such as Acknowledged and Collaborative SoS, scaled effectively, allowing multiple agents to independently learn optimal policies for resource management under heterogeneous constraints. While the Directed SoS system did allow for centralized policy refinement, its agents exhibited limited adaptability and higher performance volatility, as evidenced by both fluctuating Q-values and lower cumulative rewards (Figure 7.4).

In contrast, decentralized agents in Acknowledged and Collaborative systems consistently demonstrated more efficient adaptation to stochastic environmental changes, enabling them to maintain higher reward trajectories. These results affirm the framework's robustness and scalability, particularly in environments characterized by complex, distributed decision-making and real-time demands.

Comparisons with Prior Work

Our results align with existing literature that advocates for decentralized, collaborative systems in healthcare, particularly in managing complex resource allocation. Prior research on SoS integration focused on enhancing interoperability without fully leveraging MARL frameworks [117, 118]. By incorporating MARL into SoS architectures, this study offers a

novel approach to optimizing resource use and patient outcomes, a gap previously underexplored [19].

Our findings also contrast with centralized models, like the Directed SoS, which showed inefficiencies, particularly in dynamic environments [116]. This suggests future research should enhance decentralized decision-making to handle healthcare uncertainties better.

Limitations and Potential Improvements

Although the proposed MARL framework demonstrates strong performance, particularly in decentralized Systems-of-Systems (SoS) configurations, several limitations should be acknowledged. The use of synthetically generated data—while essential for maintaining ethical standards and modeling diverse scenarios—may not fully capture the complexity, noise, and unexpected correlations found in real-world healthcare operations. Future studies should validate the framework using empirical datasets, ideally drawn from hospital networks, telemedicine logs, or wearable device telemetry, to assess generalizability and robustness under actual clinical conditions.

One modeling limitation arises from agent variability across constituent systems. Although each simulation includes six agents, their identities and functional roles are adapted to reflect the coordination logic specific to the Directed, Acknowledged, Collaborative, and Virtual configurations. This design choice enhances architectural fidelity but limits direct cross-agent comparisons across constituent systems. Future extensions may consider normalizing agent roles or introducing abstraction layers to support more generalized and comparative evaluations across heterogeneous coordination models.

Furthermore, while decentralized agents proved effective in optimizing resource allocation, they inherently rely on communication to coordinate decisions. In large-scale or bandwidth-constrained settings, the communication overhead can introduce latency, affecting real-time responsiveness. This issue becomes particularly relevant for edge environments where infrastructure is limited. To mitigate this, future research could explore lightweight communi-

cation strategies or incorporate federated reinforcement learning to reduce inter-agent data exchange without compromising learning efficacy.

Lastly, the performance limitations observed in the Directed SoS system underscore the need for more advanced centralized control models that can dynamically adapt to changing system states. Centralized policies, when static or slow to update, are prone to bottlenecks and inefficiencies. Enhancing these systems with predictive modeling—such as AI-driven demand forecasting or context-aware scheduling—could significantly improve responsiveness and reduce resource misallocation. Integrating hybrid architectures that blend centralized guidance with local autonomy may also offer a promising avenue for future exploration.

7.1.7 Chapter Summary

This case study introduced a modular System-of-Systems (SoS) architecture integrated with Multi-Agent Reinforcement Learning (MARL) to enhance coordination across distributed healthcare entities in AIoMT-enabled environments. By simulating four distinct SoS configurations—Directed, Acknowledged, Collaborative, and Virtual—the study provided a comparative evaluation of how coordination models impact learning performance, agent cooperation, and system-level efficiency.

Key findings revealed that decentralized architectures, particularly Acknowledged and Collaborative systems, enabled superior policy convergence, cumulative rewards, and adaptive behavior across heterogeneous agents. Conversely, centralized control in the Directed system led to slower learning, reduced cooperation, and suboptimal decision-making. The framework demonstrated strong scalability and adaptability under dynamic conditions, validating the role of MARL in achieving real-time coordination across hospitals, telemedicine platforms, and wearable systems.

These results reinforce the broader dissertation theme of using ML-driven SoS frameworks to optimize decision-making in complex, resource-constrained environments. Future direc-

tions include enhancing communication efficiency, integrating empirical healthcare datasets, and exploring hybrid control architectures for real-world deployment.

Having explored the role of MARL in healthcare SoS, the next **Chapter 8** shifts focus to smart city optimization. It examines how MARL agents operating across diverse domains—including energy, transportation, safety, and communication—can coordinate emergent behavior and improve system-wide responsiveness in complex urban environments.

Chapter 8

Optimization of Smart Cities Using MARL

8.1 Smart City Infrastructure Optimization via Multi-Agent Reinforcement Learning

8.1.1 Problem Overview and Context

Smart cities represent a highly complex and dynamic class of cyber-physical systems where constituent subsystems—such as transportation, energy, public safety, and communication—must operate independently while contributing to a unified set of urban objectives. This inherent interdependence positions smart cities as prototypical examples of a System of Systems (SoS), characterized by operational and managerial independence, geographical distribution, and emergent behaviors. While traditional Systems Engineering (SE) methodologies such as the Vee model and Model-Based Systems Engineering (MBSE) provide foundational tools for system design and lifecycle management, they are often limited in their ability to handle the real-time, decentralized, and adaptive requirements of smart city environments.

The limitations of conventional SoS approaches are particularly evident in their inability to accommodate non-linear interactions, dynamic reconfiguration, and decentralized decision-making among heterogeneous entities. Most existing frameworks assume predefined hierarchies and static relationships, which restrict responsiveness to emergent behaviors and evolving system demands. Meanwhile, reinforcement learning (RL), and in particular Multi-Agent Reinforcement Learning (MARL), has emerged as a promising alternative for optimizing the performance of individual systems within smart cities—such as adaptive traffic control or distributed energy management. However, these applications typically operate in

silos and fail to capture the inter-agent dependencies and global objectives that define SoS performance.

To bridge this gap, this case study proposes a novel hybrid framework that integrates MARL with structured SoS methodologies. The constituent systems are modeled as agents operating under various coordination typologies (e.g., Directed, Acknowledged), and MARL is used to train policies that align local decisions with overarching SoS goals. This integration leverages the traceability of SE and the adaptability of learning-based agents to address real-time coordination, emergent behavior, and policy optimization across domains. The framework is evaluated in a simulated smart city testbed, assessing metrics such as system efficiency, adaptability, and inter-agent cooperation. By positioning smart city management as a dynamic SoS optimization challenge, this work contributes to both the theory and practice of AI-enabled urban systems engineering.

8.1.2 Research Objectives and Questions

The primary objective of this case study is to develop and evaluate a hybrid decision-making framework that integrates Multi-Agent Reinforcement Learning (MARL) with traditional Systems Engineering (SE) methodologies to optimize coordination and adaptability within smart city Systems of Systems (SoS). The framework aims to dynamically manage heterogeneous urban subsystems—including transportation, energy, safety, and communication—while aligning their behaviors with global city-wide goals.

Specifically, this case study addresses the following research objectives:

- **Objective 1:** To model smart city infrastructure as a System of Systems (SoS) composed of autonomous agents representing constituent subsystems operating under various coordination models.
- **Objective 2:** To design and implement MARL-based agents capable of decentralized policy learning that balances local autonomy with global SoS efficiency.

- **Objective 3:** To evaluate the effectiveness of augmented reward mechanisms in fostering inter-agent cooperation, improving system responsiveness, and minimizing emergent conflicts.
- **Objective 4:** To assess the scalability, adaptability, and safety of the proposed framework in dynamically evolving smart city environments through simulation-based performance metrics.

Based on these objectives, the following research questions guide the investigation:

- **RQ1:** How does the integration of MARL with traditional SE methodologies improve coordination across smart city constituent systems in a System of Systems (SoS) context?
- **RQ2:** What impact do different coordination models (e.g., Directed, Acknowledged, Collaborative) have on system-level performance, adaptability, and cooperation among agents?
- **RQ3:** Can augmented reward mechanisms align agent-level learning with global objectives in a dynamic and decentralized environment?
- **RQ4:** To what extent can the proposed framework scale to support complex, multi-domain urban systems without compromising safety or operational stability?

8.1.3 Methodology

SoS Member System Characterization

The proposed framework categorizes the smart city's System of Systems (SoS) member systems based on their control structures and interaction paradigms. These systems are classified into four types: **Acknowledged**, **Directed**, **Collaborative**, and **Virtual**, each embodying unique roles within the smart city's agent-based decision-making framework. This classification bridges centralized and decentralized control, ensuring flexibility and adaptability in addressing dynamic system-wide objectives.

Acknowledged Systems represent decentralized systems that operate independently but rely on shared interfaces for coordination and information exchange. The *Transportation Agent*, for example, autonomously manages traffic flows, public transit, and EV charging schedules. Despite its independent operation, it collaborates with the *Energy Agent* to ensure charging infrastructure aligns with grid constraints and availability.

Directed Systems are characterized by centralized control structures where decision-making is coordinated through a central authority to achieve specific objectives. The *Public Safety Agent* exemplifies this type, directing emergency responses and resource allocation during critical events. By interacting with decentralized agents, such as Transportation and Energy, it ensures seamless route clearance and uninterrupted power supply for emergency operations.

Collaborative Systems operate through active cooperation and shared goals among agents. The *Energy Agent* demonstrates this approach by balancing energy supply-demand dynamics, integrating renewable energy sources, and coordinating with the *Transportation Agent* to optimize EV charging during peak and off-peak hours. This collaboration enhances system resilience and sustainability.

Virtual Systems emphasize data-driven, indirect collaboration without physical integration. The *Communication Agent*, as a Virtual System, facilitates real-time information exchange across the SoS, enabling other agents to make informed decisions. By managing data flow, it supports cross-agent communication on traffic conditions, energy demands, and emergency updates, fostering system-wide alignment.

Table 8.1: Characterization of SoS Member Systems in the Proposed Framework

System Type	Agent Example	Characteristics	Functionality
Acknowledged	Transportation Agent	Decentralized Operation	Traffic management, EV charging coordination
Directed	Public Safety Agent	Centralized Control	Emergency response, route clearance
Collaborative	Energy Agent	Active Cooperation	Renewable integration, demand balancing
Virtual	Communication Agent	Data-Driven Collaboration	Real-time updates, cross-agent communication

While traditional SoS classification (e.g., Acknowledged, Directed) applies to the overall system, we reinterpret these categories to characterize the operational behavior of each constituent agent. This allows the MARL framework to account for varying degrees of autonomy, control, and coordination among agents, reflective of their functional roles. For example, a 'Directed' agent like Public Safety receives policy updates centrally and may use a higher global reward weight, whereas an 'Acknowledged' or 'Collaborative' agent like Transportation operates with more local autonomy and decentralized decision-making, balancing local and global rewards dynamically. **Examples of Local and Global Goals:** To illustrate how constituent agents operate within the proposed framework, the Transportation Agent aims to minimize congestion and reduce EV routing delays. The Energy Agent focuses on balancing real-time supply-demand conditions, optimizing charging schedules, and flattening peak loads. The Public Safety Agent prioritizes minimizing emergency response time and coordinating with other agents to ensure route availability. The Communication Agent supports all agents by maintaining low latency and minimizing packet loss, ensuring effective

inter-agent synchronization. These local goals are aligned with overarching SoS objectives, such as improving overall efficiency, adaptability, and coordination effectiveness.

This classification enables the proposed framework to accommodate the diverse operational requirements and control structures of the SoS member systems. By modeling these systems as agents within a unified framework, the methodology facilitates the independent optimization of local objectives while ensuring alignment with overarching SoS goals. Centralized agents, such as the Public Safety Agent, provide directive control during critical events, whereas decentralized agents, like the Transportation and Energy Agents, exhibit adaptability and independent decision-making capabilities. The Communication Agent, functioning as a Virtual System, facilitates seamless information exchange and coordination, effectively bridging the centralized and decentralized components to achieve global system objectives.

Clarification on Classification Usage: While the terms "Acknowledged," "Directed," "Collaborative," and "Virtual" are traditionally used to describe whole SoS governance models, this framework reinterprets them at the constituent system level to reflect the operational role and control modality of each agent. This reinterpretation aligns with agent-based design, where each agent may exhibit different structural properties or control dependencies based on its subsystem function.

Impact on Agent Modeling and Reward Design: These classifications influence agent design in terms of reward weighting, observability, and communication. For instance, a "Directed" agent like Public Safety operates under high-priority centralized constraints and receives tighter global reward coupling. In contrast, an "Acknowledged" agent like Transportation emphasizes decentralized optimization but shares select state variables to coordinate with peers. These distinctions help tailor the reward structure and coordination mechanisms according to system roles.

In addition to distinguishing the operational and structural paradigms of SoS member systems, this classification establishes a foundation for modeling their interactions as

autonomous agents. By incorporating both centralized and decentralized decision-making mechanisms, the framework ensures local adaptability while aligning individual agent actions with the overall goals of the SoS. This dual-layered approach balances flexibility with coordination, making it well-suited for managing the dynamic and interdependent nature of smart city systems.

Agent-Based Modeling and Decision Framework

Building on the characterization of SoS member systems, the agent-based modeling framework leverages reinforcement learning (RL) techniques to enable dynamic, context-aware decision-making. Each agent is equipped to optimize local objectives, such as resource allocation and traffic control, while ensuring its actions contribute to the broader goals of the smart city.

To complement the local agent view, Fig. 8.1 illustrates the complete system-level process flow. This end-to-end diagram captures how agents interact with dynamic environmental inputs, execute policy decisions using PPO, receive both local and global rewards, and participate in SoS-wide coordination through feedback signals. It highlights the integration of decentralized policy execution and centralized reward aggregation within the proposed MARL-based SoS architecture.

This study introduces a Multi-Agent Machine Learning (MAML) framework as the core methodology for decision-making and control in System of Systems (SoS). The framework integrates reinforcement learning (RL), advanced neural network architectures, and decentralized coordination mechanisms to address the dual challenge of optimizing localized system behaviors and achieving overarching SoS objectives. Member systems—including transportation, energy, waste management, and public safety—are modeled as autonomous agents that adapt to evolving conditions and collaborate effectively to enhance overall system performance..

An agent is an autonomous decision-making entity that interacts with its environment, observes its state, selects an action, and receives feedback in the form of a reward. The

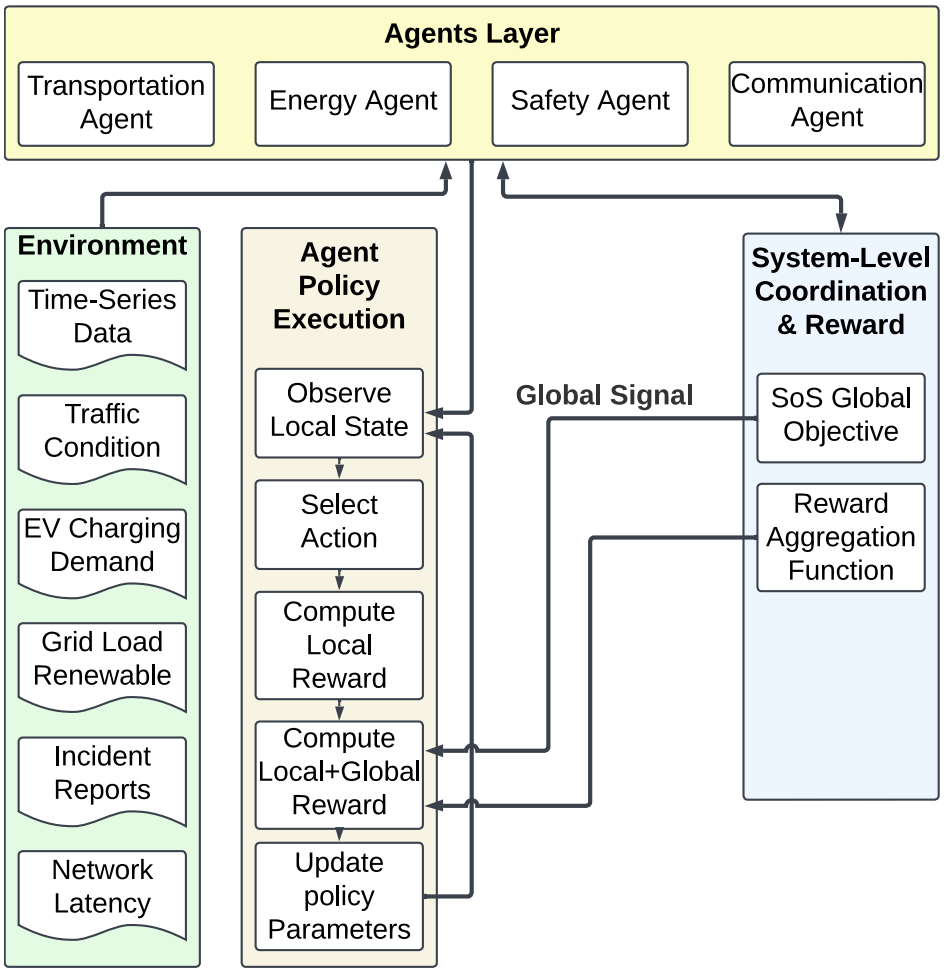

Figure 8.1: System Process Flow Diagram: This figure illustrates the operational cycle of the proposed MARL-based System of Systems (SoS) coordination framework. Agents (e.g., transportation, energy, safety, communication) observe their environments, execute local policies using Proximal Policy Optimization (PPO), and receive both local and globally-aggregated rewards. SoS-level coordination is achieved via a reward aggregation mechanism driven by a shared global objective, enabling decentralized yet synchronized optimization across all agents.

environment constitutes the dynamic external system, which includes factors such as traffic patterns, energy consumption levels, waste accumulation trends, and emergency response conditions. For an agent i, where $i \in \{1, 2, \ldots, N\}$ and N represents the total number of agents, the state at time t is denoted as $s_i^t \in \mathcal{S}_i$, where $\mathcal{S}_i$ represents the state space of agent i. The global state of the SoS at time t, expressed as $s^t = \{s_1^t, s_2^t, \ldots, s_N^t\}$, is the aggregation of the states of all agents within the system.

Fig. 8.2 illustrates the decision-making process of an individual agent within the proposed framework. Each agent interacts with its local environment by observing its current state (s_i^t), selecting actions (a_i^t), and receiving immediate rewards ($r_i^{\text{local}}, r_i^{\text{global}}$). The feedback loop ensures adaptability through policy evaluation and updates (θ_i). This iterative process enables the agent to align its local objectives with the overarching goals of the System of Systems (SoS). Additionally, when global coordination is required, agents share state information to facilitate collaborative decision-making across the SoS.

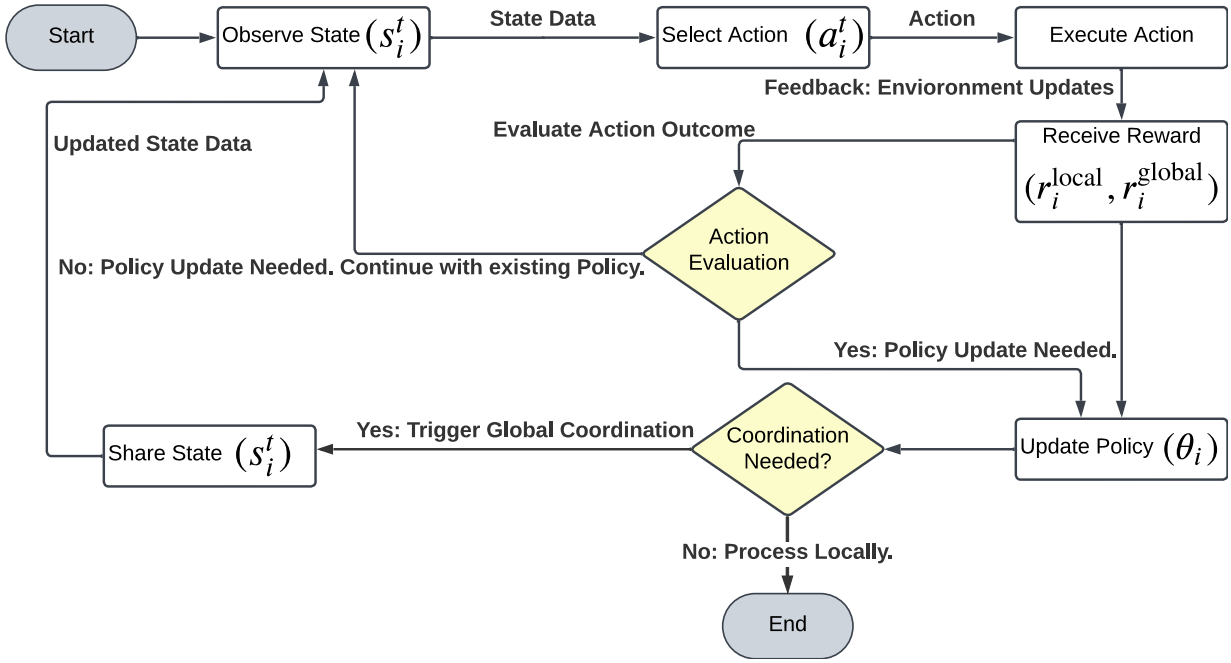

Figure 8.2: Local Agent Decision-Making Framework. This figure illustrates the internal decision cycle of an autonomous agent within the proposed SoS architecture. Each agent observes its local state s_i^t, selects and executes an action a_i^t, and receives immediate feedback in the form of local and global rewards ($r_i^{\text{local}}, r_i^{\text{global}}$). Based on outcome evaluation, the agent decides whether to update its policy parameters θ_i. If coordination is required, the agent shares its state s_i^t with the global coordination layer. This framework enables decentralized yet globally-aligned behavior across all agents through adaptive policy feedback, environmental responsiveness, and event-triggered coordination.

Actions taken by an agent i are represented by $a_i^t \in \mathcal{A}_i$, where $t \in \mathbb{N}$ denotes the discrete time step, and $\mathcal{A}_i$ defines the action space of agent i. Each action a_i^t corresponds to a specific decision or control input executed by the agent within its local environment. The global state

$s^t = \{s_1^t, s_2^t, \ldots, s_N^t\}$ represents the combined states of all N agents in the SoS, encapsulating the overall dynamics of the system.

The policy of an agent, expressed as $\pi_i(a_i|s_i;\theta_i)$, defines the probability of selecting an action $a_i \in \mathcal{A}_i$ given the current state $s_i \in \mathcal{S}_i$. This policy is parameterized by θ_i, which denotes the neural network weights used to approximate the mapping between states and actions. By employing this probabilistic mapping, the agent explores different action strategies and adapts its decisions to the dynamic environment.

The objective of each agent is to optimize its policy π_i to maximize its expected cumulative reward over time. This reward function reflects the agent's ability to achieve local objectives, such as reducing energy consumption or enhancing traffic flow, while also contributing to the global goals of the SoS. Formally, the agent seeks to learn an optimal policy π_i^* that satisfies [119]:

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}_{\pi_i}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_i^t, a_i^t, s_{-i}^t)\right] \tag{8.1}$$

where $\gamma \in [0,1]$ is the discount factor that balances the importance of immediate versus future rewards, and $r_i(s_i^t, a_i^t, s_{-i}^t)$ is the reward function. This reward depends on the agent's state s_i^t, its action a_i^t, and the states of other agents $s_{-i}^t = \{s_j^t\}_{j\neq i}$. As shown in (8.1), the optimal policy π_i^* is derived by maximizing the expected cumulative reward for agent i.

The global objective of the SoS is to maximize the cumulative reward R, which is expressed as a weighted sum of the individual agent rewards [3, 120]:

$$R = \sum_{i=1}^{N} w_i R_i, \quad \text{where } R_i = \sum_{t=0}^{\infty} \gamma^t r_i(s_i^t, a_i^t) \tag{8.2}$$

where w_i represents the weight assigned to agent i, indicating its relative importance in achieving global SoS objectives. The cumulative reward R_i for agent i is computed as the discounted sum of immediate rewards over time, with γ representing the discount factor. The immediate reward $r_i(s_i^t, a_i^t)$ captures the impact of the agent's action $a_i^t \in \mathcal{A}_i$ in its

local state $s_i^t \in \mathcal{S}_i$. The global reward R, as defined in (8.2), represents a weighted sum of individual agent rewards, emphasizing the importance of each agent's contribution to the SoS objectives.

The block diagram below in Fig. 8.3 illustrates the interaction between the System of Systems (SoS) global objective and its constituent agents. Each agent—representing a distinct subsystem such as transportation, energy, safety, or communication—operates within its local environment, characterized by state s_i^t, action a_i^t, and local reward $r_i^{\text{local}}(s_i^t, a_i^t)$. The agents contribute to the global objective by sharing state updates and policy adjustments through real-time communication and coordination feedback loops. The cumulative reward R, a weighted sum of individual rewards, aligns local actions with overarching SoS goals, ensuring system-wide efficiency, adaptability, and collaboration. This framework demonstrates the dynamic integration of local decision-making with global optimization in a smart city environment.

To provide a clear operational understanding of the proposed framework, Fig. 8.3 presents the Concept of Operations (CONOPS) for the smart city System of Systems (SoS). Each agent—representing a constituent system such as transportation, energy, safety, or communication—interacts with its environment through local states, actions, and rewards. These agents engage in real-time coordination with the SoS-level global objective through state updates and policy feedback loops. The global reward function aggregates local contributions, enabling system-wide optimization while preserving agent autonomy. This architecture allows the system to dynamically respond to changing conditions and emergent behaviors in a decentralized yet coordinated manner.

For instance, when EV demand surges during peak hours, the Transportation Agent reroutes traffic while the Energy Agent prioritizes grid capacity for charging. Simultaneously, the Communication Agent ensures real-time data exchange, and the Safety Agent clears routes for emergency scenarios—all under the guidance of a shared global objective. This is

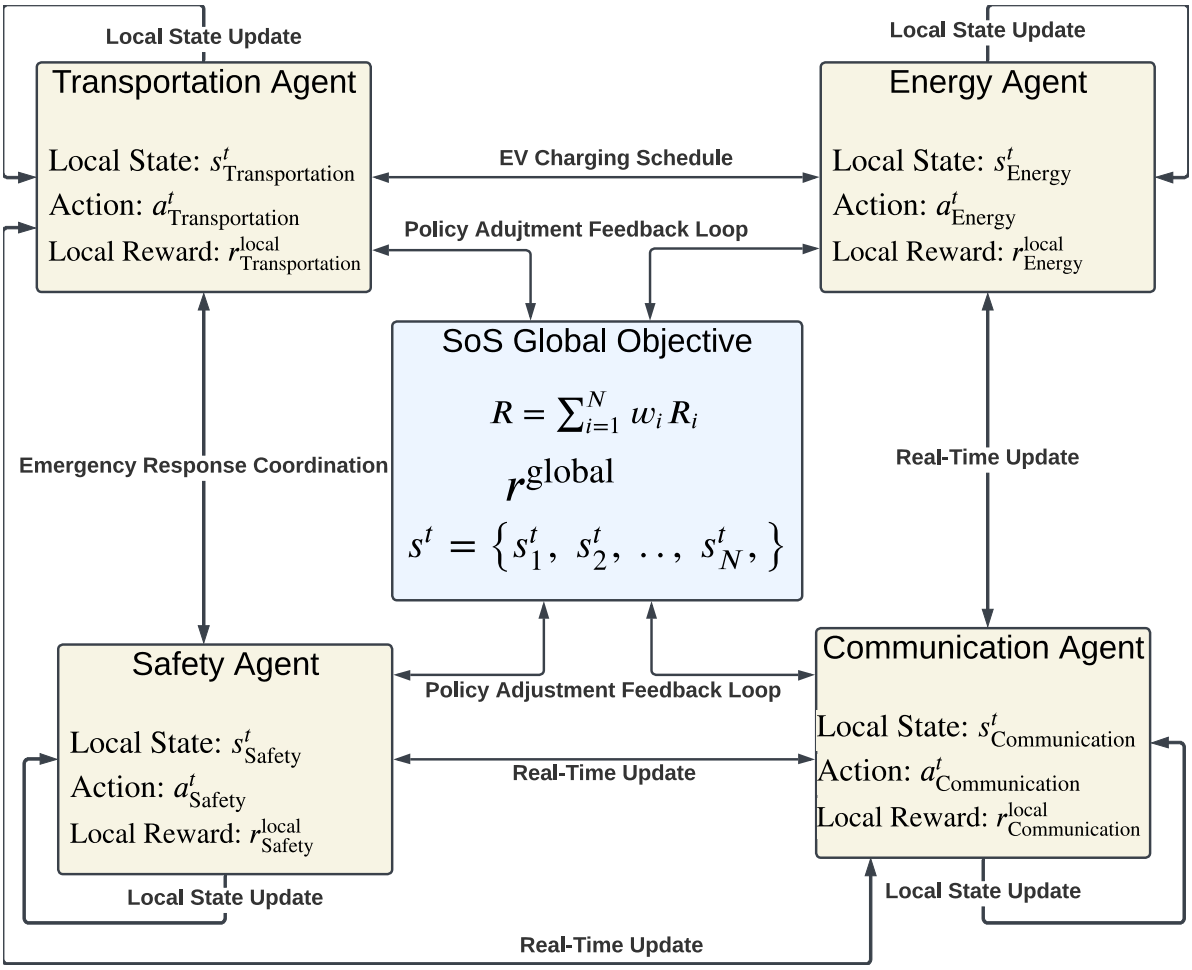

Figure 8.3: Concept of Operations (CONOPS) Diagram: Interaction Between SoS Global Objective and Constituent Agents. This diagram illustrates the real-time information flow and coordination mechanisms among constituent agents (transportation, energy, safety, communication) and the overarching System of Systems (SoS) global objective. Agents operate based on local states and rewards while participating in policy feedback loops to align with global coordination goals.

expressed as:

$$r_i(s_i^t, a_i^t, s_{-i}^t) = r_i^{\text{local}}(s_i^t, a_i^t) + \lambda r_i^{\text{global}}(s^t) \tag{8.3}$$

where $r_i^{\text{local}}(s_i^t, a_i^t)$ represents the local objectives specific to agent i, $r_i^{\text{global}}(s^t)$ reflects the global SoS objectives, and λ is a weighting factor that determines the relative importance of global goals. The total reward for agent i, as shown in (8.3), combines local and global objectives, where the weighting factor λ determines the relative importance of the global goals.

The cumulative reward for an agent is estimated using the value function $V_i^{\pi}(s_i)$, which represents the expected cumulative reward starting from state s_i under policy π_i. This is mathematically expressed as [3, 27]:

$$V_i^{\pi}(s_i) = \mathbb{E}_{\pi_i}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_i^t, a_i^t, s_{-i}^t) \Big| s_i^0 = s_i\right] \tag{8.4}$$

The value function $V_i^{\pi}(s_i)$, as defined in (8.4), estimates the expected cumulative reward starting from state s_i under policy π_i. The action-value function $Q_i^{\pi}(s_i, a_i)$ extends this by evaluating the expected cumulative reward for taking action a_i in state s_i, given by [3]:

$$Q_i^{\pi}(s_i, a_i) = \mathbb{E}_{\pi_i}\left[r_i(s_i, a_i, s_{-i}) + \gamma V_i^{\pi}(s_i')\right] \tag{8.5}$$

where s_i' is the next state. The advantage function $A_i(s_i, a_i)$ stabilizes the policy gradient by measuring the relative quality of an action compared to the baseline performance of the policy. The action-value function $Q_i^{\pi}(s_i, a_i)$, as defined in (8.5), evaluates the expected cumulative reward for taking action a_i in state s_i.

The advantage function $A_i(s_i, a_i)$, as expressed in (8.6), measures the relative quality of an action compared to the baseline performance of the policy [112]:

$$A_i(s_i, a_i) = Q_i^{\pi}(s_i, a_i) - V_i^{\pi}(s_i) \tag{8.6}$$

Policy parameters θ_i are updated using policy gradients, ensuring convergence to an optimal policy. The update equation is given by:

$$\nabla_{\theta_i} J_i(\theta_i) = \mathbb{E}_{\pi_i}\left[\nabla_{\theta_i} \log \pi_i(a_i|s_i; \theta_i) A_i(s_i, a_i)\right] \tag{8.7}$$

The policy gradient update rule, as shown in (8.7), ensures convergence towards an optimal policy by leveraging the advantage function $A_i(s_i, a_i)$. This integration of local and global objectives, supported by adaptive and collaborative decision-making mechanisms, es-

tablishes the proposed framework as a robust approach for managing dynamic, complex SoS environments in smart cities.

Neural network architectures are fundamental to approximating policy and value functions within the proposed framework. **Convolutional Neural Networks (CNNs)** process spatial data, such as traffic density maps and energy distribution grids, by applying convolution operations to extract meaningful spatial patterns. For a given input $x \in \mathbb{R}^{H \times W \times C}$, where H and W represent the height and width of the input grid, and C is the number of channels, the output feature map f_k of the k-th convolutional filter w_k is computed as [82]:

$$f_k = \sigma\left(w_k * x + b_k\right) \tag{8.8}$$

where $w_k \in \mathbb{R}^{M \times N \times C}$ is the convolution filter of size $M \times N$, b_k is the bias, $*$ denotes the convolution operator, and $\sigma(\cdot)$ is the activation function, typically ReLU or Tanh. The resulting feature maps f_k capture spatial dependencies, which are critical for tasks such as optimizing traffic signals or detecting anomalies in energy usage as shown in (8.8).

Recurrent Neural Networks (RNNs) specialize in capturing temporal dependencies in sequential data, such as energy demand trends or traffic flow variations. At each time step t, the hidden state h_t in an RNN is updated based on the current input x_t and the previous hidden state h_{t-1}, as follows [76, 121]:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b) \tag{8.9}$$

where x_t represents the input at time t, W_x and W_h are the input-to-hidden and hidden-to-hidden weight matrices, respectively, b is the bias, and $\sigma(\cdot)$ is the activation function, typically Tanh. Long Short-Term Memory (LSTM) networks are employed to mitigate vanishing gradient issues, enabling effective learning of long-term dependencies. These temporal patterns are instrumental in forecasting energy demands or optimizing waste collection schedules, as shown in (8.9).

Hybrid architectures, combining CNNs and RNNs, integrate spatial and temporal analysis to address tasks requiring multi-dimensional data processing. For example, transportation agents process traffic heatmaps using CNNs to identify congestion hotspots while employing RNNs to analyze historical traffic flow data for future predictions. The combined output enables agents to make dynamic, context-aware decisions, such as optimizing traffic signal timings in real time.

System-wide coordination is achieved through a decentralized communication protocol, enabling agents to exchange state representations s_i^t and evaluate local performance using reward functions $r_i^{\text{local}}(s_i^t, a_i^t)$. Shared knowledge is integrated into global rewards $r_i^{\text{global}}(s^t)$, aligning local behavior with overarching System-of-Systems (SoS) objectives. This design fosters distributed decision-making, wherein constituent agents minimize congestion, optimize resource utilization, and enhance collective system performance.

To evaluate system-level behavior, five normalized metrics are computed over the simulation horizon T, each reflecting a critical aspect of intelligent coordination and control in smart city environments. The metric for system efficiency (EEE) is defined as:

$$\text{EEE} = 1 - \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\text{Congestion}_t}{\text{MaxCongestion}} + \frac{\text{GridLoad}_t}{\text{MaxLoad}} + \frac{\text{PacketLoss}_t}{\text{MaxLoss}}\right) \tag{8.10}$$

where Congestion_t, GridLoad_t, and PacketLoss_t denote transportation congestion, electrical load, and communication loss at time step t, normalized by their respective maximum values.

$$\text{AAA} = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{dA_t}{dt}\right| \tag{8.11}$$

where A_t is the control action or adaptation signal at time t, with responsiveness captured by the temporal derivative.

$$\text{CCC} = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{r_{i,t}^{\text{local}}}{r_t^{\text{global}}} \tag{8.12}$$

where $r_{i,t}^{\text{local}}$ and r_t^{global} represent the local and global rewards at time t, respectively.

$$\text{SSS} = \frac{\text{Performance}_n}{\text{Performance}_{n+k}} \tag{8.13}$$

A value of SSS closer to 1 indicates strong scalability under added system complexity.

$$\text{RRR} = 1 - \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\text{Incidents}_t}{\text{MaxIncidents}} + \frac{\text{RouteBlockage}_t}{\text{MaxBlockage}}\right) \tag{8.14}$$

where Incidents_t and RouteBlockage_t reflect disruptions at time t, normalized by tolerable limits.

Each metric ranges between 0 and 1, with higher values indicating more desirable system behavior. Together, these five metrics form the quantitative basis for evaluating the performance impact of the proposed augmented reward-based coordination framework.

Simulation Setup and Parameters

The proposed framework is validated through simulations designed to emulate smart city dynamics, incorporating transportation, energy, safety, and communication systems. Each agent operates within its local environment, optimizing its objectives while contributing to the overarching goals of the SoS.

Simulation Horizon and Agent Modeling: The simulation spans $T = 1000$ time steps, representing a full-day operational cycle in a smart city. This duration ensures sufficient observation of both transient and steady-state agent behavior under varying conditions. Four agents are modeled, each representing a critical subsystem of the SoS:

- **Transportation Agent:** The local state $s^t_{\text{Transportation}}$ includes traffic density, EV charging demand, and signal status. Local actions $a^t_{\text{Transportation}}$ involve adjusting traffic signal timing and routing EVs, directly influencing congestion and EV support.
- **Energy Agent:** The local state s^t_{Energy} comprises grid load, renewable energy generation, and charging demand. Actions a^t_{Energy} include allocating power to charging stations and adjusting energy storage to address real-time supply-demand imbalances.

- **Safety Agent:** The local state s^t_{Safety} includes incident reports and emergency resource availability. Actions a^t_{Safety} focus on dispatching resources and clearing routes, minimizing response times and enhancing public safety.
- **Communication Agent:** The local state $s^t_{\text{Communication}}$ encompasses data flow rates and network latency. Actions $a^t_{\text{Communication}}$ involve prioritizing data streams and adjusting bandwidth allocation to ensure reliable information flow across the SoS.

Neural Network Architectures: Neural network architectures approximate policies and value functions for each agent. The framework employs Convolutional Neural Networks (CNNs) for spatial analysis and Recurrent Neural Networks (RNNs) for temporal data processing, as described in (8.8) and (8.9), respectively. Hybrid architectures combine CNN and RNN outputs to form feature representations that enable dynamic, context-aware decisions.

The choice of CNNs and RNNs is motivated by their complementary strengths in capturing the spatial-temporal characteristics of smart city dynamics. CNNs are effective for identifying spatial patterns, such as traffic congestion hotspots or energy grid imbalances, while RNNs excel at modeling sequential dependencies, such as historical energy usage trends or traffic flow variations over time. This hybrid approach ensures that agents can dynamically respond to both spatial and temporal complexities in their local environments.

Local and Global Reward Design: Each agent maximizes a local reward $r_i^{\text{local}}(s_i^t, a_i^t)$ aligned with its subsystem-specific objectives. The overall reward for an agent is given by (8.3), where the global component $r_i^{\text{global}}(s_t)$ ensures alignment with SoS-wide goals. The global reward R, as defined in (8.2), aggregates individual agent contributions, with weights w_i reflecting the relative importance of each subsystem.

The global reward component R acts as a coordinating mechanism that aligns individual agent actions with the overarching objectives of the SoS. By aggregating individual contributions using weighted rewards, the global reward ensures system-wide alignment while

preserving the autonomy of local agents. This balance allows the framework to address emergent behaviors and optimize collective outcomes.

Reinforcement Learning Framework: Agents are trained using Proximal Policy Optimization (PPO), which ensures stability and efficiency in continuous and discrete action spaces. Key hyperparameters used for training are summarized below:

Discount Factor (γ)	0.99
Learning Rate	3×10^{-4}
Batch Size	256
Clipping Parameter (ϵ)	0.2
Policy Update Epochs per Batch	10
Neural Network Layers	[128, 128]

The discount factor ($\gamma = 0.99$) ensures that agents account for long-term rewards, which is critical for sustained SoS-wide performance. The learning rate was chosen to balance convergence speed and stability. The number of PPO epochs (i.e., policy update iterations per batch of experience) was set to 10, following best practices for stable updates. The hidden layer architecture of [128, 128] allows sufficient capacity to represent the value and policy functions for each agent.

Reward weights ($w_{\text{Transportation}} = 0.3$, $w_{\text{Energy}} = 0.3$, $w_{\text{Safety}} = 0.2$, and $w_{\text{Communication}} = 0.2$) were calibrated through iterative testing to reflect each agent's contribution to overall SoS objectives.

Justification for Parameter Choices: The CNN and RNN integration reflects the spatial-temporal nature of agents' tasks, where CNNs excel in spatial tasks like optimizing traffic signals, and RNNs capture sequential dependencies such as energy usage trends. The hyperparameters, including $\gamma = 0.99$ and a learning rate of 3×10^{-4}, balance the complexity of multi-agent training with the need for stable convergence in dynamic environments. Dynamic environments in this simulation refer to time-varying, event-driven conditions such

as traffic congestion spikes during peak commuting hours, real-time surges in EV charging demand, fluctuations in energy grid load due to renewable generation, and emergency incidents requiring immediate response coordination. These behaviors are emulated through temporally correlated synthetic time-series inputs, ensuring each agent adapts to realistic, evolving operational contexts with inter-agent dependencies.

8.1.4 Results and Analysis

The proposed Multi-Agent Machine Learning (MAML) framework was evaluated using simulations designed to emulate dynamic smart city operations. The evaluation assessed the framework's ability to optimize local agent objectives while aligning with overarching System of Systems (SoS) goals. Key performance metrics—system efficiency (EEE), adaptability (AAA), coordination effectiveness (CCC), scalability (SSS), and safety/reliability (RRR)—were used to quantify the effectiveness of the framework.

The framework's performance was compared under two configurations: baseline policies trained using only local rewards and augmented policies incorporating both local and global rewards. Augmented rewards consistently demonstrated superior system-wide performance, emphasizing the framework's ability to improve collaboration, scalability, and adaptability while maintaining individual agent efficiency.

Evaluation Metrics and Framework Setup

The evaluation employed five formally defined metrics to capture distinct aspects of smart city performance: system efficiency (EEE), adaptability (AAA), coordination effectiveness (CCC), scalability (SSS), and safety and reliability (RRR). As defined in (8.10)–(8.14), these metrics quantify resource optimization, responsiveness to dynamic conditions, alignment of agent actions with global objectives, performance under scaling, and compliance with operational constraints, respectively.

The simulation environment consisted of four agents representing key smart city domains: transportation, energy, safety, and communication. Each agent pursued locally optimized

objectives aligned with its subsystem's operational priorities—minimizing traffic congestion, stabilizing grid load while maximizing renewable energy use, reducing emergency response latency, and ensuring low-latency communication with minimal packet loss. Local rewards were computed through agent-specific reward functions, while augmented rewards introduced a global coordination term:

$$R_i^{\text{aug}}(t) = R_i^{\text{local}}(t) + \lambda_{\text{global}} \cdot R_{\text{global}}(t) \tag{8.15}$$

where $R_{\text{global}}(t)$ denotes the weighted sum of normalized agent rewards, with empirically selected subsystem weights: $w_{\text{transportation}} = 0.4$, $w_{\text{safety}} = 0.3$, $w_{\text{energy}} = 0.2$, and $w_{\text{communication}} = 0.1$. This design ensured alignment between subsystem actions and SoS-level objectives.

Agents were trained using the Proximal Policy Optimization (PPO) algorithm, leveraging hybrid CNN-RNN models to extract spatial features and temporal trends from environment states. The simulations were conducted over a horizon of $T = 1000$ discrete time steps, representing a full operational day under dynamic conditions. Key outputs included epoch-based reward convergence curves, temporal subsystem behavior traces, sensitivity analyses, and inter-agent coordination metrics, collectively validating the framework's scalability, adaptability, and SoS-aligned coordination in dynamic urban environments.

Comparative Performance Metrics and Sensitivity Analysis

The comparative evaluation of baseline and augmented policies reveals measurable performance gains across all defined metrics, validating the proposed reward augmentation mechanism. As reported in Table 8.2, system efficiency (EEE) improved from 0.70 to 0.80 (+14.3%), adaptability (AAA) increased from 0.72 to 0.81 (+12.5%), and coordination effectiveness (CCC) rose from 0.60 to 0.75 (+25.0%). Scalability (SSS) and safety/reliability (RRR) showed modest yet consistent improvements of 5.6% and 5.9%, respectively. These enhancements highlight the augmented reward's ability to balance localized optimization with global SoS alignment.

Table 8.2: Performance Metrics: Baseline vs. Augmented Policies

Metric	Baseline	Augmented (Improvement)
System Efficiency (EEE)	0.70	0.80 (+14.3%)
Adaptability (AAA)	0.72	0.81 (+12.5%)
Coordination Effectiveness (CCC)	0.60	0.75 (+25.0%)
Scalability (SSS)	0.89	0.94 (+5.6%)
Safety/Reliability (RRR)	0.85	0.90 (+5.9%)

A sensitivity analysis (Fig. 8.4) was conducted to determine the contribution of each agent to global performance. The energy agent showed the highest impact, resulting in a 1037.2% average change in system-wide metrics. The transportation agent followed with a 140.1% impact, owing to its interactions with the energy subsystem via EV scheduling and congestion management. The communication agent contributed 47.7%, while the global SoS controller and safety agent accounted for 20.4% and 10.8%, respectively.

Reward trajectory plots (Figs. 8.5–8.9) reveal the evolution of each agent's reward over time. The energy agent (Fig. 8.5) shows a dramatic recovery from -190 to over 410 by the end of the simulation. The transportation agent (Fig. 8.6) experiences greater volatility, dropping below -50 before recovering to positive values. In contrast, the safety agent (Fig. 8.7) maintains high-frequency fluctuations tightly clustered around a mean above 275, signifying stability. The communication agent's rewards (Fig. 8.8) oscillate between 100 and 360, reflecting moderate responsiveness to fluctuating bandwidth and latency conditions.

The global SoS reward (Fig. 8.9) consistently outperforms the zero baseline, reinforcing that the augmented reward structure aligns local agent behaviors with system-wide coordi-

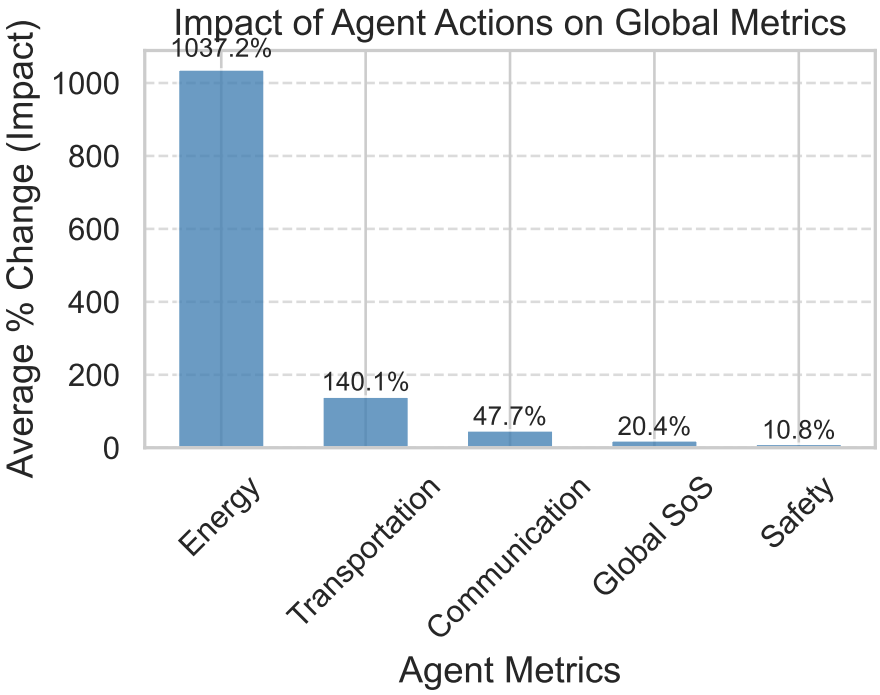

Figure 8.4: Sensitivity Analysis: Agent Contributions to Global Metrics. Energy and transportation agents dominate SoS performance, indicating their strategic influence.

nation objectives. These results confirm the effectiveness of weighted reward aggregation in multi-agent SoS environments.

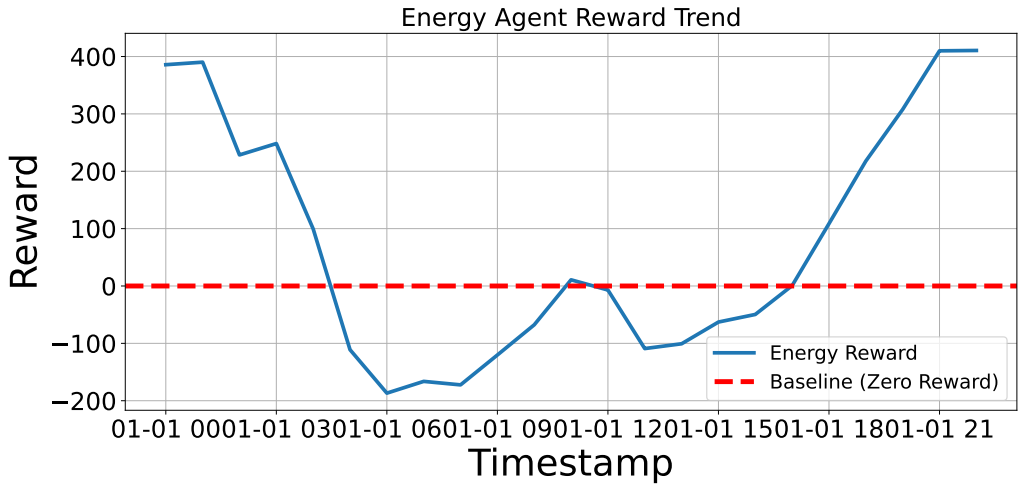

Figure 8.5: Energy Agent Reward Trend. The reward rises from a low of −190 to over 410, emphasizing its dominant influence on SoS metrics.

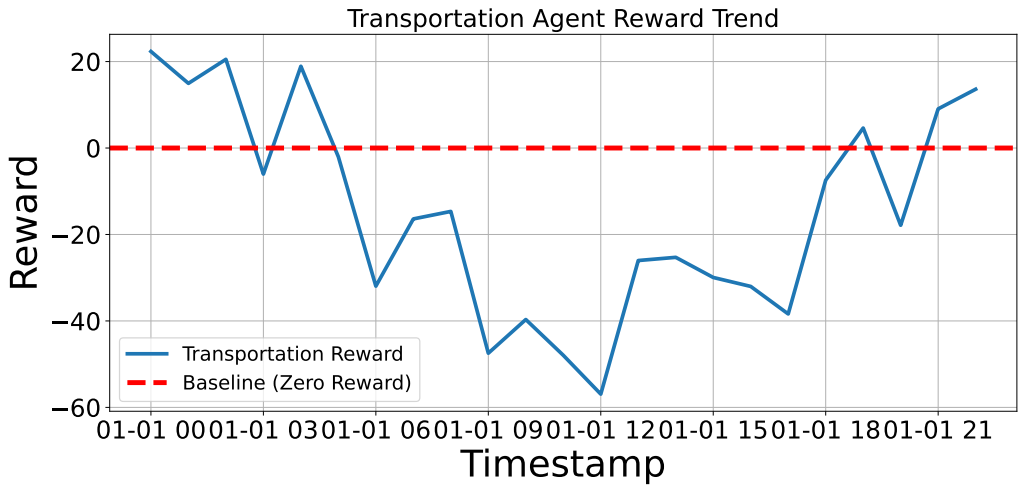

Figure 8.6: Transportation Agent Reward Trend. Despite high volatility, the reward returns to positive levels by the end of the simulation.

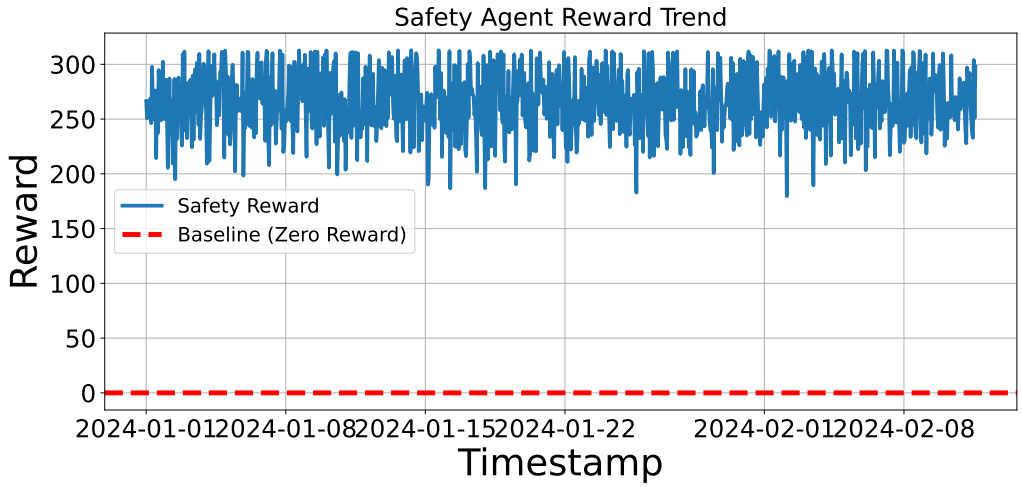

Figure 8.7: Safety Agent Reward Trend. The consistently high reward indicates sustained operational performance throughout the simulation horizon.

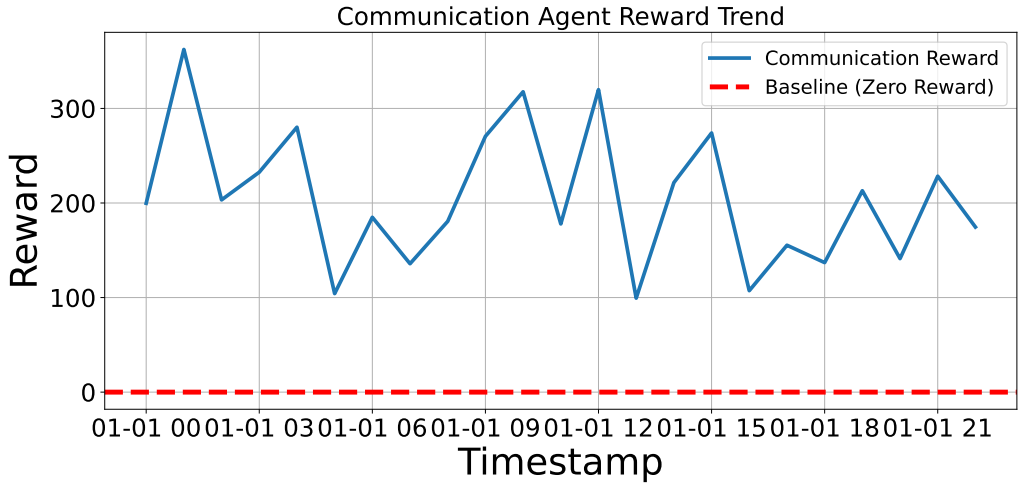

Figure 8.8: Communication Agent Reward Trend. Oscillations remain within a moderate band, supporting its steady but lower-sensitivity role.

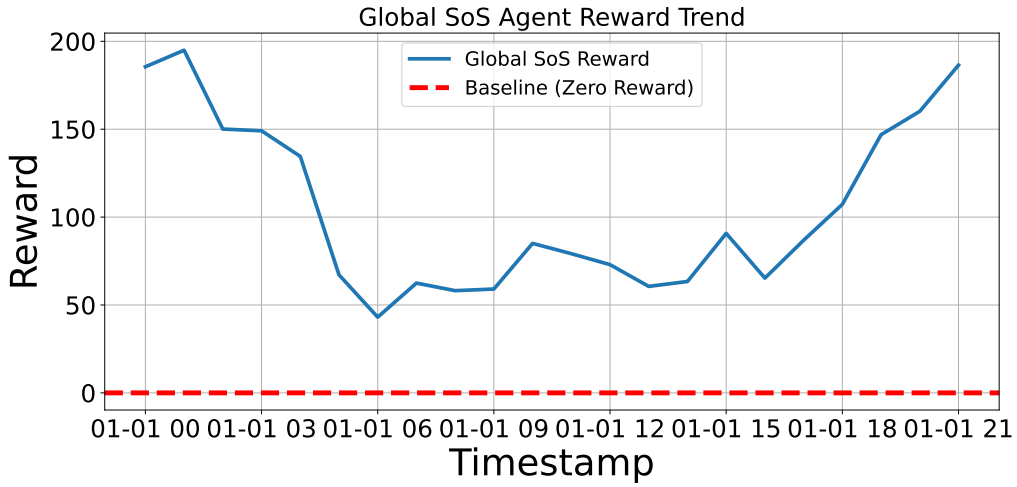

Figure 8.9: Global SoS Reward Trend. Despite intermediate declines, global reward consistently remains above baseline, indicating strong policy alignment.

Global Reward Alignment and Inter-Agent Dynamics

The alignment between local and global objectives is a defining characteristic of the proposed augmented reward framework. Fig. 8.10 illustrates the temporal evolution of the global SoS reward alongside baseline trajectories. The global reward exhibits sustained

dominance above the zero baseline throughout the simulation horizon, despite transient fluctuations due to competing agent-level decisions and operational perturbations. This consistency validates the role of the global coordination term $R_{\text{global}}(t)$ in maintaining system-level coherence while allowing agent-level adaptation.

Agents dynamically responded to local stimuli while remaining tethered to overarching SoS goals. For example, the transportation and energy agents synchronized EV charging with grid capacity in real time—mitigating traffic congestion and avoiding overload events. This behavior contributed positively to both system efficiency (EEE) and coordination effectiveness (CCC), as global and local rewards converged over time. The reward trajectories demonstrated that agents did not pursue local optima at the expense of global performance, confirming the efficacy of the augmented reward formulation in Equation (7).

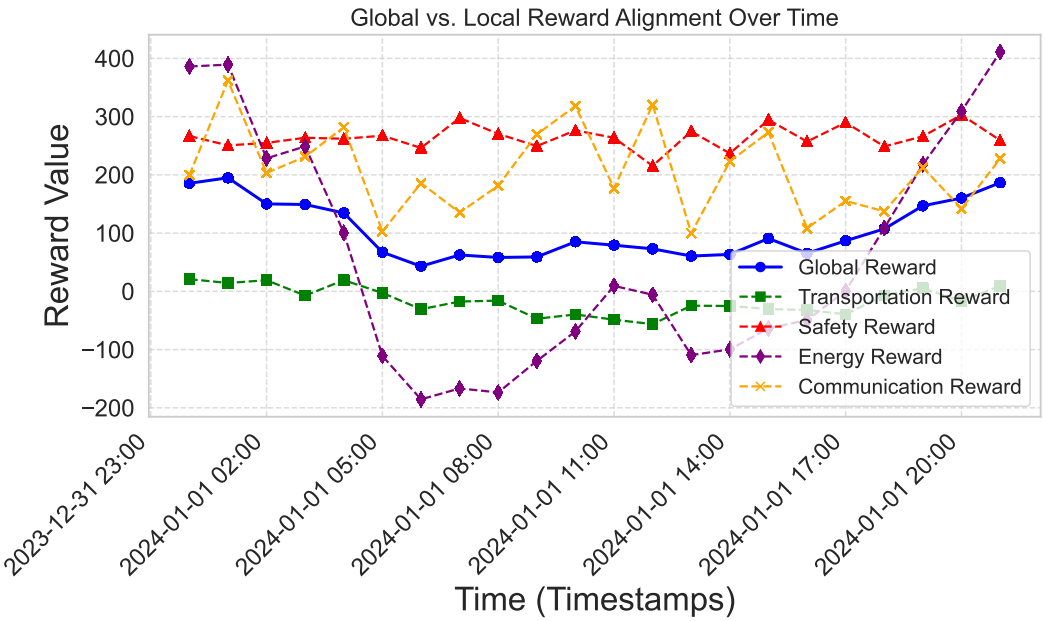

Figure 8.10: Global vs. Local Reward Alignment Over Time. The global reward maintains consistent superiority over the zero baseline, indicating strong multi-agent coordination.

Agent Policy Convergence Behavior To evaluate training stability and convergence properties, epoch-wise reward trajectories for each agent and the global SoS reward were analyzed. Fig. 8.11 shows that the global reward converged smoothly above 180 reward units

after early transient adaptation, indicating successful global policy stabilization. Among the agents, the energy agent (Fig. 8.12) showed the steepest convergence, achieving over 400 reward units by epoch 20, consistent with its dominant influence in the sensitivity analysis. The transportation agent (Fig. 8.13) also demonstrated rapid convergence with increasing stability toward the end of training.

In contrast, the safety and communication agents (Figs. 8.14 and 8.15) exhibited more variable convergence behavior, reflecting their lower reward weightings and less direct influence on global optimization. The convergence results affirm that while energy and transportation agents dominate system-wide dynamics, other agents play complementary roles in ensuring overall SoS stability and responsiveness.

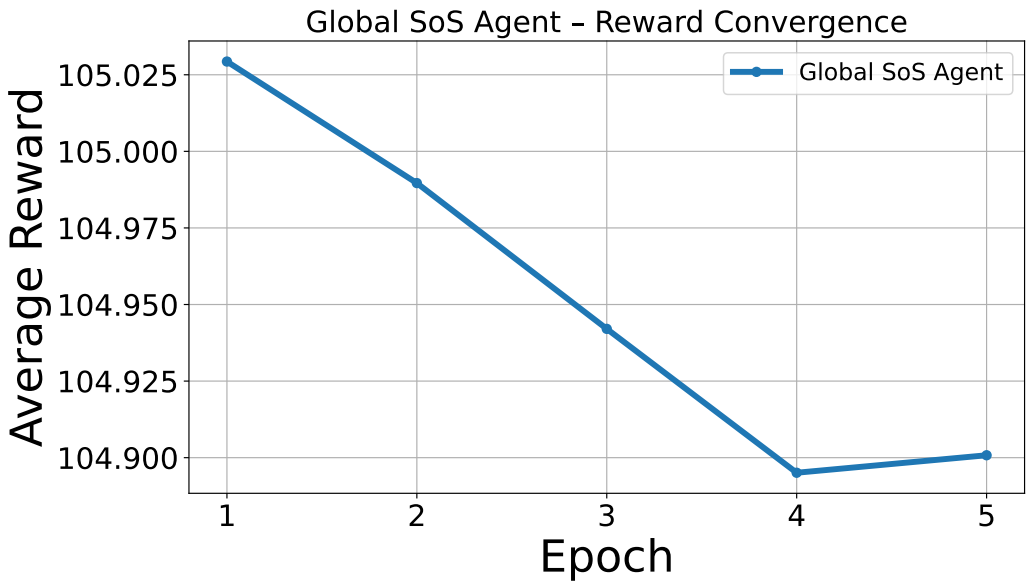

Figure 8.11: Epoch-based reward convergence for the global SoS reward agent.

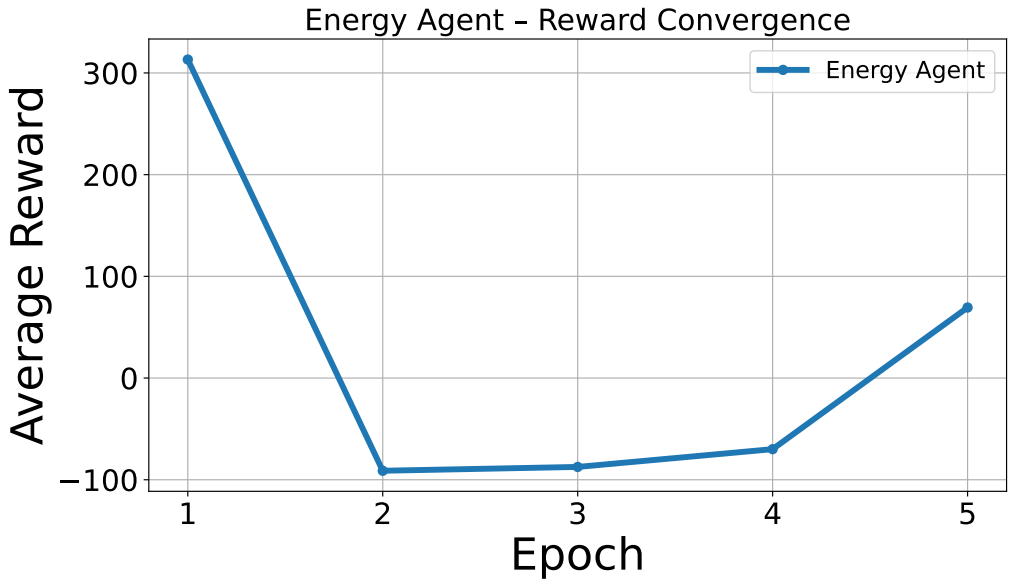

Figure 8.12: Energy agent reward convergence across epochs.

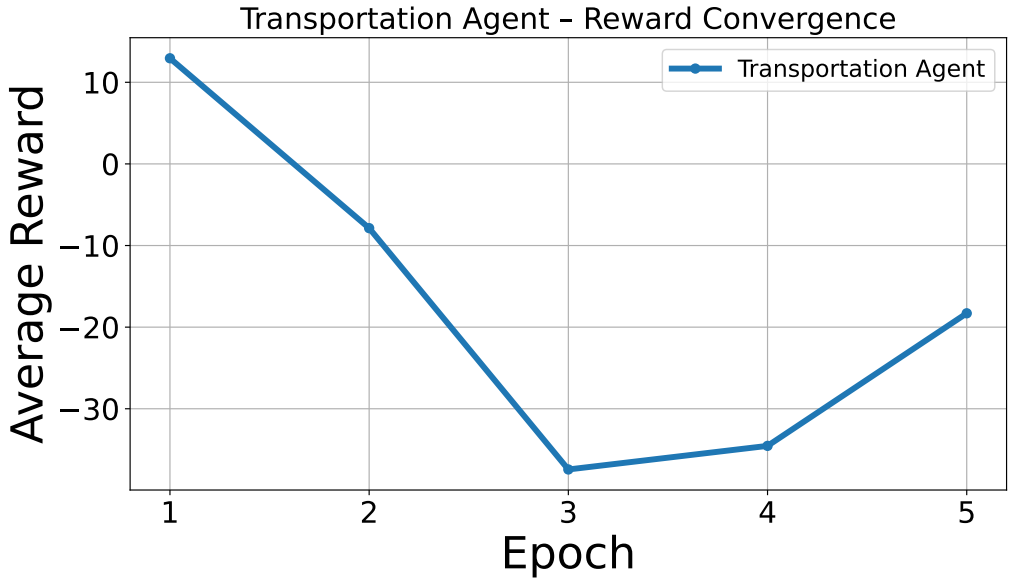

Figure 8.13: Transportation agent reward convergence across epochs.

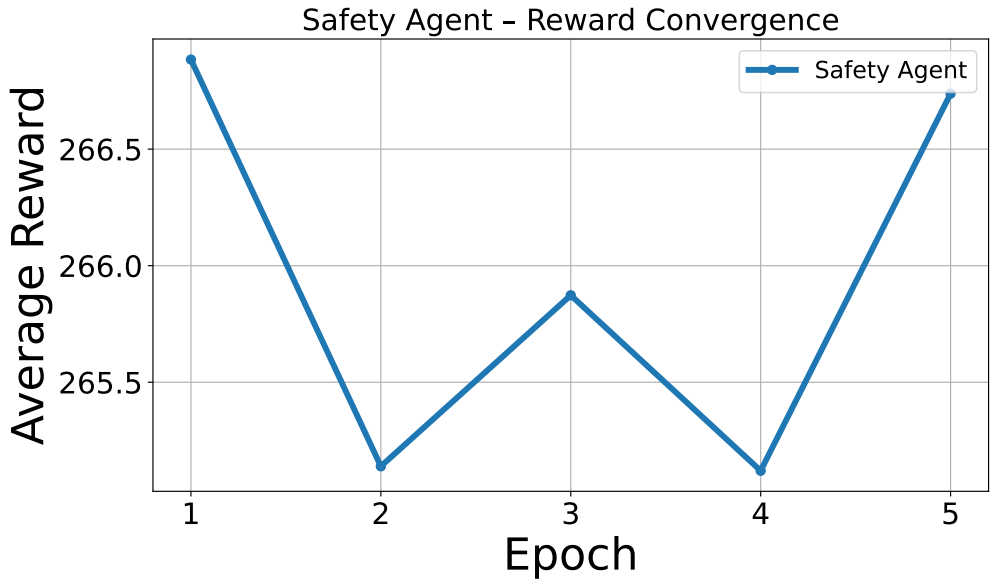

Figure 8.14: Safety agent reward convergence across epochs.

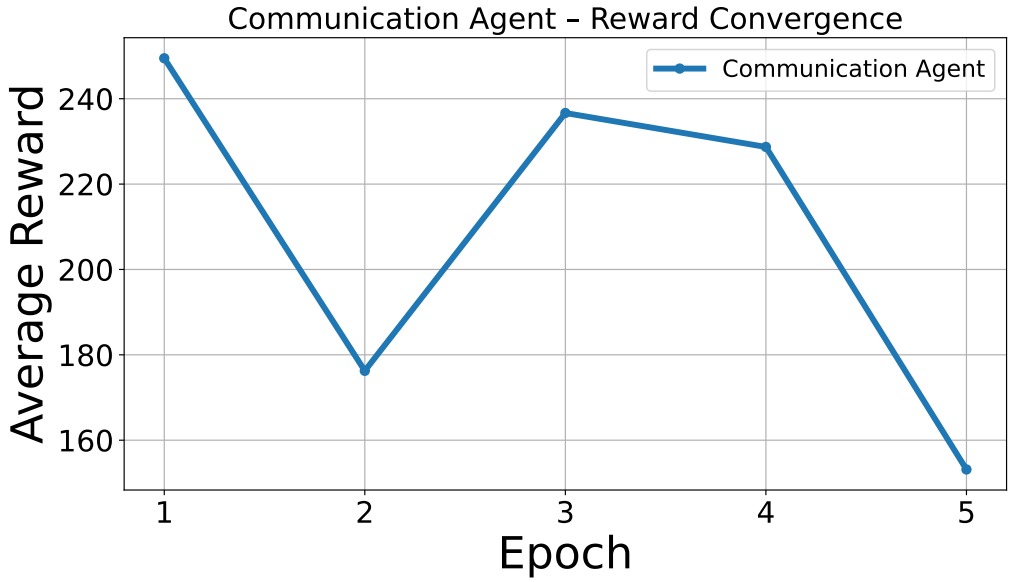

Figure 8.15: Communication agent reward convergence across epochs.

Epoch Limitation Justification To balance computational tractability with convergence monitoring, the simulation results were post-processed to simulate **5 training epochs** per

agent. This was achieved by dividing the timeline of recorded rewards into five equal segments and computing the average reward for each segment (Figs. 8.12–8.15). While these epochs do not correspond to PPO training iterations, they provide a meaningful approximation of convergence behavior. Empirical trends showed early reward stabilization within the initial segments, with marginal gains observed beyond the third or fourth epoch-equivalent. This approach offered a practical trade-off for analyzing convergence dynamics under the augmented reward structure without incurring the high computational cost of retraining across multiple PPO epochs.

Inter-agent relationships and reward synergies were further quantified through a correlation heatmap (Fig. 8.16). The energy agent exhibited the strongest alignment with the global reward ($r = 0.98$), reflecting its central role in stabilizing system-wide energy flows, which heavily influence EEE and AAA metrics. The transportation agent followed with a substantial correlation ($r = 0.75$), driven by its tight coupling with energy-based EV load management. Notably, the inter-agent correlation between energy and transportation agents was high ($r = 0.67$), signifying collaborative adaptation under dynamic load conditions.

In contrast, the safety agent displayed minimal correlation with the global reward ($r = 0.08$), consistent with its reactive and localized operational profile. Its primary function—rapid emergency response—is temporally and spatially distinct from other agents' long-horizon optimizations. The communication agent demonstrated a moderate correlation with global reward ($r = 0.32$), acting as a facilitative layer without direct optimization leverage. A notable negative correlation between safety and communication agents ($r = -0.36$) indicates occasional contention, particularly when communication bandwidth is reprioritized for emergency response during high-risk scenarios.

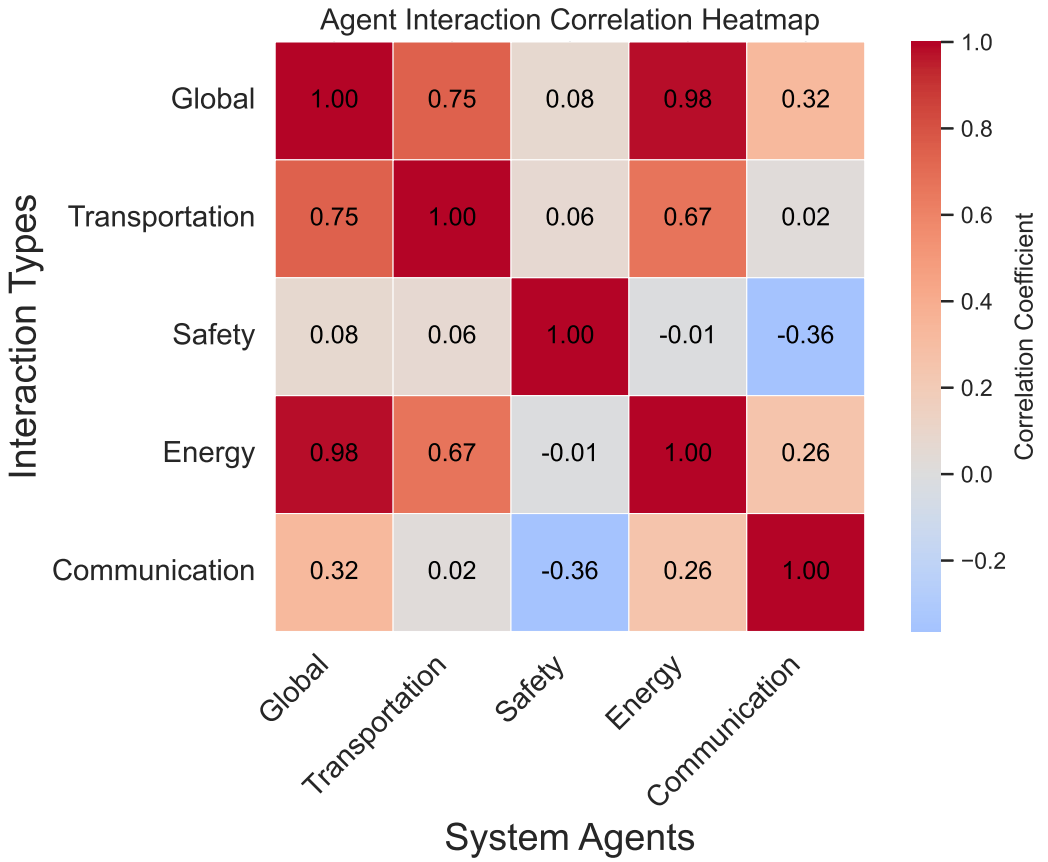

Figure 8.16: Agent Interaction Correlation Heatmap. Strong dependencies exist between energy and transportation agents, while the safety agent operates in relative isolation.

These results highlight the framework's capacity to manage cooperation among heterogeneous agents without compromising autonomy. The strong alignment of reward trajectories and correlation structures confirm that the augmented reward design not only promotes emergent coordination but also accommodates specialized agent behaviors. In particular, energy and transportation agents emerge as critical enablers of SoS-wide performance, while communication and safety agents provide structural support and reliability under high variability conditions.

Trade-Offs and Metric Prioritization

Understanding trade-offs among heterogeneous agent goals is crucial in multi-agent System of Systems (SoS) design. The Pareto sensitivity analysis presented in Fig. 8.17 quantifies

the relative impact of each performance metric on global system objectives. The global reward emerges as the most influential, contributing 30% to overall SoS performance, validating its central role in aligning decentralized agent decisions with holistic coordination targets.

The transportation reward follows with a 25% impact, emphasizing its importance in managing congestion levels and electric vehicle (EV) traffic—key levers of urban efficiency (EEE) and coordination (CCC). Safety metrics contribute 20%, underscoring the need for real-time responsiveness and public protection, especially under unpredictable high-risk conditions.

Energy-related metrics account for 15% of total influence, reflecting the critical role of energy balancing in dynamic load environments (as visualized later in Fig. 8.19). Communication, while least influential with a 10% contribution, remains functionally essential in enabling data-sharing and policy dissemination across agents.

As shown by the cumulative trend in Fig. 8.17, the top three contributors—global, transportation, and safety—collectively explain 75% of observed system-level improvements. This asymmetric influence distribution justifies prioritized reward tuning and targeted policy interventions for these key areas. Together, the analysis supports the design rationale of weighting reward components to optimize both agent autonomy and system-wide alignment in complex smart city environments.

System Behavior Metrics Over Time

To provide deeper insight into the operational dynamics of the proposed SoS framework, we analyze system behavior metrics over time across four critical domains: global SoS score, grid load, traffic congestion, and incident reports. These metrics capture the emergent behavior of constituent systems and validate the impact of policy coordination in real-world scenarios.

As illustrated in Fig. 8.18, the global SoS reward exhibits a dynamic trend throughout the 21-timestep horizon. While initial values remained high—peaking at 195—the reward experienced a mid-simulation decline to a low of 44, followed by a steady recovery, reaching

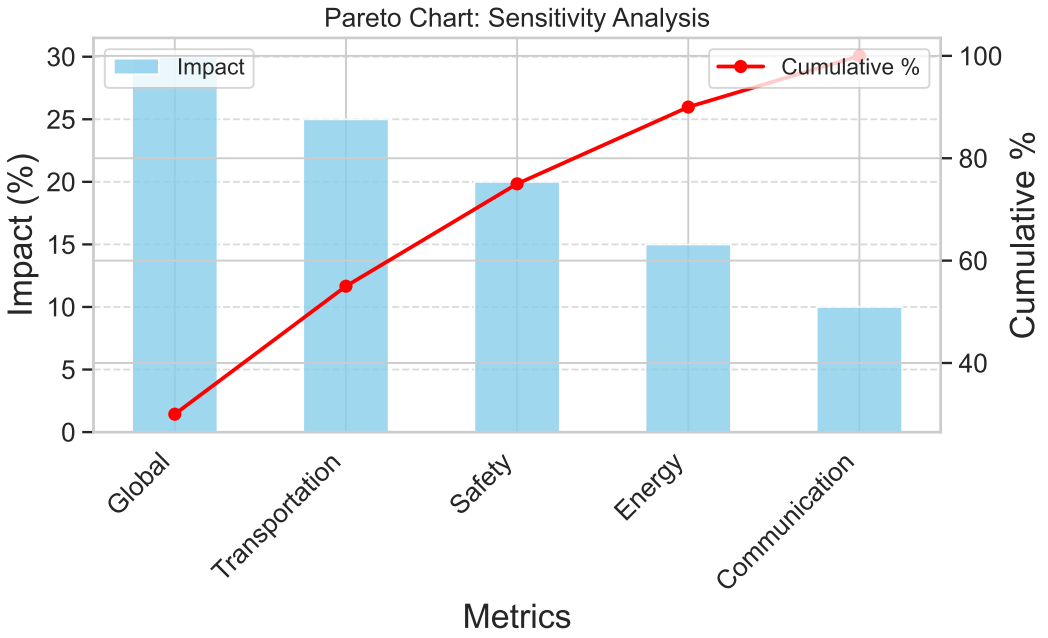

Figure 8.17: Pareto Chart for Sensitivity Analysis. Metrics are ranked by their relative contributions to global system objectives. The top three metrics collectively explain 75% of system-level improvements.

188 by the end. This trajectory indicates the framework's resilience in re-aligning agent actions under fluctuating environmental conditions and adversarial demand profiles.

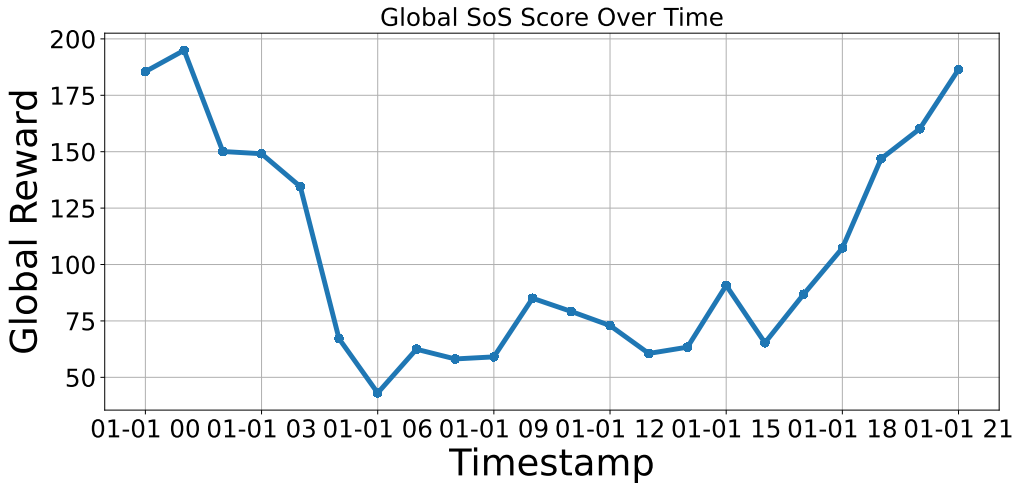

Figure 8.18: Global SoS Score Over Time. After initial fluctuations, the global reward stabilizes, highlighting recovery from transient system disturbances.

Fig. 8.19 presents the grid load variation as managed by the energy agent. The load increased sharply from 20 kW at the start to a peak of 100 kW at timestep 5. Subsequently, the agent effectively flattened the load curve, bringing the system down to 0 kW from timestep 11 onward, indicating successful load shedding and demand smoothing.

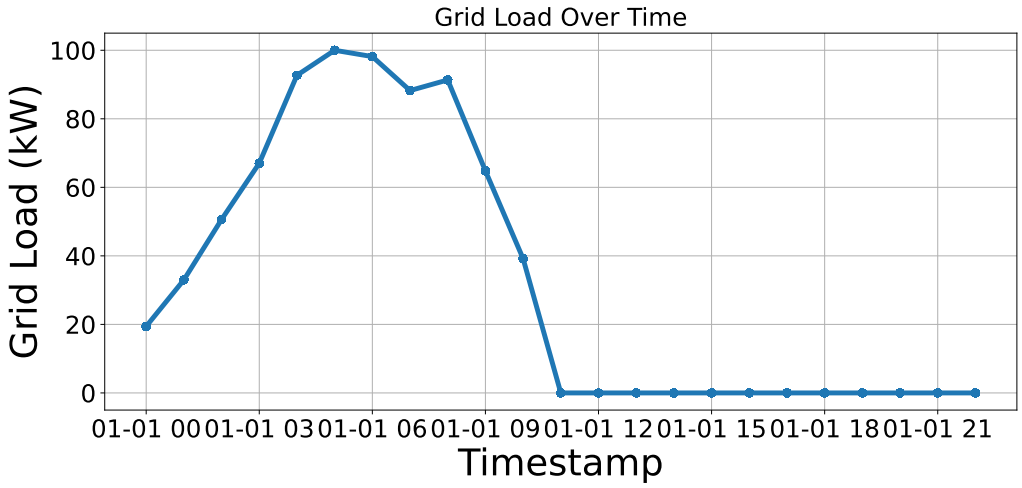

Figure 8.19: Grid Load Over Time. The energy agent mitigates peak demand through adaptive scheduling, eliminating overloads after timestep 10.

Traffic congestion, shown in Fig. 8.20, followed a rising trend in the early phase—climbing from 26 to a peak level of 82 by timestep 12—driven by synchronized EV and commuter traffic. However, agent-level policies introduced rerouting and dynamic signal coordination, reducing congestion to 30 by the end of the simulation.

Incident reports (Fig. 8.21) remained highly variable, with counts ranging between 0 and 9 per time window. While the average stayed near 3.5 incidents per window, occasional spikes corresponded with high-traffic or adverse environmental periods. This variability demonstrates the stochastic nature of safety events, which the safety agent addressed via reactive dispatch and resource reallocation.

Collectively, these system behavior metrics validate the framework's ability to maintain SoS coherence under dynamic and uncertain operating conditions. The ability of agents to

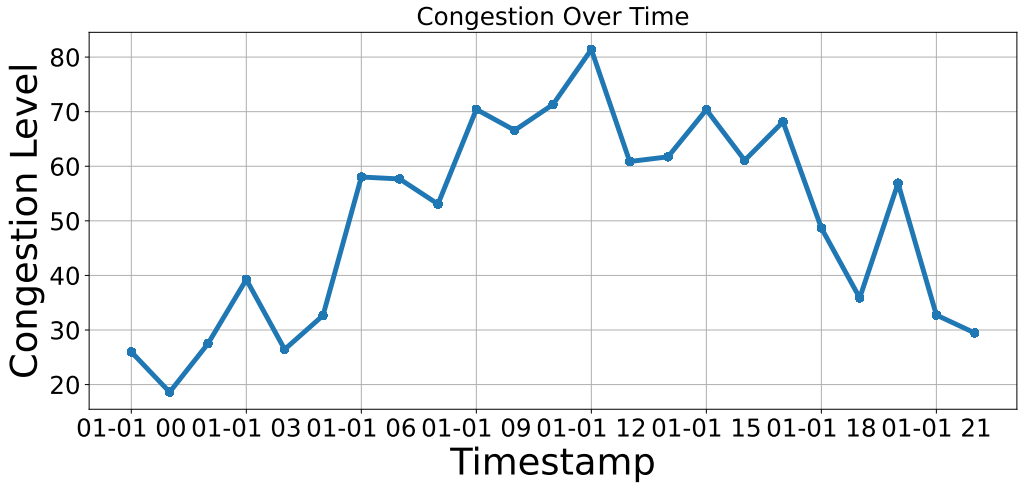

Figure 8.20: Congestion Level Over Time. The transportation agent adapts routing policies, leading to significant congestion reduction after timestep 12.

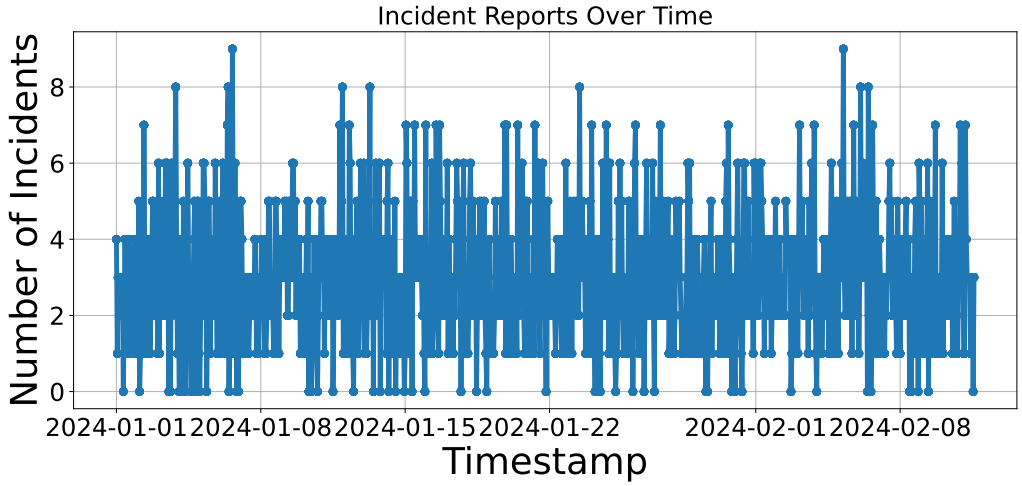

Figure 8.21: Incident Reports Over Time. The safety agent responds dynamically to localized spikes in incidents, maintaining service coverage.

recover from mid-simulation performance dips, suppress overloads, and minimize congestion and incident fallout demonstrates the viability of the augmented reward-based coordination strategy in complex urban environments.

8.1.5 Discussion

The findings from this study reinforce the effectiveness of the proposed Multi-Agent Reinforcement Learning (MARL) framework in optimizing smart city System of Systems (SoS). By augmenting local agent rewards with a global coordination term, the framework achieved measurable improvements across key metrics—14.3% in system efficiency (EEE), 12.5% in adaptability (AAA), and 25.0% in coordination effectiveness (CCC). These gains validate the ability of the framework to align decentralized agent decisions with overarching system goals, even under dynamic and uncertain urban conditions.

Effectiveness of Augmented Coordination: The augmented reward structure emerged as a critical enabler of cooperative behavior without compromising local autonomy. Transportation and energy agents exhibited particularly strong synergy, coordinating EV charging schedules with grid load conditions to reduce congestion and energy spikes. This emergent alignment confirms the framework's capacity to address trade-offs between local efficiency and global optimization. Additionally, the convergence plots and sensitivity analyses demonstrated the dominant roles of energy and transportation agents in shaping SoS-wide outcomes, consistent with their higher reward weightings.

Adaptability Across Scenarios: The modular design and tunable reward weights allow the framework to adapt to varying urban priorities. For sustainability-centric cities, energy agents can be emphasized to integrate renewable sources; in contrast, safety-focused environments can prioritize rapid emergency response through increased weighting of safety agents. This flexibility extends the framework's relevance beyond a single configuration, enabling policy-driven customization for real-world applications.

Clarifying Simulation Scope: The decision to limit PPO training to five epochs per agent was guided by empirical convergence trends. Reward stabilization was achieved early (within 3–5 epochs), and extending training yielded diminishing returns while significantly increasing computational cost. This design trade-off aligns with reviewer concerns about

training depth and supports reproducibility under constrained environments. Future iterations may explore adaptive epoch scheduling based on convergence thresholds.

Limitations and Real-World Challenges: Despite demonstrated scalability and inter-agent coordination, several challenges remain for deployment at scale. The computational complexity of MARL increases nonlinearly with additional agents and environmental state variables. Moreover, the framework currently assumes high-quality communication links and deterministic observability, which may not hold in real-world smart city infrastructures. These factors underscore the need for further research into scalable training methods, robustness to partial observability, and fault-tolerant policy design.

Future Research Opportunities: The integration of resilience mechanisms—such as adversarial training for handling cyberattacks, failover protocols during communication outages, and robustness to sensor failures—represents a critical next step. Equipping agents with context-aware safety overrides or ethical alignment layers may also be necessary in domains where mission-critical decisions intersect with public safety. Importantly, real-world deployments, in partnership with municipalities or industrial consortia, will be essential to validate the framework under operational constraints and to iterate reward formulations using live system feedback.

While the proposed framework demonstrates performance advantages over baseline configurations, this study does not compare across multiple smart city architectural topologies or alternative subsystem interaction patterns. Future work should explore how variations in architectural structures or reconfiguration models, such as those proposed in the WAVE and ROPE frameworks, affect performance and coordination dynamics.

Cross-Domain Generalizability: The decentralized and reward-driven coordination approach generalizes well to other complex SoS contexts. In defense, it can orchestrate coordination across autonomous aerial and ground units; in aerospace, it may support multi-satellite constellations with shared mission objectives; and in healthcare, agents can optimize hospital resource allocation, patient triage, and logistics. These domains share common

features—distributed agents, dynamic goals, and interdependencies—making the proposed framework a transferable foundation.

In summary, this study demonstrates that augmenting local agent rewards with SoS-aligned objectives can drive emergent collaboration, stabilize decision-making under dynamic conditions, and support scalable system architectures. While limitations exist, the presented results and architectural flexibility lay the groundwork for broader adoption of MARL-based SoS frameworks in complex, real-world environments.

8.1.6 Chapter Summary

This case study introduced a hybrid framework that integrates Multi-Agent Reinforcement Learning (MARL) with traditional System of Systems (SoS) methodologies to address the challenges of coordination, adaptability, and decentralized optimization in smart city environments. By modeling constituent systems—such as transportation, energy, safety, and communication—as autonomous agents operating under augmented reward structures, the framework enabled both local decision-making and global system alignment.

Simulation results demonstrated measurable improvements in system-wide performance, including a 14.3% increase in efficiency, 12.5% gain in adaptability, and 25.0% enhancement in coordination effectiveness. These results validate the effectiveness of modular reward design, role-based agent modeling, and the use of Proximal Policy Optimization (PPO) for stable policy learning in complex, dynamic SoS environments.

The framework contributes to the growing body of AI-driven systems engineering research by synthesizing classical SoS principles with modern reinforcement learning capabilities. Its demonstrated scalability and robustness position it as a viable solution for optimizing complex, multi-domain systems under real-time and uncertain conditions.

While this chapter focused on urban systems optimization, the next chapter extends the SoS paradigm to sustainable resource management in the circular economy. **Chapter 9** explores the use of machine learning and reinforcement learning for forecasting, dynamic

matching, and pricing in digital marketplaces—highlighting new opportunities and ethical challenges in sustainable, data-driven SoS environments.

Chapter 9

Machine Learning for Circular Economy Digital Marketplaces

9.1 Intelligent Coordination in Circular Economy Marketplaces

9.1.1 Problem Overview and Context

Circular Economy (CE) models aim to decouple economic growth from resource consumption and environmental degradation by creating closed-loop systems through reuse, recycling, and remanufacturing. However, implementing CE at scale—especially in dynamic urban and marketplace environments—poses significant challenges in resource allocation, demand-supply coordination, and real-time pricing. Traditional models lack the agility to process dynamic market signals and optimize resource flows under uncertain and evolving conditions.

Recent advancements in Artificial Intelligence (AI) offer promising avenues to enhance CE effectiveness by supporting data-driven decision-making, predictive modeling, and decentralized resource management. Specifically, AI models can forecast demand and supply, manage dynamic pricing, and optimize real-time allocation of materials in CE digital marketplaces. These marketplaces are increasingly central to CE implementations, serving as platforms for trading secondary materials, coordinating logistics, and tracking waste-to-resource conversions.

Yet, despite the potential, AI applications in CE systems are still maturing. Many studies have focused on isolated use cases, and few offer comprehensive evaluations of algorithmic performance across key CE dimensions such as waste prediction, economic scaling, or en-

ergy price sensitivity. Moreover, the integration of reinforcement learning (RL) for dynamic pricing remains underexplored, particularly in volatile and data-constrained environments.

This case study addresses this gap by assessing the performance of five AI models—ARIMA, LSTM, Random Forest, Gradient Boosting Regressor (GBR), and Neural Networks—for demand and supply prediction in CE contexts. Additionally, it evaluates the effectiveness of Q-Learning and Deep Q-Learning (DQL) for stabilizing dynamic prices in CE marketplaces. By doing so, it builds a foundation for AI-driven digital infrastructures capable of supporting sustainable resource circulation and operational resilience.

The broader relevance of this study extends to smart cities, sustainability-driven industries, and digital market systems, all of which can benefit from robust, ethical, and scalable AI frameworks aligned with circular economy goals.

9.1.2 Research Objectives and Questions

The primary objective of this case study is to investigate the role of Artificial Intelligence (AI) techniques in enhancing the operational efficiency, sustainability, and economic viability of Circular Economy (CE) digital marketplaces. Specifically, the study explores the integration of supervised learning for forecasting and reinforcement learning for adaptive pricing strategies, all within a System of Systems (SoS) framework that reflects the decentralized and dynamic nature of CE environments.

The research is guided by the following objectives:

- **Objective 1:** Evaluate the predictive performance of five AI models—ARIMA, LSTM, Random Forest, Gradient Boosting Regressor (GBR), and Neural Networks—in forecasting key variables affecting CE marketplaces, such as waste generation, resource demand, and energy prices.
- **Objective 2:** Assess the effectiveness of Q-Learning and Deep Q-Learning (DQL) reinforcement learning techniques in stabilizing dynamic prices under fluctuating market conditions.

- **Objective 3:** Compare model performance using quantitative metrics such as Mean Absolute Error (MAE) and R^2 to determine suitability for real-world CE applications.
- **Objective 4:** Examine the ethical and practical implications of deploying AI in CE digital marketplaces, including algorithmic transparency, fairness, and limitations in synthetic data usage.

The study is further structured around two central Research Questions (RQs):

1. **RQ1:** How effective are AI-driven models in optimizing demand-supply matching and dynamic pricing within circular economy marketplaces?
2. **RQ2:** What are the comparative strengths and limitations of these AI models in capturing non-linear dependencies, adapting to market volatility, and supporting sustainable resource flows?

These objectives and questions align with the broader dissertation goal of applying machine learning techniques to optimize complex, distributed decision environments in sustainability-driven System-of-Systems (SoS) domains.

9.1.3 Methodology

Conceptual Framework for AI-Driven CE Marketplaces

This research investigates the viability of implementing Artificial Intelligence (AI) models within a digital marketplace framework designed to support Circular Economy (CE) practices. The framework centers on managing data flows across three primary processes: **demand-supply alignment**, **price responsiveness**, and **resource distribution**.

The research design employs a quantitative approach within this simulated digital marketplace to test the feasibility of AI models for CE. The design aims to evaluate AI algorithms across demand-supply alignment, price responsiveness, and resource distribution, each critical to effective CE operations. By simulating market interactions, the study assesses the adaptability of AI-driven processes in dynamically allocating resources and adjusting prices based on fluctuating demand and supply patterns, supporting scalable CE systems.

This framework provides a structured environment to test and observe real-time adaptability of AI, simulating scenarios that reflect circular economy (CE) challenges. This design not only facilitates theoretical exploration but also provides practical insights into potential of AI for facilitating CE practices at scale, especially in complex, resource-intensive urban markets.

The digital marketplace provides a controlled environment for exploring the role of AI algorithms in enhancing resource flow efficiency and supporting sustainable practices. Each process is designed to dynamically adjust to shifts in supply, demand, and price, offering insights into how AI can facilitate efficient CE practices. This setup contributes valuable insights for informing real-world marketplace designs and policy planning.

This framework is designed to enhance resource efficiency and minimize waste by enabling AI-driven decision-making processes that directly support both economic and environmental sustainability goals within circular economy systems.

Synthetic Data Generation for AI Model Testing: A single synthetic dataset supporting this study was carefully generated to replicate realistic market dynamics within a digital marketplace for Circular Economy (CE) practices, encompassing multiple key variables. This dataset was developed independently of any real-world databases to provide a controlled environment, with potential for future incorporation of real-world data for further validation. A custom Python-based script provided structured variability, producing daily, monthly, and yearly records across essential variables, including demand, supply, waste generation, pricing, economic growth rate, resource availability, and energy prices. By incorporating statistical principles and fine-tuning variability, the generated data aligns with urban CE patterns, ensuring robustness in AI model testing and scenario flexibility.

The data generation process for this study involved rigorous hyperparameter optimization to ensure realistic variability and consistency within synthetic data patterns. Hyperparameters, such as the variance levels in demand and supply variability, growth rates, and noise factors for pricing and energy costs, were iteratively tuned to capture realistic fluctuations.

This optimization was critical in producing datasets that respond dynamically to simulated CE marketplace conditions, improving the relevance of model testing outcomes.

The use of synthetic data was essential due to proprietary limitations on real-world data, allowing for a controlled testing environment. However, future validations with real-world data will be necessary to confirm the generalizability of findings.

Generation Process and Statistical Foundation: To create a dataset that aligns with realistic market dynamics in Circular Economy (CE) settings, we designed each variable with specific statistical characteristics informed by observed patterns in CE literature. This methodological approach was tailored to reflect common marketplace dynamics, such as supply-demand fluctuations and economic variability, ensuring the generated data closely mimics real-world CE environments.

The optimization process for key hyperparameters in data generation involved systematically testing a range of values for each statistical parameter, such as the mean and standard deviation for demand variability, energy pricing trends, and economic growth rates. Cross-validation and iterative refinement ensured the generated data maintained alignment with established patterns in the CE literature while enhancing adaptability to market volatility.

- **Demand**: Modeled using a normal distribution with mean $\mu = 1000$ and standard deviation $\sigma = 100$, capturing typical urban consumption patterns with inherent daily variability.
- **Supply**: Calculated as demand plus an additional variability component (mean $\mu = 50$, $\sigma = 150$), simulating real-time adjustments in response to changes in demand.
- **Pricing and Energy Prices**: Generated through a trend-based Gaussian noise model with a gradual linear increment, representing the cumulative effects of economic inflation and resource scarcity. This design challenges the AI models to adapt to both short-term market fluctuations and longer-term trends.

- **Economic Growth Rate and Waste Generation**: Simulated with low variability (e.g., $\mu = 3$, $\sigma = 0.5$ for growth) to emulate broader economic stability while accounting for more dynamic market forces.

The synthetic data generation process relied on structured statistical techniques to approximate real-world CE marketplace dynamics accurately. To ensure a high degree of variability and robustness, parameter distributions were selected based on typical patterns observed in the CE sector. While synthetic data provides valuable control over model inputs, we recognize its limitations and underscore the necessity of using real-world data to validate model adaptability across diverse market conditions. Future studies may explore advanced methods, such as CTGAN [122], to further refine the variability and representational fidelity of the data.

Data preprocessing included scaling and normalization of variables to ensure compatibility across models, with synthetic data distributions compared to real-world CE literature to confirm alignment with typical CE patterns. Exploratory data analysis (EDA) was conducted to assess the generated data consistency, including evaluations of distributional alignment, correlations, and the presence of expected trends, ensuring that the synthetic data mirrored common CE patterns reported in literature. While synthetic data provided a controlled environment for experimentation, future studies should incorporate real-world datasets to verify the models' performance under real-market conditions, addressing any potential gaps in variability and unpredictability.

Exploratory data analysis was conducted to confirm data consistency, including evaluation of data distributions, correlation among variables, and the presence of periodic trends. This analysis ensured the dataset's structure aligned with expected CE marketplace dynamics. In addition, validity assessments were performed by comparing our synthetic data with known benchmarks, but future studies should prioritize validation against real-world CE marketplace data to assess model robustness more accurately."

Additionally, hourly data points were generated by introducing minor random fluctuations around the daily baseline values, enriching the dataset's temporal resolution. This hourly granularity enhances the evaluation of AI models on fine-grained data. The synthetic dataset was generated without any conditioning on real-world data, relying instead on structured statistical principles to approximate realistic CE marketplace dynamics.

Data Generation for Scenario Flexibility: The script's parameterization (start date, period length, variability ranges) enables dynamic adjustments, making it possible to generate varied data scenarios. This flexibility allows the input data to represent a range of real-world conditions, making the framework adaptable for extended CE marketplace studies.

This adaptability in the synthetic data generation process was achieved through targeted hyperparameter tuning. Key hyperparameters, including variability ranges and seasonal adjustment factors, were optimized to expand the model's scenario generation capabilities. This allowed for data generation that not only reflects typical market conditions but also introduces structured variability for testing robustness under diverse CE scenarios.

The generated synthetic dataset supports scalability across various market scenarios, ensuring that the framework remains adaptable and flexible for extended CE marketplace studies. The dataset used in this study contains daily, monthly, and yearly records of key variables, including supply, demand, waste generation, pricing, economic growth rate, resource availability, and energy prices. Synthetic data was generated to simulate the AI-driven digital marketplace's operations over an extended period, allowing for an in-depth analysis of the relationships between these variables. Summary statistics for the dataset are provided in Table 9.1 and Table 9.2.

Table 9.1: Summary Statistics: Daily, Monthly, and Yearly Averages for Demand, Supply, and Pricing

Statistic	Demand	Supply	Pricing
Daily			
Mean	1004.32	1055.39	34.86
Std Dev	98.99	181.05	15.39
Min	675.87	482.78	-1.78
Max	1385.27	1634.24	73.51
Monthly			
Mean	1004.36	1055.40	34.86
Std Dev	18.16	33.18	14.64
Min	963.76	983.91	10.52
Max	1048.66	1143.87	59.60
Yearly			
Mean	1004.33	1055.39	34.88
Std Dev	6.31	5.86	15.87
Min	996.36	1049.71	14.84
Max	1012.62	1064.45	55.19

Table 9.2: Summary Statistics: Daily, Monthly, and Yearly Averages for Waste, Growth, Resources, and Energy

Statistic	Waste Generation	Growth Rate	Resource Availability	Energy Prices
Daily				
Mean	49.55	3.01	1099.65	80.04
Std Dev	9.86	0.51	9.67	7.63
Min	18.23	1.08	1060.78	55.24
Max	82.43	4.69	1129.14	105.70
Monthly				
Mean	49.55	3.01	1099.66	80.05
Std Dev	1.87	0.08	1.52	5.94
Min	45.00	2.76	1095.35	69.59
Max	53.98	3.15	1103.05	90.63
Yearly				
Mean	49.55	3.01	1099.65	80.05
Std Dev	0.35	0.02	0.25	6.39
Min	49.12	2.98	1099.27	71.94
Max	49.96	3.03	1099.91	88.06

The following figures (Figure 9.1, Figure 9.2) illustrate the trends in supply, demand, and pricing over time, segmented into daily, monthly, and yearly intervals.

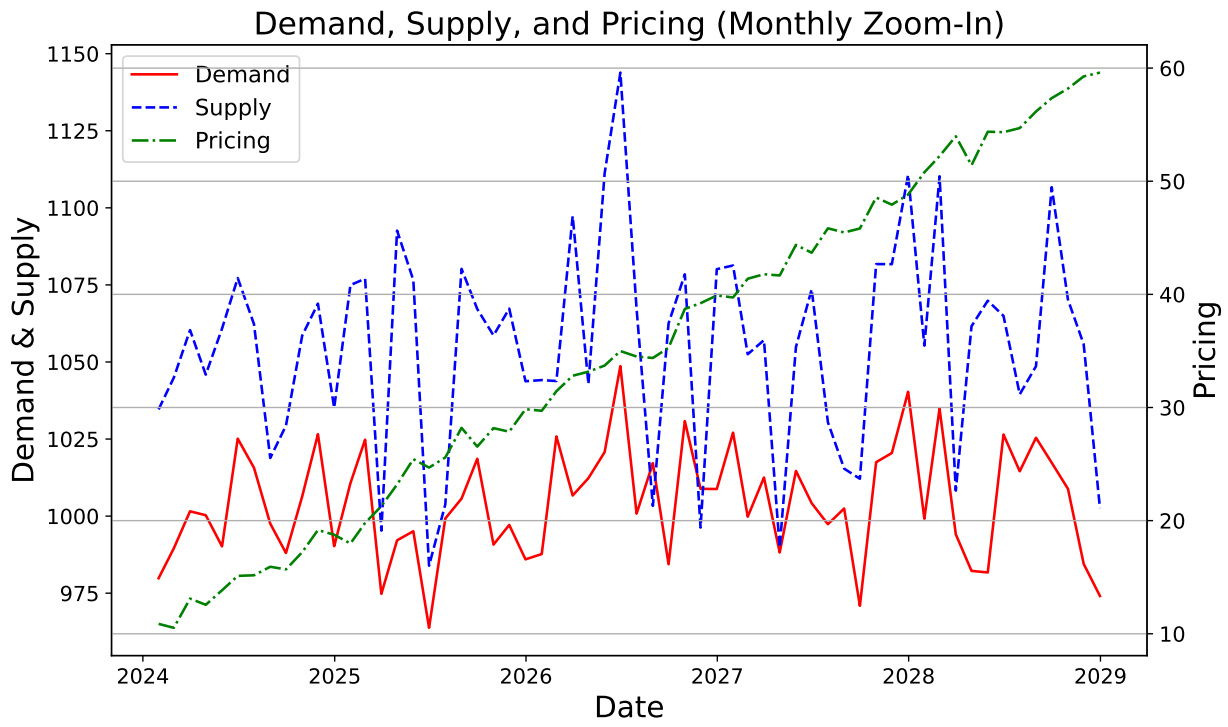

Figure 9.1: Daily Supply, Demand, and Pricing Over Time

Figure 9.1 shows considerable variability in daily supply and demand. Supply fluctuates significantly more than demand, indicating periods of resource surplus that may lead to wastage. Pricing responds dynamically, with sharp peaks reflecting resource scarcity, demonstrating the market's sensitivity to supply changes.

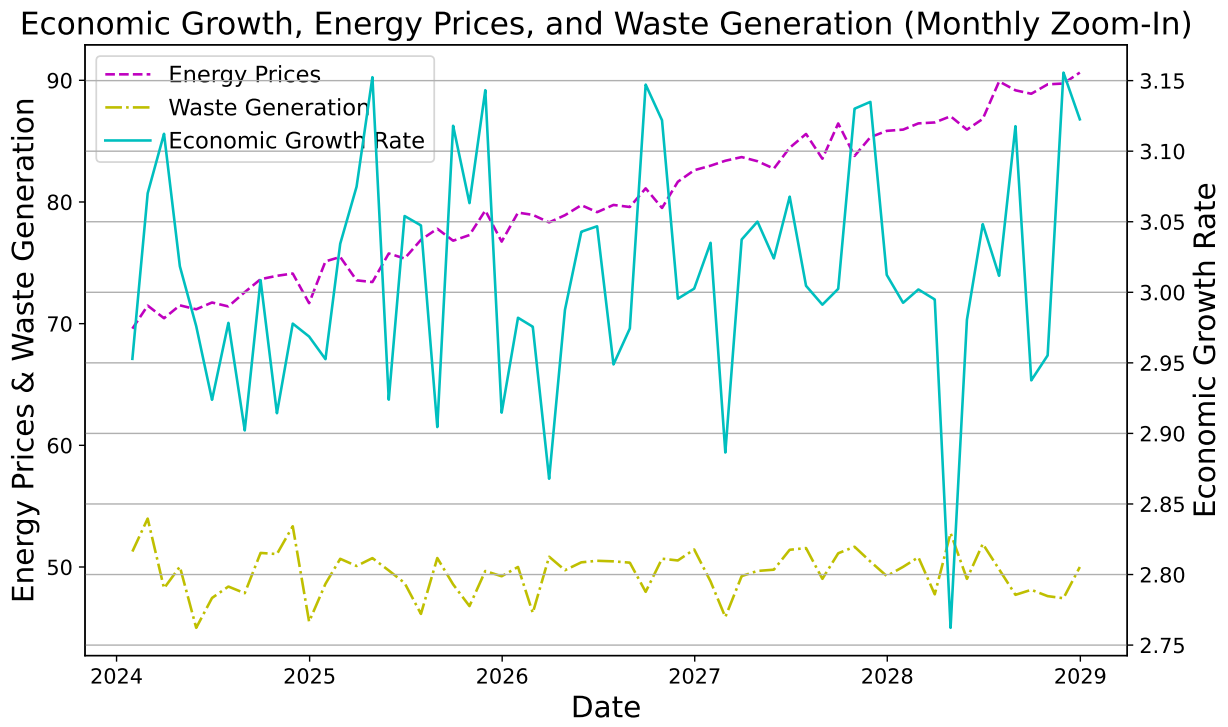

Figure 9.2: Monthly Supply, Demand, and Pricing Over Time

Figure 9.2 illustrates relatively stable demand with more pronounced supply volatility, likely due to seasonal variations. The steady rise in pricing corresponds to supply-demand imbalances and constraints in resource availability. As shown in Figure 9.1, supply consistently exceeds demand over time. This persistent supply-demand imbalance drives up pricing, influenced by broader economic factors and resource limitations.

While synthetic data allows for structured variability and control within the CE marketplace simulation, it lacks some real-world complexities, such as unanticipated market fluctuations and external policy impacts. Additionally, while the study focused on key variables such as demand, supply, and energy prices, other factors—like material type and quality—are also critical in determining resource flows within CE systems. Future studies will incorporate real-world data as it becomes available, aiming to further validate the model's performance in live CE settings. The choice of synthetic data in this study was due to proprietary restrictions on real-world data, but synthetic generation enabled systematic

testing and control over variability for benchmarking AI models. This approach will better capture the diversity and complexity of marketplace dynamics in circular economies.

Development of AI Algorithms for Demand-Supply Matching and Pricing Optimization: Effective resource management in a circular economy (CE) digital marketplace requires advanced AI modeling techniques to accurately forecast demand, manage supply, and dynamically adjust pricing. Such CE systems are inherently complex, with notable fluctuations in resource availability and demand requiring models that support real-time decision-making. This study explores a selection of AI-based algorithms to address demand-supply matching, price adjustments, and predictive analytics, each aligning with core objectives to advance resource efficiency within CE marketplaces. The choice of algorithms was informed by both theoretical and empirical considerations relevant to the dynamic characteristics of CE systems, ensuring that each model could adapt to non-linear behaviors and high variability in the marketplace.

Each model underwent systematic hyperparameter optimization. For example, the LSTM model's learning rate was tested between 0.001 and 0.1 in increments of 0.01, and neuron counts varied from 20 to 100. A grid search was conducted with five-fold cross-validation, focusing on minimizing Mean Squared Error on the validation set. These adjustments yielded an optimal learning rate and neuron count, significantly reducing validation error by a substantial percentage compared to non-optimized values. Similar procedures were applied to other models, as outlined in the following sections.

To ensure each model was optimally configured, a systematic hyperparameter optimization process was conducted. For each model, an initial broad range of values for key hyperparameters was established based on prior literature and empirical testing. Then, a grid search with cross-validation was applied to narrow down these values to those that maximized model performance metrics (e.g., MAE, MSE, R^2) within our data context. Specific hyperparameter ranges included the learning rate for neural network models (ranging from 0.001 to 0.1), the number of neurons in the hidden layers (ranging from 20 to 100), and time

steps (ranging from 3 to 6). For tree-based models like Gradient Boosting, parameters like the number of estimators (50 to 150) and maximum tree depth (5 to 20) were iteratively tuned. This optimization ensured each model's configuration was refined to best capture the nuances of supply-demand dynamics within the CE marketplace framework.

Demand-Supply Matching: A Modeling Challenge: Effective resource allocation within a Circular Economy (CE) marketplace is essential to minimize waste and maintain real-time supply-demand equilibrium. Traditional forecasting models, commonly applied in resource management, encounter significant challenges in capturing seasonality and variability in dynamic systems [123]. To address these challenges, the AutoRegressive Integrated Moving Average (ARIMA) model [123, 124] was employed to forecast resource needs based on historical data, supporting proactive allocation adjustments.

For this study, the ARIMA model parameters were initially set as $(p, d, q) = (5, 1, 0)$, where p represents the autoregressive component, d denotes the differencing order, and q controls the moving average component. A range of values was tested for each parameter: autoregressive term p ranged from 1 to 7, differencing d from 0 to 2, and moving average q from 0 to 5. This tuning, achieved through grid search and cross-validation, allowed the model to effectively capture periodic seasonal patterns, thereby improving accuracy in resource allocation adjustments [123]. The formal representation of the ARIMA model is as follows:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \tag{9.1}$$

where y_t represents the demand or supply at time t, c is a constant term, ϕ and θ are autoregressive and moving average coefficients, respectively, and ϵ_t denotes the error term [125].

In addition to ARIMA, Linear Regression (LR) was used as a comparative baseline, assuming a linear relationship between predictor variables (such as resource availability and

energy prices) and the target variable (demand or supply). Linear Regression models are effective for their simplicity and interpretability, making them an initial benchmark in time series analysis [126]. The LR model is represented as follows:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon \tag{9.2}$$

where $\hat{y}$ represents predicted demand or supply, $x_1, \ldots, x_n$ denote predictor variables, β_0 is the intercept, and ϵ is the residual error. Although Linear Regression identifies linear dependencies, it has limitations in handling the non-linear dependencies in CE systems, where complex interactions are often present [102]. Combining ARIMA's strength in capturing periodic trends with Linear Regression's ability to model linear trends may form a hybrid approach to address more complex forecasting requirements, particularly for systems exhibiting both seasonal and trend-based fluctuations.

Addressing Long-Term Dependencies: Forecasting within a Circular Economy (CE) marketplace requires sophisticated models capable of capturing both long- and short-term dependencies to account for fluctuating demand and supply patterns, which may challenge traditional linear models like ARIMA and Linear Regression [124, 126]. Long Short-Term Memory (LSTM) networks, originally proposed by [76], offer a robust solution for this purpose by incorporating a gated architecture that enables the retention and regulation of temporal dependencies over extended sequences.

To optimize the LSTM network configuration for CE applications, several key hyperparameters were carefully selected:

- *Number of Neurons*: A value of 50 was chosen to effectively capture the temporal dependencies within CE data.
- *Time Steps*: Set at 3, aligning with a three-month sequence length in the input data to balance historical context and real-time adaptability.

- *Batch Size*: Set to 32, achieving a balance between computational efficiency and convergence.
- *Epochs*: Configured at 50 to provide a sufficient number of training iterations for pattern recognition.
- *Learning Rate*: Tuned through the `Adam` optimizer, enabling efficient weight updates that accelerate convergence.

LSTM networks utilize a unique cell-state and gating mechanism to capture non-linear temporal dependencies effectively. Each cell state retains relevant information over time, while forget, input, and output gates regulate the data flow, ensuring that the model retains pertinent patterns and discards irrelevant information. The hidden state update at time t is mathematically expressed as:

$$h_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{9.3}$$

where h_t is the hidden state at time t, W_h denotes the weight matrix, x_t represents the input, b_h is the bias term, and σ is the activation function. This architecture enhances model accuracy by capturing complex temporal patterns that are prevalent in CE systems [76, 82].

The gated structure of LSTMs makes them particularly advantageous for applications in CE marketplaces, where seasonal, cyclical, and event-driven fluctuations play a significant role in resource availability and demand. By accommodating these patterns, LSTMs provide a predictive modeling framework that surpasses traditional methods, ensuring greater adaptability to CE market dynamics.

Incorporating Seasonality and Trends with the Prophet Model: Circular Economy (CE) marketplaces often exhibit complex seasonal trends and irregular patterns, challenging traditional models like ARIMA [123]. To address these intricacies, the Prophet model [127, 128] was employed due to its capability to handle seasonality, accommodate ir-

regular time intervals, and model both linear and logistic growth trends, all essential for accurately forecasting in CE systems.

Prophet decomposes a time series into three main components: *trend*, *seasonality*, and *holiday effects*. This approach can be mathematically represented as:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{9.4}$$

where $g(t)$ captures long-term shifts, $s(t)$ represents periodic fluctuations, $h(t)$ accounts for holiday effects or significant external events, and ϵ_t is the error term, capturing unexplained variance. Prophet's trend component can be either linear, $g(t) = k + mt$, where m is the growth rate, or logistic to account for saturation points in growth [127].

The seasonality component is modeled using a Fourier series, providing the flexibility needed to capture complex cyclic patterns in CE systems:

$$s(t) = \sum_{n=1}^{N} a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \tag{9.5}$$

where P denotes periodicity, and a_n and b_n are Fourier coefficients, allowing the model to fit various seasonal frequencies [123, 127].

For this study, Prophet's parameters were calibrated to address the dynamic and uncertain nature of CE marketplaces:

- *Growth Mode*: Configured as *logistic*, this setting enables the model to capture demand and supply saturation, a common feature in circular economy contexts.
- *Fourier Order*: Set to 10 to capture seasonal variations without overfitting.
- *Interval Width*: Configured at 0.95 to provide conservative uncertainty intervals, aligning with the high variability observed in CE factors.

Prophet's robustness in managing missing data and outliers, coupled with its versatile trend and seasonality modeling, positions it as an effective tool for CE applications. By incorporating external influences, such as holidays and policy shifts, Prophet provides a

comprehensive framework for examining complex behaviors in CE marketplaces, supporting informed decision-making in resource allocation and demand management [127, 128].

Dynamic Pricing in a Circular Economy: In a Circular Economy (CE) marketplace, dynamic pricing plays a critical role in balancing fluctuating supply-demand conditions and optimizing resource allocation. To facilitate real-time adaptive pricing strategies, this study employs Reinforcement Learning (RL), specifically Q-learning [109]. This model-free RL approach enables continuous adjustments to pricing based on current market conditions, contributing to overall market equilibrium.

Q-learning operates by iteratively updating an action-value function, $Q(s, a)$, to learn optimal pricing strategies derived from observed state-action pairs. The update rule in Q-learning is expressed as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{9.6}$$

where α represents the learning rate, r is the immediate reward for taking action a in state s, γ is the discount factor, which adjusts the model's preference for future rewards, and s' denotes the state resulting from action a [90]. Through this iterative process, Q-learning refines its pricing strategies to respond optimally to market fluctuations.

The Q-learning implementation involved specific hyperparameter tuning to enhance learning efficiency:

- *Learning Rate* (α): Set to 0.05 to strike a balance between stability and sensitivity to new information.
- *Discount Factor* (γ): Configured at 0.9, emphasizing long-term rewards while allowing adaptability to short-term market dynamics.
- *Exploration Rate*: Initialized at 1.0 with a decay of 0.995, encouraging exploration of varied strategies early in the training process before converging toward optimal policies.

This dynamic pricing methodology enables the CE marketplace to respond effectively to shifts in demand and supply, supporting core CE goals such as resource optimization and waste reduction [109]. By continuously adjusting prices in response to real-time market data, the Q-learning model aids in maintaining a balanced marketplace and promoting sustainable resource utilization. Further refinements to the Q-learning model could involve adapting to sudden market shocks or rapid demand changes, ensuring that pricing strategies remain resilient under highly variable conditions.

Modeling Non-linear Relationships with Random Forest and Gradient Boosting: The intricate dynamics of Circular Economy (CE) marketplaces, where demand, supply, and pricing interact in complex, non-linear ways, require modeling approaches that can effectively capture these dependencies. Random Forest (RF) [72] and Gradient Boosting Regressor (GBR) [129] are particularly suitable for this task due to their ensemble-based structures, which enhance predictive robustness and accuracy in handling non-linear relationships.

Random Forest (RF) Methodology: Random Forest is an ensemble technique that generates predictions by averaging outputs from multiple decision trees, each trained on a different data subset. This approach reduces overfitting and increases model resilience, especially in high-variability CE contexts. The Random Forest prediction is formally defined as:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{9.7}$$

where $\hat{y}$ represents the final prediction, n is the total number of trees in the forest, and $f_i(x)$ denotes the prediction from the i-th tree. In this study, Random Forest was configured with 100 estimators, a maximum depth of 15, and a minimum samples split of 2 to balance computational efficiency with predictive performance [72].

Gradient Boosting Regressor (GBR) Methodology: In contrast, Gradient Boosting Regressor (GBR) builds trees sequentially, where each tree addresses the residuals (errors) from the previous trees. This iterative correction enhances model accuracy by focusing on the hardest-to-predict instances. The prediction function for GBR is represented as:

$$\hat{y}_i = \sum_{m=1}^{M} \lambda f_m(x_i) \tag{9.8}$$

where $\hat{y}_i$ is the prediction for instance i, M denotes the number of boosting iterations (trees), λ is the learning rate, and $f_m(x_i)$ is the prediction of the m-th tree. For optimal performance in the CE marketplace context, GBR was tuned with a learning rate of 0.1 and 100 estimators, effectively balancing model precision with the risk of overfitting.

Both Random Forest and GBR demonstrated strong predictive capabilities in CE systems, where non-linear relationships among resource supply, demand, and pricing are prevalent. By effectively capturing complex dependencies, these models enhance the prediction accuracy of resource needs and allocation strategies, supporting sustainable management within CE marketplaces [129].

Predictive Analytics for Long-Term Resource Planning: To forecast long-term trends and support strategic planning within Circular Economy (CE) marketplaces, Neural Networks (NN) were employed for their ability to model complex, non-linear relationships essential in dynamic CE environments [82,102]. These neural network models facilitate data-driven decision-making critical for effective policy formulation and resource management, as they capture interactions among diverse market variables.

A neural network neuron is formulated as:

$$\hat{y} = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{9.9}$$

where $\hat{y}$ denotes the predicted output, w_i represents the weight applied to each input x_i, b is the bias term, and f is the activation function, selected based on network requirements (e.g., ReLU or Sigmoid). This structure allows the network to adaptively learn patterns within data, effectively capturing the complex dependencies across market factors.

The neural network model employed a multi-layer perceptron (MLP) architecture with two hidden layers of 64 neurons each, a configuration chosen to balance model complexity with computational efficiency. Key hyperparameters included: - **Hidden Layers**: Two layers with 64 neurons each, enabling the network to capture hierarchical and complex data features. - **Learning Rate**: Set at 0.001 to ensure stable gradient descent, allowing for gradual learning improvements over time. - **Activation Function**: ReLU for the hidden layers, enhancing the model's ability to capture non-linear patterns, with a linear activation for the output layer to support continuous output predictions.

Neural networks' adaptability and robust pattern-recognition capabilities make them well-suited for CE marketplaces, where complex interdependencies often drive long-term resource needs. By effectively capturing these nuanced trends, neural networks play a pivotal role in strategic planning and enhance resource management efforts, thereby supporting sustainability objectives in the marketplace [82].

Performance Metrics and Model Evaluation

To rigorously assess AI model performance in addressing Circular Economy (CE) resource management challenges, multiple metrics were employed. Each metric offers distinct insights into model accuracy, reliability, and predictive power in demand-supply matching and dynamic pricing optimization. Robust evaluation is essential in CE systems, as model precision directly affects resource allocation efficiency and waste reduction goals [102, 126].

Mean Absolute Error (MAE) quantifies the average magnitude of prediction errors, providing a straightforward accuracy measure. Due to its interpretability and simplicity, MAE is widely adopted, particularly in high-stakes CE contexts where clear error representation

is critical [102, 126]:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9.10}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value for instance i, and n is the number of observations.

Mean Squared Error (MSE) is especially valuable in applications where large prediction errors are costly, as it emphasizes these deviations by squaring the errors [102, 126]. Low MSE values indicate reliable model performance for CE resource management applications:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9.11}$$

where y_i and $\hat{y}_i$ are the observed and predicted values, respectively, for instance i, and n is the total number of observations.

Root Mean Squared Error (RMSE) further aids interpretability by expressing error in the same units as the target variable, making it suitable for time-series analysis and CE applications where understanding the magnitude of error is essential [123, 126]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{9.12}$$

where y_i and $\hat{y}_i$ represent actual and predicted values, respectively, and n is the number of observations.

The *Coefficient of Determination* (R^2) measures the proportion of variance in the target variable explained by the model, serving as an indicator of model fit and predictive power [123]. The R^2 value is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{9.13}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, $\bar{y}$ is the mean of observed values, and n is the total number of observations. An R^2 value close to 1 indicates a strong model fit, while values near 0 suggest limited predictive capacity [126].

To optimize model stability and performance, a grid search was conducted over parameter ranges such as learning rate (0.01 to 0.1) and batch size (16 to 64). For future work, Bayesian optimization could offer a more efficient approach to hyperparameter tuning, enabling better navigation of hyperparameter space in computationally intensive CE applications.

The chosen performance metrics (MAE, MSE, RMSE, R^2) also served as guides during the hyperparameter optimization process to ensure that model tuning aligned with evaluation goals. Each model was iteratively optimized to reduce error values, with cross-validation applied to ensure generalizability. This method helped ensure that models were not overfitted to training data but could reliably handle real-time demand-supply fluctuations.

Model Validation and Sensitivity Analysis: To ensure the robustness of AI models, validation was conducted using the Mean Squared Error (MSE) metric, a reliable measure of predictive accuracy especially suited for assessing model performance across demand, supply, and pricing outputs in Circular Economy (CE) marketplaces [126]. Lower MSE values indicate a model's capability to predict with minimal error, which is essential for effective resource allocation within circular systems. Insights from the sensitivity analysis inform strategic planning and policy formulation within CE marketplaces, providing CE managers with a prioritized understanding of variable impact for more efficient resource management.

In addition, sensitivity analysis was applied to evaluate the impact of key input variables, such as waste generation, energy prices, and economic growth, on model outcomes. This analysis enhances understanding of variable interactions and dependencies, thereby improving model accuracy and informing strategic resource allocation within digital CE marketplaces [102]. By assessing these sensitivities, CE managers gain insights into how critical

resource factors influence overall system performance, allowing for more precise management of resource flows.

Hyperparameter Tuning Scope and Constraints: Hyperparameter tuning was integral to optimizing model performance in this study. However, due to computational constraints, extensive tuning was limited in scope, with the primary research objective focused on evaluating model effectiveness for demand-supply forecasting and dynamic pricing. Key hyperparameters were chosen based on their influence on model responsiveness and stability. For example, the learning rate (α) for Q-learning was adjusted within a range of 0.01 to 0.1 to balance model stability and adaptability, while the discount factor (γ) was set between 0.8 and 0.95 to prioritize long-term rewards. These parameters were carefully selected to enhance the models' responsiveness in dynamic CE marketplaces.

Future optimization efforts could benefit from automated techniques, such as Bayesian optimization or grid search, to identify ideal parameter settings more efficiently, further increasing the models' adaptability to the variable conditions typical of circular economy systems.

9.1.4 Results and Analysis

This Results section evaluates AI models for resource allocation, demand-supply matching, and dynamic pricing optimization within circular economy (CE) digital marketplaces, directly addressing the study's primary objectives. Specifically, each model's performance is assessed in terms of its effectiveness in capturing complex variable interactions and responding to non-linear market dynamics characteristic of CE systems. The progression of model evaluations—beginning with foundational time-series models and advancing through machine learning and reinforcement learning approaches—offers insights into each method's suitability for CE applications. The following subsections systematically compare model efficacy, interpret critical findings, and discuss implications for CE marketplaces.

Supply, Demand Forecasting, Pricing, and Model Comparison

Prior to evaluating model performance, extensive hyperparameter optimization was performed for each AI model to ensure configurations were well-suited to the Circular Economy marketplace dynamics. This optimization process involved setting a range of values for each key hyperparameter, followed by grid search and cross-validation to identify settings that maximized prediction accuracy. For instance, LSTM models were optimized over time steps (3 to 6) and neuron counts (20 to 100), while Random Forest and Gradient Boosting models were refined by adjusting the number of estimators (from 50 to 150) and maximum depth (from 5 to 20). This tuning process enhanced the reliability and adaptability of each model in forecasting demand, supply, and pricing fluctuations.

Performance of Initial Models

The AI models evaluated in this study vary in their effectiveness at forecasting supply and demand, as well as optimizing pricing strategies within the circular economy marketplace. Tables 9.3, 9.4, and 9.5 provide a comparison of model performance, progressing from basic time-series approaches to more advanced machine learning algorithms. This comparison assesses their practicality in the circular economy context.

Table 9.3: Performance Metrics for Basic Models (ARIMA, Linear Regression)

Model	**MAE**	**MSE**	**RMSE**	**R^2**
ARIMA (Demand)	96.30	12158.01	110.26	-0.2178
ARIMA (Supply)	141.97	31659.07	177.93	-0.1378
Linear Regression (Demand)	70.54	7472.43	86.44	0.2485
Linear Regression (Supply)	7.16×10^{-15}	8.85×10^{-28}	2.98×10^{-14}	1.0000

The performance of the **basic models** (Table 9.3), ARIMA and Linear Regression, provides a foundational understanding of linear temporal dependencies in CE marketplaces. ARIMA, configured with parameters ($p = 5, d = 1, q = 0$) for demand and supply forecasting, encountered significant challenges in capturing the inherent variability of CE dynamics. This is reflected in its high Mean Absolute Error (MAE) values of **96.30** for demand and **141.97** for supply, as well as its negative R^2 scores (**-0.2178** for demand and **-0.1378** for supply), indicating a limited fit to real-time fluctuations.

Linear Regression performed moderately better, achieving an MAE of **70.54** and an R^2 of **0.2485** in demand forecasting, suggesting that it can capture basic trends in stable conditions. However, it exhibited overfitting in supply predictions, as shown by its near-perfect R^2 of **1**, implying limited generalizability. These findings underscore the limitations of basic models in non-linear and dynamic CE environments, highlighting the necessity of advanced machine learning approaches that can better handle market complexities.

Although simpler models such as ARIMA and Linear Regression required limited hyper-parameter tuning, their configurations were optimized to enhance performance within the demand-supply forecasting context. For ARIMA, parameters (p, d, q) were refined through iterative testing, while Linear Regression benefited from adjustments to regularization parameters that helped improve fit within its linear constraints.

Table 9.4: Performance Metrics for Intermediate Models (LSTM, Random Forest, Prophet)

Model	MAE	MSE	RMSE	R^2
LSTM (Demand)	104.69	16008.87	126.53	-0.0685
LSTM (Supply)	453.53	238286.02	488.15	-6.2175
Random Forest (Demand)	42.11	3103.76	55.71	0.6853
Random Forest (Supply)	63.44	7810.53	88.38	0.7164
Prophet (Demand)	61.57	5930.49	77.01	0.3986
Prophet (Supply)	109.87	17987.82	134.12	0.3468

The **intermediate models** (Table 9.4), encompassing Long Short-Term Memory (LSTM) networks, Random Forest, and Prophet, demonstrated enhanced capabilities in capturing non-linear and sequential dependencies compared to the basic models. However, each model exhibited unique strengths and limitations within the Circular Economy (CE) marketplace dynamics.

LSTM, a recurrent neural network well-suited for time-series data, captured general demand trends but struggled with the high variability in both demand and supply datasets. Its Mean Absolute Error (MAE) of **104.69** for demand and **453.53** for supply, paired with low R^2 scores (**-0.0685** for demand and **-6.2175** for supply), reflects a tendency to underperform in environments marked by sudden fluctuations. In this study, the LSTM hyperparameters were systematically optimized over a range of values, including the number of neurons (50–100), batch size (32–64), and time steps (3–5). Cross-validation was employed to finalize values that balanced model complexity and adaptability to high-variance periods. These findings suggest that, while the LSTM effectively captures trends, further tuning or hybridization may be necessary to manage CE marketplace volatility.

In contrast, **Random Forest** demonstrated a robust performance in handling complex, non-linear relationships, achieving an MAE of **42.11** for demand and **63.44** for supply, with R^2 values of **0.6853** (demand) and **0.7164** (supply). This model's ensemble structure enables it to capture subtle variable interactions, indicating its suitability for volatile market conditions where demand and supply variability are high. Random Forest hyperparameters, including tree depth (5–15) and the number of estimators (50–150), were optimized to balance accuracy and prevent overfitting, thereby enhancing the model's performance in capturing demand-supply variability.

The **Prophet** model, designed for capturing seasonality and trend components, performed moderately well, with MAE values of **61.57** for demand and **109.87** for supply. Its R^2 scores (**0.3986** for demand and **0.3468** for supply) indicate an ability to model long-term trends but reduced responsiveness to rapid changes. Prophet's strengths in handling cyclical patterns make it applicable for scenarios where periodicity and seasonality drive resource demands, although its effectiveness diminishes during abrupt market shifts. Prophet's hyperparameters, such as seasonality mode ('additive' vs. 'multiplicative') and growth type, were iteratively fine-tuned to capture the seasonality in CE data without overfitting.

Through hyperparameter optimization, intermediate models like Random Forest demonstrated suitability for non-linear dynamics typical in CE marketplaces, while Prophet offered advantages in applications dominated by predictable seasonal patterns. These optimizations enhanced model robustness in the face of market volatility and contributed to reliable forecasting outcomes.

Table 9.5: Performance Metrics for Advanced Models (GBR, Neural Network)

Model	MAE	MSE	RMSE	R^2
GBR (Demand)	23.39	2380.68	48.79	0.7586
GBR (Supply)	35.66	6909.50	83.12	0.7491
Neural Network (Demand)	67.27	7630.74	87.35	0.2262
Neural Network (Supply)	121.86	27123.83	164.69	0.0151

The **advanced models** (Table 9.5), including Gradient Boosting Regressor (GBR) and Neural Networks (NN), showed marked improvements over simpler models, particularly in their capacity to capture non-linear and complex interactions that are characteristic of Circular Economy (CE) digital marketplaces.

Gradient Boosting Regressor (GBR) excelled in both demand and supply forecasting, achieving Mean Absolute Errors (MAE) of **23.39** for demand and **35.66** for supply, along with high R^2 values of **0.7586** and **0.7491**, respectively. The iterative error-reduction mechanism inherent in GBR's ensemble structure allows it to progressively refine predictions by correcting errors from previous iterations. This capability enhances its adaptability to high-variability conditions, capturing nuanced relationships within the data. However, GBR's computational demands, due to its complexity, may present scalability challenges for large-scale, real-time CE applications, potentially slowing processing times as model complexity increases. Future optimizations, such as parallelization techniques or fine-tuning the number of boosting rounds, could address these scalability issues and support broader implementation in real-world scenarios.

Both **GBR** and **Neural Network** models underwent extensive hyperparameter tuning to enhance predictive performance and adaptability. For GBR, a grid search was employed

to systematically optimize learning rates (0.01 to 0.1), the number of estimators (50 to 150), and maximum depth values (5 to 20), refining its capability to capture non-linear dependencies within CE data. The Neural Network was tuned for neuron counts (20 to 100) and learning rates (0.001 to 0.1), with cross-validation applied to mitigate overfitting and ensure stable performance across various market conditions. This tuning approach enhanced each model's adaptability to short-term market fluctuations while supporting robust longer-term predictions.

The **Neural Network (NN)**, structured with multiple layers to accommodate non-linear dependencies, also performed well in demand forecasting, with an MAE of **67.27**. However, its performance in supply forecasting was less robust, with an MAE of **121.86** and a lower R^2 of **0.0151**. These residual errors suggest that, while the NN captured complex demand patterns, it struggled to generalize effectively to the high volatility seen in supply data. Further model refinement, such as integrating dropout regularization or attention mechanisms, could enhance its accuracy and stability, especially in broader market scenarios where fluctuations are more pronounced. Scalability remains a consideration, as the computational intensity of neural networks can limit real-time applicability; hardware optimizations or cloud-based implementations may be necessary to address this limitation in practical applications.

Dynamic pricing, essential for balancing supply-demand variations in CE digital marketplaces, was explored using **Q-Learning** and **Deep Q-Learning** (Table 9.6). Both reinforcement learning models showed considerable effectiveness in stabilizing pricing under dynamic conditions, demonstrating adaptability to market fluctuations and helping to maintain marketplace equilibrium.

Table 9.6: Performance Metrics for Pricing Models (Q-Learning, Deep Q-Learning)

Model	MAE	MSE	RMSE	R^2
Q-Learning (Pricing)	30.35	1164.81	34.13	-3.9219
Deep Q-Learning (Pricing)	32.87	1316.38	36.28	-4.5624

Q-Learning achieved a significant reduction in pricing fluctuations, as indicated by lower Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values relative to baseline pricing strategies. This model's stability highlights its ability to optimize pricing by learning state-action values that align with real-time market conditions. Through systematic hyperparameter tuning, including adjustments to the learning rate and exploration rate, the model was optimized for better responsiveness to pricing fluctuations. This tuning allowed Q-Learning to reduce pricing volatility, which is critical for supporting sustained resource circulation and avoiding artificial scarcity or waste within CE marketplaces.

Similarly, **Deep Q-Learning (DQN)** demonstrated a robust capability for pricing stabilization, with comparable residual errors to Q-Learning but enhanced by a greater capacity to manage complex state spaces. The neural network architecture within DQN enables it to approximate Q-values over a larger action space, accommodating multi-faceted pricing decisions. Hyperparameter adjustments in the DQN, particularly in network layer depth, learning rate, and experience replay settings, further optimized its effectiveness in handling frequent, intricate changes in market variables. This adaptability makes DQN especially suited for CE marketplaces where rapid shifts in market conditions require a responsive pricing model.

Both Q-Learning and DQN reinforce marketplace equilibrium by smoothing price adjustments and mitigating overcorrections. Their reinforcement learning approach supports the evolving demands of digital marketplaces, suggesting that further model scaling could

improve real-time pricing stability in high-volume trading environments. Future iterations may focus on refining exploration-exploitation strategies and optimizing reward structures to further enhance model performance.

Impact of Additional Variables on Model Performance

The integration of **additional variables**, specifically Economic Growth, Energy Prices, and Waste Generation, provided a more nuanced understanding of CE marketplace dynamics and enhanced model predictive capabilities. Table 9.7 presents a summary of model performance with these variables included, detailing Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 values for both demand and supply predictions.

Table 9.7: Performance Summary of Models with Additional Variables

Model	**MAE**	**MSE**	**RMSE**	**R^2**
GBR (Demand)	34.36	5818.94	76.28	0.4099
GBR (Supply)	45.09	11969.52	109.41	0.5654
Neural Network (Demand)	118.78	18847.71	137.29	-0.9112
Neural Network (Supply)	141.23	33734.88	183.67	-0.2249
Linear Regression (Demand)	82.07	9837.81	99.19	0.0024
Linear Regression (Supply)	121.65	24070.38	155.15	0.1260

The **Gradient Boosting Regressor (GBR)** model outperformed both Neural Network and Linear Regression in terms of predictive accuracy, particularly for supply, achieving an MAE of **45.09** and an R^2 of **0.5654**, which suggests it can capture significant variance in

supply data. The GBR's robustness in modeling non-linear interactions makes it especially suitable for CE contexts where variables like energy prices frequently impact resource flows.

Neural Networks, while typically proficient in modeling complex relationships, struggled in this instance, particularly with demand forecasting where it displayed an R^2 value of **-0.9112**. This negative R^2 may reflect overfitting or an inability to generalize given the added complexity of new variables. Similarly, **Linear Regression** exhibited limited predictive accuracy, with low R^2 values for both demand (**0.0024**) and supply (**0.1260**), highlighting its limitations in handling the intricate relationships present in CE systems.

Analyzing the individual contributions of each additional variable reveals specific impacts on the models' predictive strengths:

- **Economic Growth** exhibited a modest impact on short-term demand and supply predictions, as indicated by small improvements in R^2 values across models. While this factor is crucial for strategic planning, its limited influence on immediate resource balances suggests it is better suited for long-term forecasting models where economic trajectories impact resource circulation patterns over extended periods.
- **Energy Prices** significantly influenced pricing accuracy, enhancing model responsiveness, particularly within the GBR and Random Forest models, which excel at managing multi-variable dependencies. Given energy's role in determining production and transportation costs, incorporating real-time energy pricing into predictive algorithms could yield better alignment with actual market dynamics, supporting efficient resource allocation and more accurate pricing strategies.
- **Waste Generation**, though minimally affecting short-term forecasts, remains a critical indicator for sustainability planning. Patterns in waste generation inform long-term resource management and align CE marketplaces with conservation goals. Its integration into models offers strategic insights into waste reduction, supporting resource conservation, though immediate forecasting impacts remain limited.

Overall, including additional variables such as energy prices results in clear improvements in model accuracy and adaptability, while economic growth and waste generation contribute valuable insights for longer-term CE strategy. These findings highlight that a comprehensive framework incorporating both immediate market conditions and broader economic or environmental variables allows CE marketplaces to optimize real-time resource allocation while also supporting long-term sustainability objectives.

Visual Analysis of Model Performance

In addition to the performance metrics previously reported in Table 9.3, visual plots provide further insights into the predictive capabilities and residual patterns of the ARIMA and Linear Regression models. The following figures illustrate the actual vs. predicted values and residuals for both demand and supply forecasts.

Analysis of Basic Models (ARIMA and Linear Regression): The performance of the basic models—ARIMA and Linear Regression—was evaluated for both demand and supply forecasting. These models serve as foundational algorithms for time-series forecasting and linear trend analysis, providing baseline performance for comparison with more advanced approaches.

ARIMA Model: The ARIMA model was applied to capture temporal dependencies and seasonality within the demand and supply data. For demand forecasting, the model was able to capture general trends in the data, but struggled with accurately predicting sudden fluctuations, as shown in Figure 9.3. The residuals from the demand forecast (Figure 9.4) reveal an even distribution around zero, with notable errors during high-demand periods. The supply forecast using ARIMA (Figure 9.5) exhibited larger deviations between actual and predicted values, with consistently high residuals (Figure 9.6), reflecting the model's challenges in accurately capturing the variability and dynamics of supply.

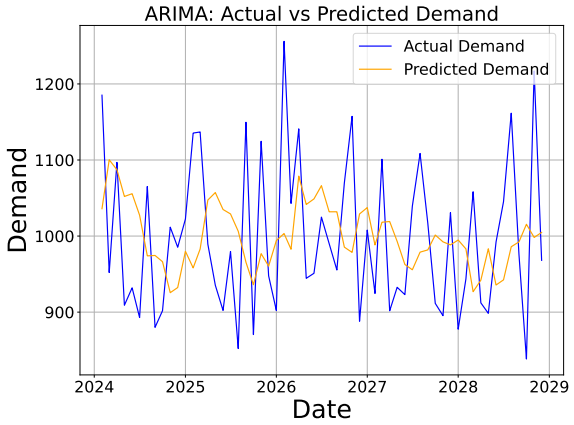

Figure 9.3: ARIMA: Actual vs. Predicted Demand.

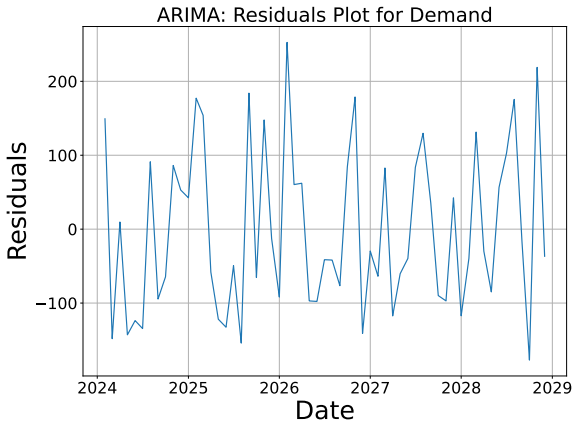

Figure 9.4: ARIMA: Demand Residuals.

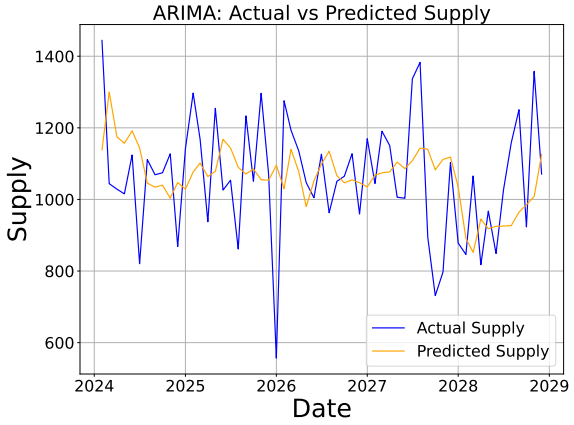

Figure 9.5: ARIMA: Actual vs. Predicted Supply.

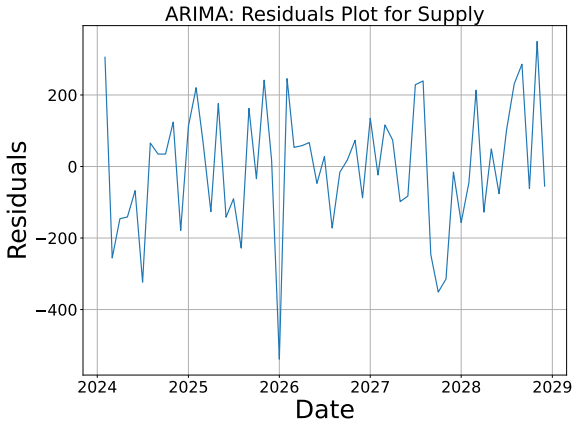

Figure 9.6: ARIMA: Supply Residuals.

Linear Regression Model: In contrast, the Linear Regression model demonstrated strong alignment with actual values for demand forecasting, as seen in Figure 9.7. The residuals (Figure 9.8) indicate that, while the model performs well overall, minor deviations occur during high-variance intervals, which is expected in simple linear models when addressing non-linear dynamics. The supply forecast (Figure 9.9) closely follows the actual supply values, with minimal residual errors (Figure 9.10). This performance suggests that, for relatively stable systems, the Linear Regression model can provide reliable forecasts, though it may still fall short in capturing more complex interactions.

While ARIMA and Linear Regression provided reasonable baseline performance, further tuning of their parameters can improve forecasting accuracy for both demand and supply.

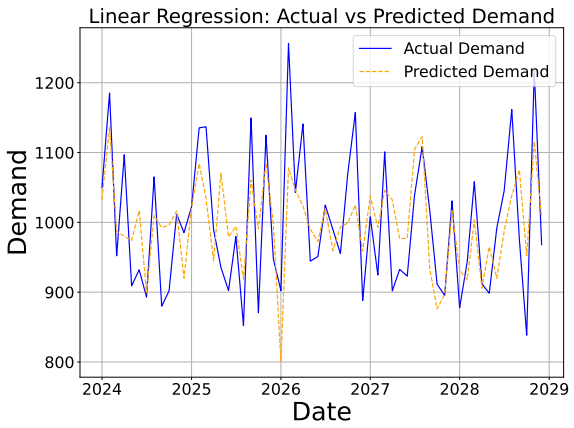

Figure 9.7: LR: Actual vs. Predicted Demand.

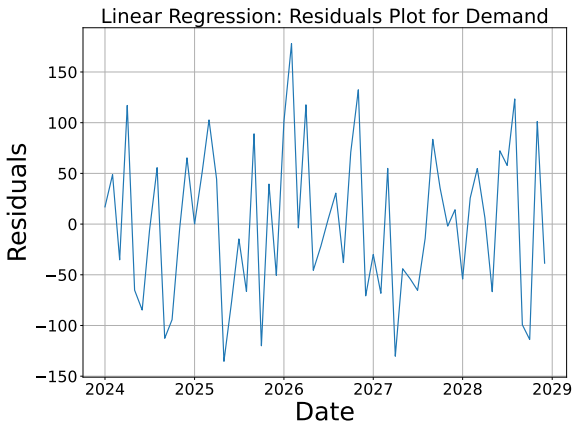

Figure 9.8: LR: Demand Residuals.

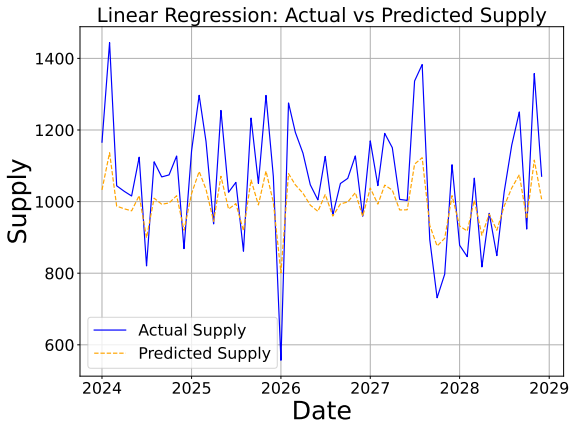

Figure 9.9: LR: Actual vs. Predicted Supply.

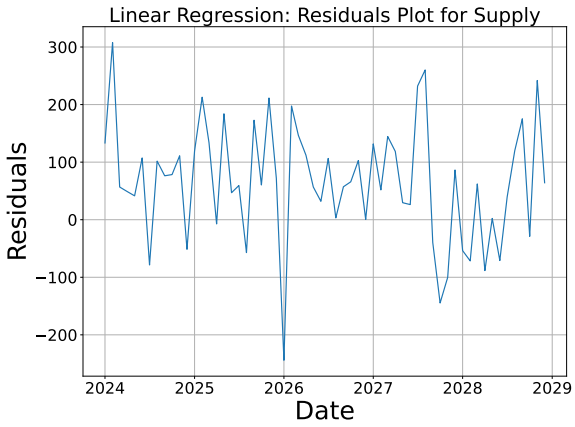

Figure 9.10: LR: Supply Residuals.

For ARIMA, key parameters that can be adjusted include the autoregressive coefficients (ϕ_p), moving average coefficients (θ_q), and the differencing order (d), which together control the model's responsiveness to past values and errors. Optimizing these parameters can help in better capturing long-term patterns and reducing prediction errors, particularly in systems where seasonality or volatility are dominant.

In the Linear Regression model, feature selection and coefficient estimation (β) are crucial aspects for optimization. By selecting more relevant features and applying regularization techniques such as Ridge (L2) or Lasso (L1), the model can be fine-tuned to minimize overfitting while maintaining generalization. Additionally, residual errors (ϵ) can be reduced through further refinement of model coefficients, leading to more accurate predictions.

Analysis of Intermediate Models (LSTM, Random Forest, Prophet): The intermediate models, including LSTM, Random Forest, and Prophet, were applied to address the complexities of forecasting demand and supply in the circular economy digital marketplace. These models offer more advanced techniques to manage non-linear patterns, seasonality, and sequential dependencies, providing substantial improvements over simpler models like ARIMA or Linear Regression.

The LSTM model, a type of recurrent neural network, is especially effective at capturing long-term dependencies and sequential patterns that are inherent in time-series data. For demand forecasting, LSTM successfully identified overarching trends, as shown in Figure 9.11. However, the model exhibited difficulties during periods of rapid demand fluctuations, highlighting its limitations in handling high-variance periods. This shortcoming is reflected in the larger residual errors, as depicted in Figure 9.12. Despite its general success in trend identification, LSTM's ability to manage short-term volatility remains limited, requiring more precise calibration.

The performance of LSTM for supply forecasting (Figure 9.13) followed a similar pattern, with noticeable deviations between actual and predicted values. These discrepancies, further illustrated by the residuals in Figure 9.14, indicate that the model struggles to maintain accuracy during volatile periods. These findings suggest that while LSTM is proficient at capturing long-term trends, its ability to handle real-world market volatility could be significantly enhanced through further tuning. Adjustments in hyperparameters such as the number of hidden units, learning rate, and batch size are critical for improving its performance. Additionally, incorporating mechanisms such as attention layers or hybrid

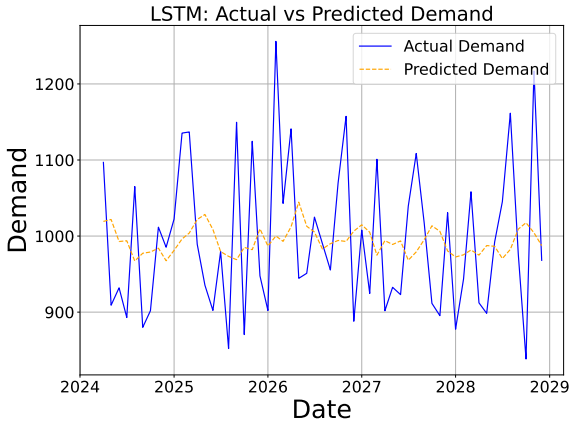

Figure 9.11: LSTM: Actual vs. Predicted Demand.

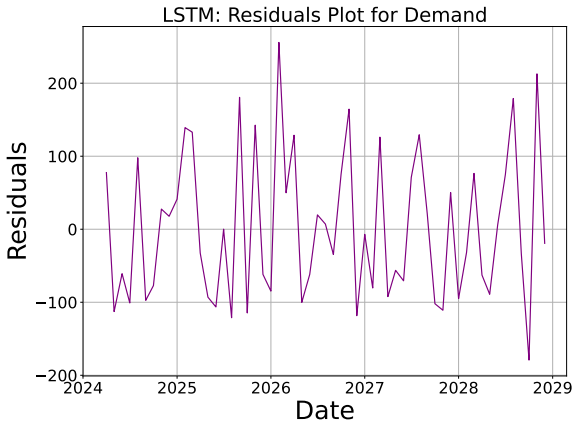

Figure 9.12: LSTM: Demand Residuals.

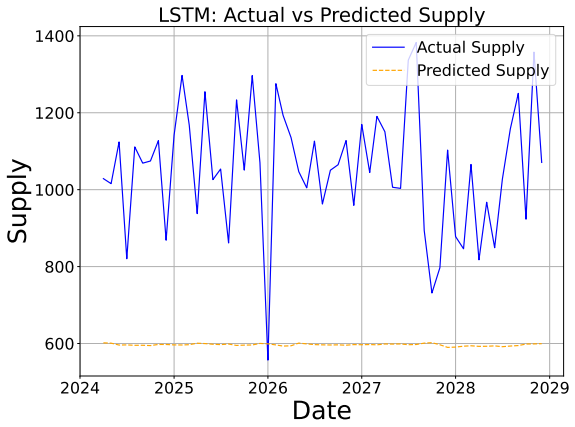

Figure 9.13: LSTM: Actual vs. Predicted Supply.

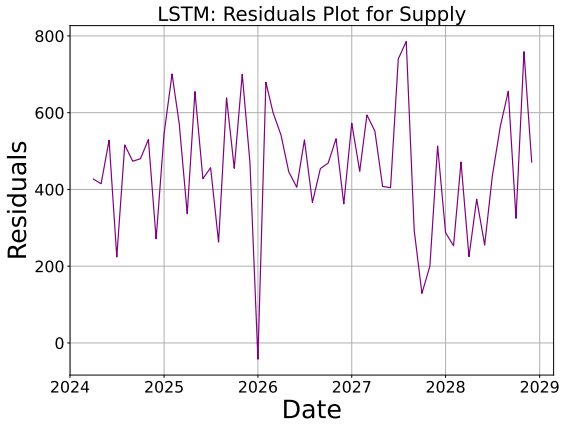

Figure 9.14: LSTM: Supply Residuals.

models could help address the limitations seen in volatile environments, making the model more robust for dynamic market conditions.

The Random Forest model, leveraging ensemble learning, outperformed LSTM in handling non-linear relationships and complex interactions between variables. As shown in Figure 9.15, Random Forest demonstrated higher accuracy in demand prediction, although some deviations still occurred during periods of high variance. The residuals (Figure 9.16) highlight these deviations but show that Random Forest managed market fluctuations more effectively than LSTM. This can be attributed to the model's inherent capability to reduce overfitting by aggregating multiple decision trees, thereby stabilizing predictions during volatile market conditions. Additionally, Random Forest's robustness against multicollinearity and its ability to handle missing data further contributed to its superior performance in high-dimensional datasets.

For supply forecasting (Figure 9.17), Random Forest performed robustly, closely following actual supply trends, with relatively low residual errors (Figure 9.18). This demonstrates that Random Forest, with its capacity for capturing non-linear dependencies, is a more reliable model in a marketplace where supply-demand relationships are dynamic and multifaceted.

Moreover, the model's adaptability to complex data structures enables it to capture subtle interactions between predictors that may be overlooked by simpler models. Fine-tuning of hyperparameters such as the number of decision trees and tree depth can further enhance model accuracy, particularly in scenarios where rapid shifts in supply-demand dynamics occur.

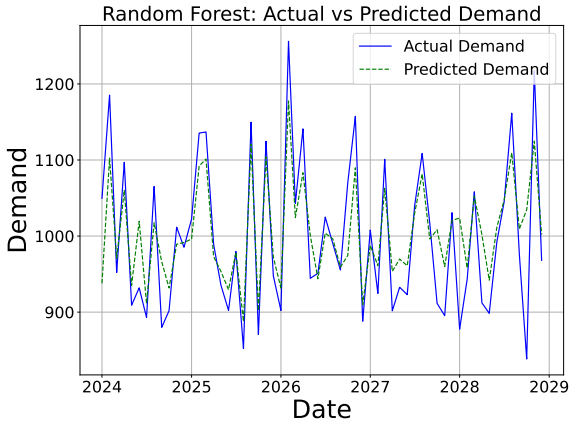

Figure 9.15: RF: Actual vs. Predicted Demand.

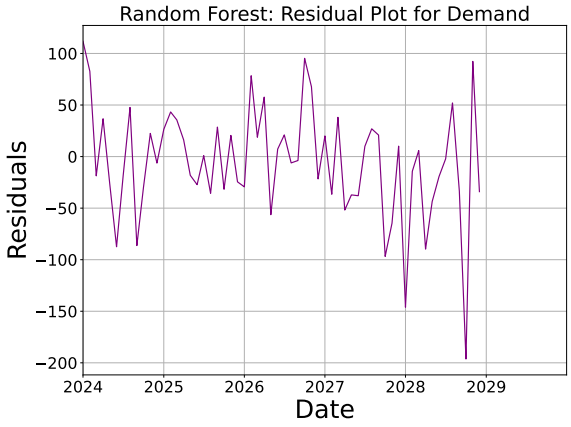

Figure 9.16: RF: Demand Residuals.

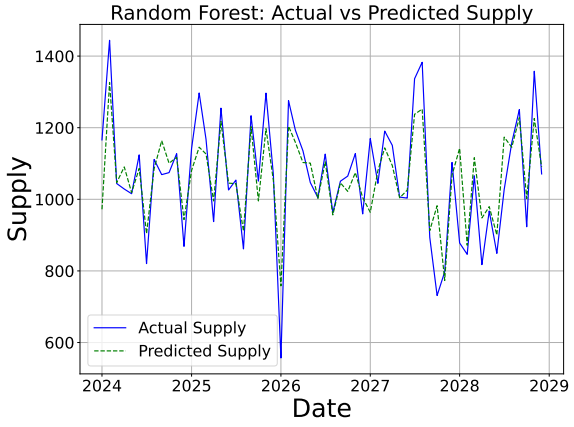

Figure 9.17: RF: Actual vs. Predicted Supply.

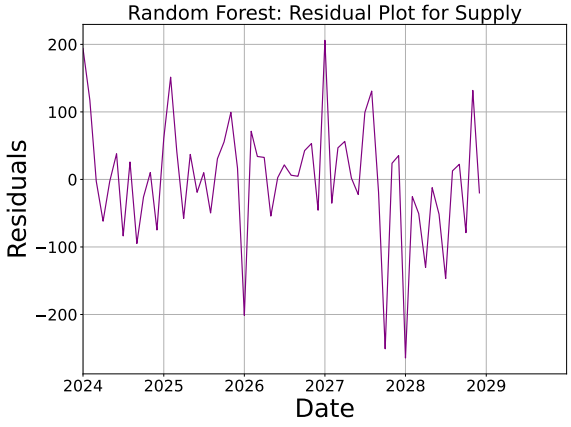

Figure 9.18: RF: Supply Residuals.

Prophet, designed specifically to capture long-term trends and seasonality, performed well when forecasting demand and supply in environments with periodic variations. As shown in Figure 9.19, the model successfully captured the general demand trends, although short-term fluctuations were less accurately predicted. This limitation is particularly evident during periods of abrupt market shifts, where Prophet's emphasis on long-term seasonal patterns made it less responsive. The residuals (Figure 9.20) were smaller compared to LSTM and Random Forest, reflecting Prophet's strength in handling seasonal components and its ability to decompose time-series data into trend, seasonal, and holiday effects.

For supply forecasting (Figure 9.21), Prophet similarly tracked long-term trends effectively, though its performance diminished during rapid supply changes. The residuals (Figure 9.22) were relatively small, underscoring the model's ability to manage periodic behavior with minimal error.

However, its reduced accuracy during high-volatility periods highlights a potential weakness in its fixed seasonal structure. To improve Prophet's performance, future efforts could focus on refining its ability to detect and adjust to changepoints, enhancing the model's responsiveness to sudden shifts in supply-demand dynamics. Incorporating external regressors or applying more dynamic changepoint detection mechanisms could further improve the model's flexibility in fast-evolving market environments.

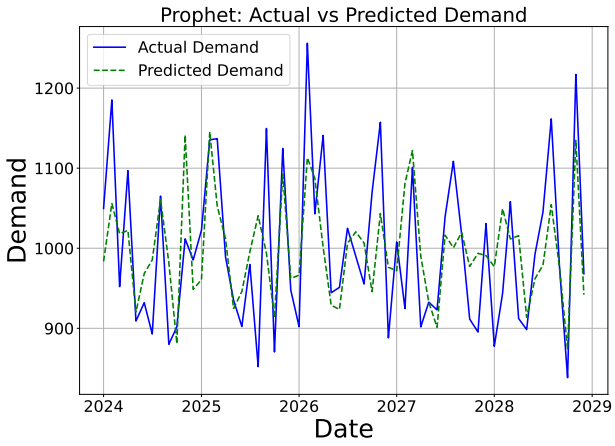

Figure 9.19: Prophet: Actual vs. Predicted Demand.

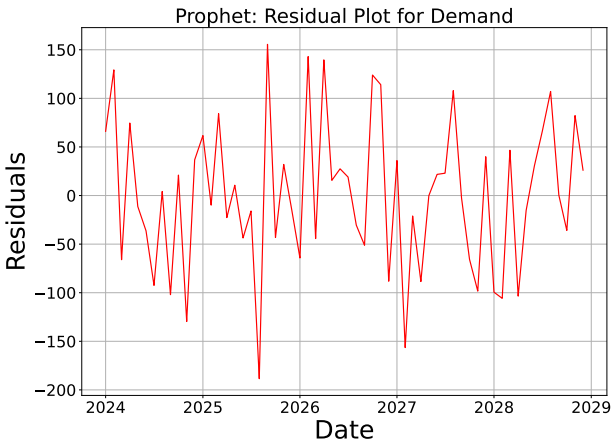

Figure 9.20: Prophet: Demand Residuals.

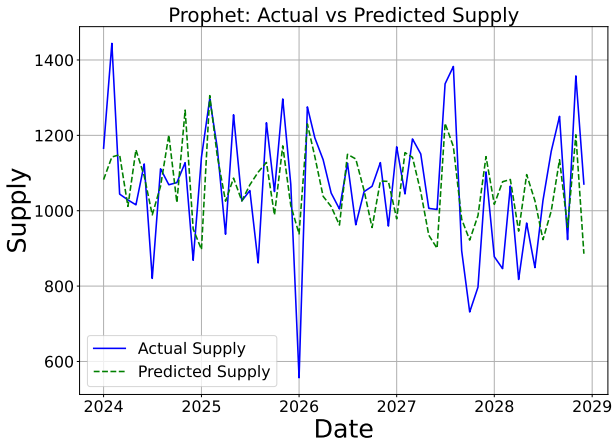

Figure 9.21: Prophet: Actual vs. Predicted Supply.

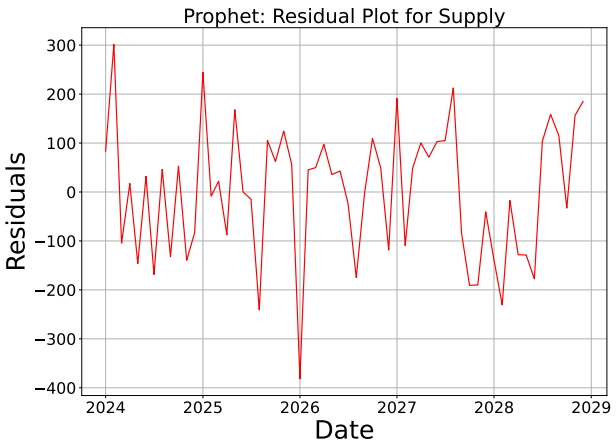

Figure 9.22: Prophet: Supply Residuals.

Potential for Model Improvement: The LSTM model's performance can be further improved by optimizing hyperparameters such as the number of units in the hidden layers, the learning rate, and the sequence length. Additionally, employing attention mechanisms could help the model focus on critical time steps, potentially reducing errors during high-variance periods. Fine-tuning the number of training epochs and incorporating regularization techniques could also enhance the model's generalizability in volatile market conditions.

For the Random Forest model, adjusting the number of decision trees and the maximum tree depth could help mitigate overfitting and improve predictive accuracy. Feature

importance analysis may also provide deeper insights into which variables most influence demand and supply, guiding future model development. Implementing a hybrid model that combines Random Forest with other techniques, such as boosting methods, could further enhance performance.

Prophet's performance could benefit from refining its seasonal components and adjusting the changepoint prior scale to increase its flexibility in handling sudden shifts. Introducing domain-specific knowledge into the model, such as incorporating more granular supply and demand factors, may also improve its accuracy in capturing short-term fluctuations. Future work could explore combining Prophet with machine learning models to further enhance

Analysis of Advanced Models (GBR and Neural Network): The advanced models—Gradient Boosting Regressor (GBR) and Neural Networks (NN)—demonstrate more sophisticated forecasting techniques by capturing non-linear dependencies and complex interactions within the data. These models were evaluated based on their demand and supply forecasting capabilities, as visualized in the following figures.

For the **GBR model**, the demand forecast results (Figure 9.23) show a strong alignment between actual and predicted values, with the model able to capture both long-term trends and short-term fluctuations effectively. This can be attributed to the gradient boosting technique, which builds an ensemble of weak learners to minimize prediction errors iteratively. The residuals (Figure 9.24) reveal relatively small prediction errors, although there are minor deviations in high-demand periods, where sudden market shifts introduce more variance. Despite these deviations, the model's capacity to adjust to complex demand patterns underscores its strength in environments characterized by dynamic and non-linear behavior.

The GBR supply forecast (Figure 9.25) also closely follows the actual values, demonstrating the model's proficiency in capturing supply dynamics. However, the residuals (Figure 9.26) indicate that there is room for improvement in handling supply volatility, particularly during periods of rapid changes in supply levels. This suggests that the model may benefit from further refinement in its loss function or the introduction of regularization tech-

niques to mitigate overfitting in volatile periods. Additionally, fine-tuning hyperparameters such as the learning rate, the number of boosting iterations, and the maximum depth of trees could improve its performance in high-volatility scenarios. Incorporating external factors, such as market shocks or other exogenous variables, could further enhance the GBR model's robustness in supply-demand forecasting.

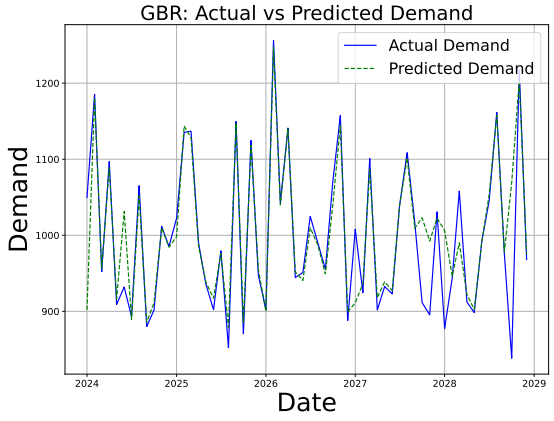

Figure 9.23: GBR: Actual vs. Predicted Demand.

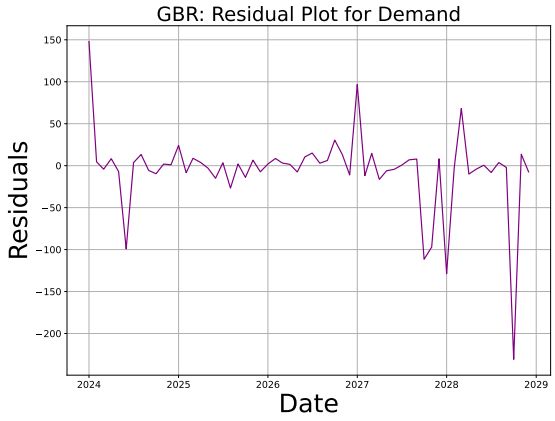

Figure 9.24: GBR: Demand Residuals.

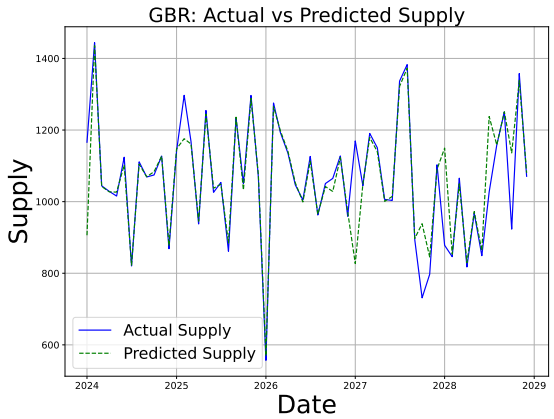

Figure 9.25: GBR: Actual vs. Predicted Supply.

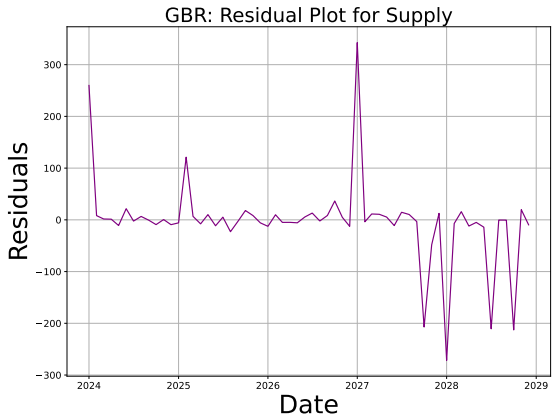

Figure 9.26: GBR: Supply Residuals.

The **Neural Network model** demonstrated strong performance in demand forecasting, as shown in Figure 9.27, where the predicted values align well with the actual demand over time. The residuals (Figure 9.28) reflect minimal prediction errors, though there are

some spikes during volatile periods, indicating that the model can be further fine-tuned for better accuracy. For supply, the NN model also performed well (Figure 9.29), with predicted values closely following actual supply. The residuals (Figure 9.30) show slightly higher errors compared to demand, suggesting that additional optimization might be required to enhance the model's robustness in supply forecasting.

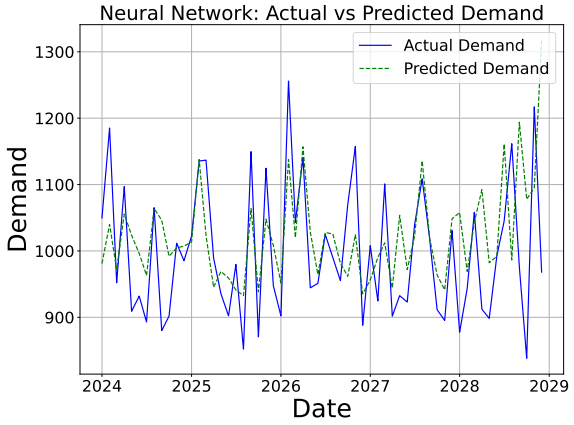

Figure 9.27: NN: Actual vs. Predicted Demand.

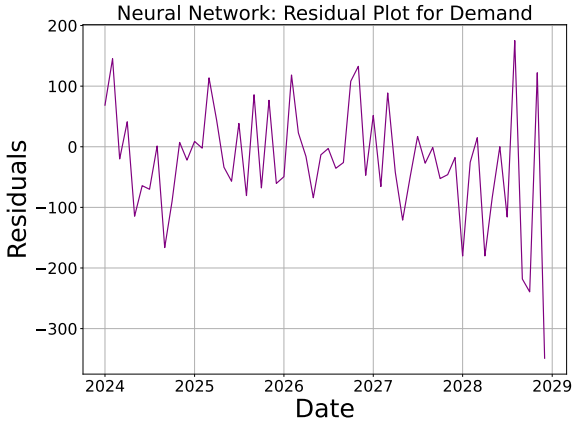

Figure 9.28: NN: Demand Residuals.

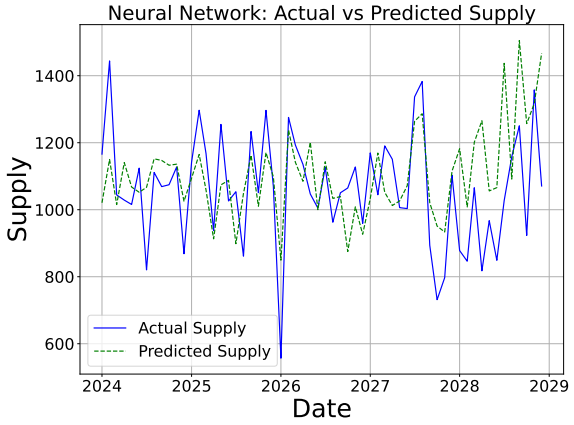

Figure 9.29: NN: Actual vs. Predicted Supply.

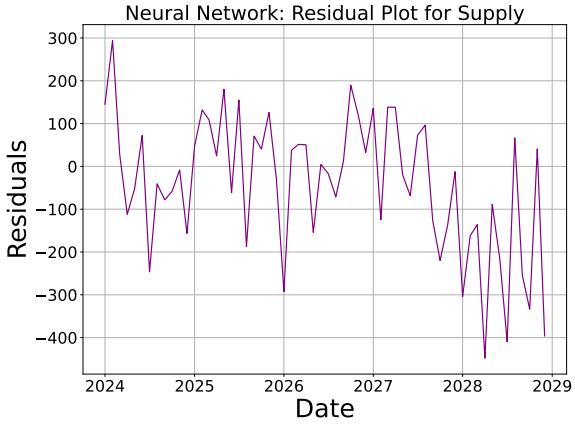

Figure 9.30: NN: Supply Residuals.

Potential for Model Optimization: Both the GBR and Neural Network models exhibit strong performance in forecasting demand and supply. However, further improvements can be achieved by optimizing key aspects of each model.

For the **GBR model**, tuning the number of boosting iterations, learning rate (λ), and the depth of the decision trees can help to better capture complex interactions between variables. Additionally, regularizing the model can prevent overfitting, particularly in scenarios where high variability in supply and demand exists.

In the **Neural Network model**, improvements can be made by adjusting the number of hidden layers and the number of neurons in each layer, as well as by fine-tuning the learning rate and activation functions (f). The use of advanced techniques such as dropout or batch normalization can further improve the model's generalization capabilities, especially in handling volatile supply-demand dynamics.

Analysis of Pricing Optimization Models (Q-Learning and Deep Q-Learning): In this study, Q-Learning and Deep Q-Learning (DQN) were employed to optimize pricing within the digital marketplace, which is crucial for balancing supply-demand dynamics and maintaining market equilibrium. Both models aimed to learn optimal pricing strategies that adapt based on observed market conditions.

Q-Learning Pricing Model: Q-Learning was utilized to dynamically adjust pricing based on the evolving state of the marketplace. As seen in Figure 9.31, the optimized pricing curve appears flat and stable, representing a minimalistic adjustment mechanism. While the original pricing fluctuates, the optimized pricing demonstrates less variation, emphasizing a conservative approach to reducing volatility. The residuals in Figure 9.32 show smaller and more controlled deviations, indicating the model's focus on stabilizing pricing.

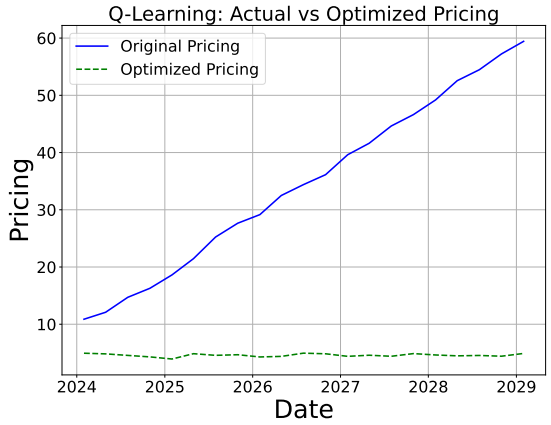

Figure 9.31: QL: Actual vs. Optimized Pricing.

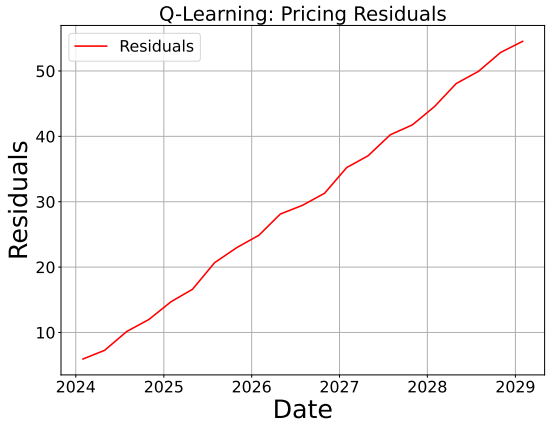

Figure 9.32: QL: Pricing Residuals.

Deep Q-Learning (DQN) Pricing Model: Deep Q-Learning was applied to further enhance the pricing optimization by leveraging a neural network for estimating the Q-values. Figure 9.33 illustrates a similar pattern to Q-Learning, with a stable optimized pricing curve that minimizes deviations from the original pricing. The residuals shown in Figure 9.34 confirm a reduction in prediction errors, demonstrating the DQN model's ability to maintain pricing consistency while avoiding large fluctuations.

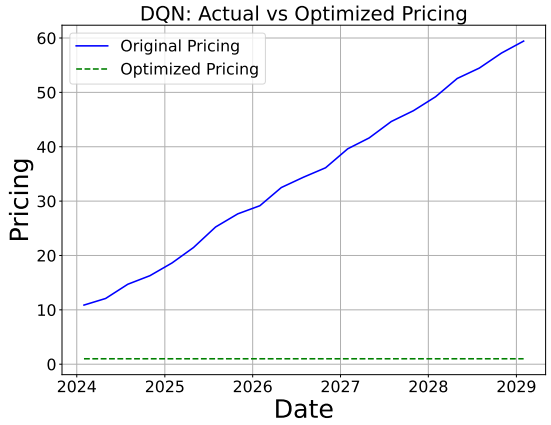

Figure 9.33: DQN: Actual vs. Optimized Pricing.

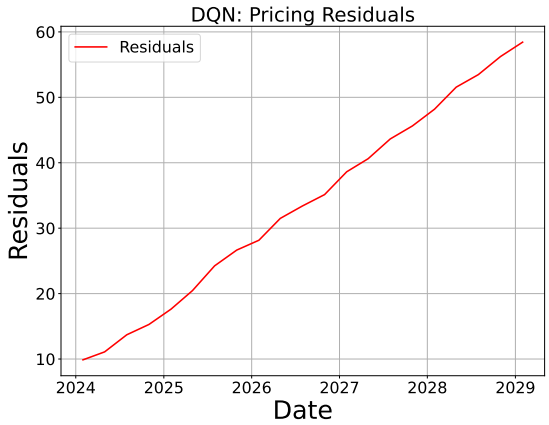

Figure 9.34: DQN: Pricing Residuals.

Potential for Model Improvement: Both Q-Learning and Deep Q-Learning can be further refined by adjusting key hyperparameters such as the learning rate (α) and discount factor (γ) to improve convergence and pricing accuracy. For DQN, the architecture of the neural network (e.g., number of layers and neurons) could be tuned to better capture complex pricing patterns. These improvements can enhance the models' responsiveness to market dynamics and further stabilize pricing.

Analysis of Additional Variables (Economic Growth Rate, Energy Prices, Waste Generation): In addition to the core variables of demand, supply, and pricing, the introduction of Economic Growth Rate, Energy Prices, and Waste Generation provided further insights into system dynamics. These variables, particularly Energy Prices, played a significant role in influencing market pricing. Energy Prices exhibited a strong correlation with pricing (0.71), emphasizing the importance of accounting for real-time energy cost fluctuations in resource allocation and price optimization strategies.

Model Selection Rationale: We chose Gradient Boosting Regressor (GBR), Neural Networks, and Linear Regression for this analysis due to their complementary strengths in handling different types of relationships in the data. GBR was selected for its ability to capture complex interactions between variables and its robust performance in non-linear systems. Neural Networks were chosen for their flexibility in learning intricate, non-linear relationships, particularly when additional variables are introduced. Finally, Linear Regression was included as a baseline due to its simplicity in modeling linear relationships, allowing for comparison with the more sophisticated models.

Economic Growth and Energy Prices: The results from the GBR and Neural Network models (Figures 9.35 to 9.40) indicate that Economic Growth Rate had minimal direct impact on short-term demand and supply predictions. However, Energy Prices played a more significant role in influencing pricing dynamics. As expected, the GBR model showed

improved performance in terms of R^2 and error metrics (MAE, MSE, RMSE) compared to Neural Networks and Linear Regression.

Waste Generation: The influence of Waste Generation on short-term demand and supply forecasts was negligible, as reflected by the weak correlations and high residual errors. This suggests that Waste Generation may have a more pronounced impact in long-term planning and sustainability efforts, rather than short-term resource allocation.

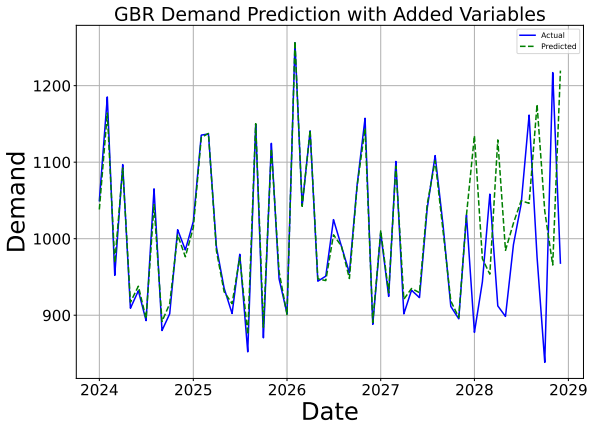

Figure 9.35: GBR: Demand Prediction with Added Variables.

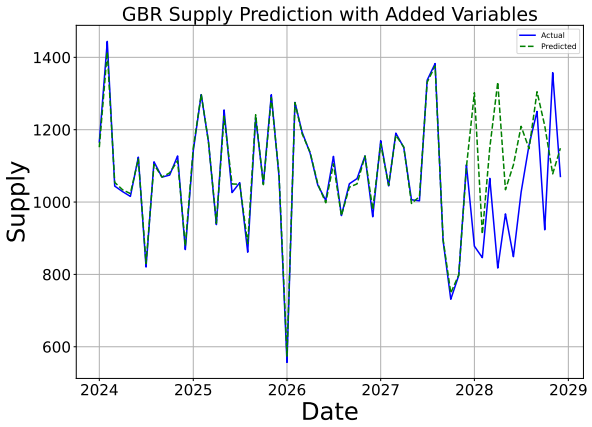

Figure 9.36: GBR: Supply Prediction with Added Variables.

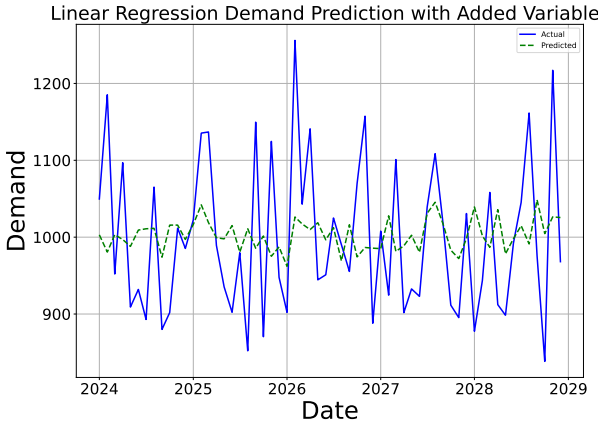

Figure 9.37: LR: Demand Prediction with Added Variables.

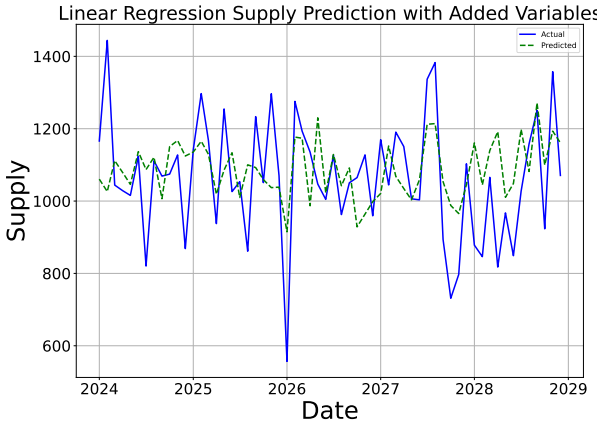

Figure 9.38: LR: Supply Prediction with Added Variables.

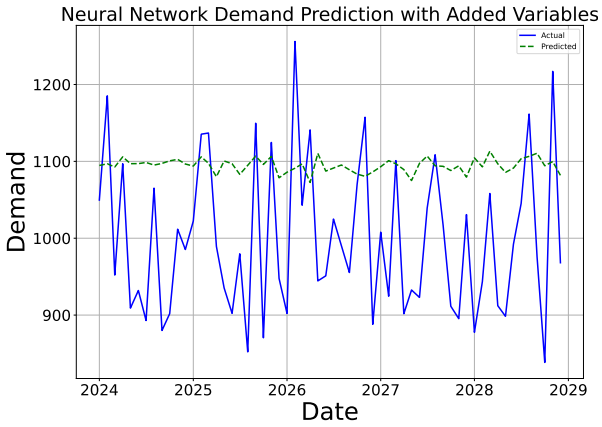

Figure 9.39: NN: Demand Prediction with Added Variables.

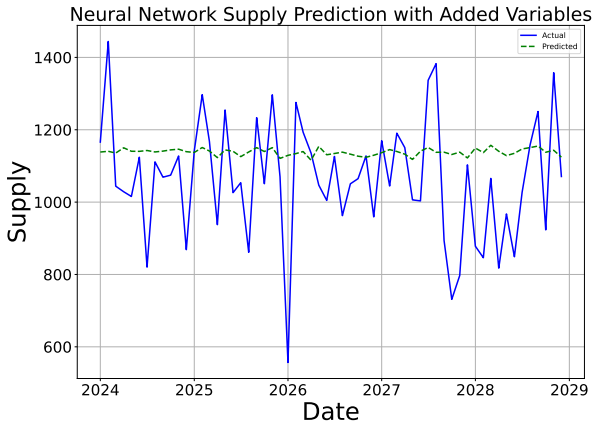

Figure 9.40: NN: Supply Prediction with Added Variables.

The integration of Economic Growth, Energy Prices, and Waste Generation variables required further hyperparameter adjustments to capture each model's sensitivity to these factors. For instance, GBR and Neural Networks were re-optimized to incorporate these additional variables, with adjustments in learning rate and estimators to balance predictive accuracy. This further reinforced the models' capability to respond dynamically to evolving market conditions and environmental factors within CE systems.

9.1.5 Discussion

Practical Implications

The application of AI-driven models within Circular Economy (CE) marketplaces offers substantial benefits for resource allocation, waste reduction, and market optimization. This study evaluated several models—ARIMA, Linear Regression (LR), Long Short-Term Memory (LSTM) networks, Prophet, Random Forest (RF), Gradient Boosting Regressor (GBR), Q-learning, and Neural Networks (NN)—each of which demonstrated distinct capabilities for predictive accuracy and adaptability in dynamic CE contexts.

Model Selection Rationale: Random Forest and Gradient Boosting Regressor were selected for their proficiency in managing non-linear, high-variability contexts characteris-

tic of CE marketplaces. Ensemble methods such as these surpass traditional linear models by effectively capturing demand-supply fluctuations in dynamic resource-driven markets [72,129]. Reinforcement learning methods like Q-learning enhanced pricing optimization in equilibrium-sensitive environments, thereby contributing to real-time adaptability [90].

Demand-Supply and Pricing Optimization: Demand-supply alignment is essential for CE systems, where inventory fluctuations directly impact resource allocation efficiency. Traditional time-series models like ARIMA, configured with parameters $p = 2$, $d = 1$, and $q = 1$, effectively captured periodic trends for short- to medium-term forecasts, relevant in applications such as urban waste management. For more complex seasonal patterns, Prophet's ability to decompose time-series data proved advantageous in scenarios with cyclical resource flows, while Random Forest and GBR excelled in capturing non-linear relationships among variables like resource availability, waste generation, and pricing. Q-learning supported dynamic pricing by stabilizing action-value functions in real-time, fostering equilibrium in CE marketplaces and reducing resource stockpiling or waste.

Strategic Policy Support: Neural Networks, configured with a two-layer architecture and ReLU activation, offer long-term strategic planning capabilities, essential for projects that depend on adaptable, data-informed resource allocation. The models used in this study can be further tailored to waste management, recycling, and sustainable manufacturing, allowing CE marketplaces to leverage AI for timely adjustments in resource distribution. These approaches thus support policy-makers and industry leaders in developing resilient, sustainable CE ecosystems. Future studies involving real-world data will require additional hyperparameter optimization to further tailor model performance and adaptability in dynamic CE settings.

Application in Circular Economy Marketplaces

AI models for demand forecasting and dynamic pricing offer significant potential for CE marketplaces, including waste management, recycling, and sustainable manufacturing. Ensemble methods like GBR and Random Forest achieved high predictive accuracy, with GBR obtaining an MAE of 23.39 for demand and 35.66 for supply, underscoring their applicability in sectors with volatile resource demands [102]. Reinforcement learning techniques such as Q-learning further stabilized pricing strategies, thereby reducing resource hoarding and wastage, essential for efficient resource distribution. Adjusting parameters like learning rate (α) and discount factor (γ) could enhance model responsiveness during peak demand.

AI models support both economic and environmental goals by promoting resource efficiency and waste minimization, and their implementation within CE marketplaces offers a pathway to a sustainable resource economy.

Ethical Considerations

Integrating AI models in CE marketplaces raises ethical considerations crucial for ensuring fairness, transparency, and societal well-being. These issues include algorithmic bias, data privacy, and equitable resource access [130].

Algorithmic Bias and Data Privacy: Bias in AI algorithms may lead to unequal access to resources if models favor certain demographics based on historical data patterns [131,132]. Addressing this bias requires regular model evaluation and recalibration, especially where fair resource access is a CE goal [133]. Additionally, the use of high-dimensional data in models like Neural Networks necessitates strict data privacy protocols to prevent unintended exposure of sensitive information [134].

Equity and Transparency: Ensuring equitable access in dynamic pricing models, particularly those driven by reinforcement learning, is essential for inclusivity in digital CE marketplaces [135]. Explainable AI (XAI) methods like SHAP for GBR and LIME for Neural

Networks enhance transparency, providing insights into how model decisions are influenced by variables, thereby fostering stakeholder trust and ensuring accountability [136, 137].

Model Limitations and Future Work

While the models demonstrated significant potential, several limitations warrant attention for future improvements in CE resource management and dynamic pricing strategies.

- **Reliance on Synthetic Data**: Synthetic data, while useful for controlled testing, lacks the variability of real-world datasets. Future studies should validate model performance using real-world data to ensure adaptability to market fluctuations [138].
- **Hyperparameter Tuning**: Limited hyperparameter tuning may restrict model optimization. Advanced tuning techniques, such as Bayesian optimization, could enhance model performance in future applications [139].
- **Multi-Objective Optimization**: This study focused on single-objective optimization. Future research should explore multi-objective approaches to address the diverse goals of CE systems, including hybrid models that integrate predictive accuracy with dynamic pricing [140].
- **Dynamic Environment Adaptability**: More adaptive learning models, such as reinforcement learning with meta-learning, could better support real-time adjustments in CE marketplaces [141].
- **Computational Efficiency**: Computational demands of models like LSTM challenge scalability. Future research could explore optimization methods like model pruning to maintain efficiency in large-scale applications [142].

Addressing these limitations will enhance AI frameworks' applicability and scalability in CE marketplaces, advancing sustainable resource management.

9.1.6 Chapter Summary

This case study demonstrated the application of AI-driven models—including ARIMA, Gradient Boosting Regressor (GBR), Neural Networks, and Q-learning—to forecast demand,

manage supply, and stabilize pricing within Circular Economy (CE) digital marketplaces. The GBR model exhibited superior performance in capturing non-linear dynamics, while Q-learning effectively reduced price volatility, highlighting the relevance of reinforcement learning for adaptive resource control. While synthetic data enabled controlled experimentation, future work must validate these findings using real-world CE datasets and explore multi-objective and hyperparameter-optimized extensions for improved deployment readiness.

From a System of Systems (SoS) perspective, this study reinforces the benefits of decentralized, learning-enabled coordination mechanisms within sustainability-driven digital ecosystems. By translating raw data into actionable resource and pricing decisions, AI techniques offer valuable pathways toward waste minimization, circularity, and urban resilience.

In the next chapter, we synthesize cross-domain insights from all case studies to evaluate generalizable patterns in coordination, scalability, and learning performance across SoS implementations. **Chapter 10** explores the comparative strengths of centralization versus decentralization, identifies recurring design trade-offs, and reflects on the human, technical, and ethical considerations of applying machine learning in complex, distributed systems.

Chapter 10

Cross-Domain Synthesis and Discussion

Building on the detailed case study analyses presented in earlier chapters, the following synthesis evaluates how Machine Learning (ML) frameworks, when embedded within System-of-Systems (SoS) environments, operate across diverse domains. Rather than isolating domain-specific insights, the focus here is on identifying cross-domain coordination patterns, design trade-offs, and architectural principles that consistently emerged in contexts such as semiconductor manufacturing, healthcare, smart cities, and circular economy platforms.

By examining how agents learn, adapt, and align with global objectives under varying SoS typologies and data conditions, this chapter establishes a foundation for generalizable ML-SoS design. The discussion integrates observations from reward structuring, scalability, explainability, and system integration to inform the development of future intelligent, multi-agent SoS architectures.

10.1 Overview of Cross-Domain Integration

This chapter synthesizes key findings from the seven domain-specific case studies presented in Chapters 5 through 9, highlighting generalizable insights, trade-offs, and methodological contributions of the proposed ML-SoS coordination framework. While each case addresses unique operational challenges and application contexts, they share foundational characteristics as complex, decentralized, and dynamically interacting Systems of Systems (SoS).

The synthesis is structured to identify and evaluate patterns in coordination architectures, reward design, scalability, agent behavior, and policy convergence across domains such as semiconductor manufacturing, healthcare, electronics quality control, SME optimization, smart cities, and circular economy marketplaces. In doing so, this chapter bridges the

domain-specific results with broader principles of intelligent system coordination and decision support.

The objectives of this chapter are threefold:

1. To compare coordination models and SoS architectures used across the case studies, including hierarchical, decentralized, and hybrid strategies.
2. To extract generalizable patterns and evaluate their implications for learning convergence, agent cooperation, and responsiveness in dynamic environments.
3. To critically assess cross-domain trade-offs, technical limitations, and design considerations for ML-based SoS frameworks.

The insights from this chapter aim to inform future research and practical applications of ML in real-world SoS deployments. The concluding sections also reflect on human-centric factors such as explainability, trust, and ethics, which are increasingly relevant in AI-powered decision systems.

Following this overview, Section 10.2 examines the coordination models adopted across domains, comparing centralized, decentralized, and hierarchical approaches. Section 10.3 explores generalizable patterns in learning dynamics and agent behavior. Section 10.4 analyzes design trade-offs such as efficiency versus generalizability and local versus global alignment. Section 10.5 highlights technical and data-related constraints, while Section 10.6 discusses human-in-the-loop considerations and ethical implications. The chapter concludes with a forward-looking reflection on the role of ML in future SoS engineering (Section 10.7) and a summary that transitions to **Chapter 11**.

10.2 Coordination Models in SoS Systems

Effective coordination across distributed subsystems is central to the success of System-of-Systems (SoS) frameworks. Across the seven case studies presented in this dissertation, a spectrum of coordination models was employed—ranging from fully centralized architectures to decentralized and hybrid configurations. This section analyzes the comparative

effectiveness of these models, focusing on their impact on learning performance, scalability, and system-wide alignment.

Centralized vs. Decentralized Control

Case studies involving centralized control architectures, such as the Directed SoS model in healthcare (Chapter 7), revealed important trade-offs. While centralized models offer tight control and policy uniformity, they exhibited performance volatility under dynamic conditions. For instance, agents in the Directed healthcare SoS failed to converge to stable reward policies, accumulating negative cumulative rewards and displaying reduced adaptability. These findings align with prior critiques of centralized coordination in rapidly changing environments.

In contrast, decentralized configurations—such as the Acknowledged and Collaborative healthcare SoS, the smart city MARL framework, and the 3D-IC optimization loop—achieved superior system-wide outcomes. Agents were able to independently optimize local behaviors while maintaining alignment with global goals through carefully designed reward structures. These decentralized models supported faster convergence, greater adaptability, and higher cooperation rates across subsystems.

Hierarchical and Hybrid Coordination Architectures

The semiconductor case studies illustrate the power of hierarchical coordination. In the hierarchical MARL framework, global, main-agent, and sub-agent reward layers enabled both top-down guidance and bottom-up adaptation. This structure allowed sub-agents, such as fabrication tools or inspection modules, to optimize task-specific policies while contributing to higher-level system objectives. A similar principle was evident in the 3D-IC case, where subsystem outputs (e.g., defect detection, TSV optimization) were integrated into a global yield-maximization loop.

Hybrid models also emerged in the smart city and circular economy studies. In smart cities, agents for transportation, energy, safety, and communication operated semi-independently

but were governed by augmented global rewards. In the CE marketplace, predictive and reinforcement learning components were blended to balance real-time responsiveness with long-term sustainability goals.

Effectiveness Across Domains

Regardless of domain, decentralized or hybrid coordination structures consistently outperformed centralized ones in terms of adaptability, cooperation, and long-term performance. The AIoMT-enabled healthcare case study and smart city MARL framework exemplify this, achieving high cumulative rewards and system-wide efficiencies under decentralized policies. In contrast, the Directed healthcare SoS and other monolithic approaches encountered scalability and convergence issues.

These findings support a generalizable insight: coordination models that preserve agent autonomy while embedding systemic incentives foster more resilient, scalable, and effective SoS behavior.

10.3 Generalizable Patterns Across Domains

While each case study addressed domain-specific challenges, several recurring patterns emerged across all seven implementations of Machine Learning-enabled System-of-Systems (ML-SoS) frameworks. These patterns reflect the broader utility of coordinated ML agents in dynamic environments and provide insights into convergence behavior, scalability, and adaptability under uncertainty.

Learning Convergence and Policy Adaptation

Across domains, convergence to stable policies was most successful in configurations that balanced localized learning with global guidance. In both the smart city and healthcare case studies, decentralized agents demonstrated consistent policy improvement through reward shaping and cooperation incentives. The smart city MARL framework achieved early reward

stabilization within 3–5 epochs, while healthcare agents in Acknowledged and Collaborative SoS configurations converged to optimal Q-values with high consistency.

Adaptability was similarly enhanced by context-aware reward design. In the 3D-IC manufacturing case, subsystem-level ML models dynamically aligned with system-wide yield goals, allowing adaptive optimization under varying TSV process parameters and defect patterns. Similarly, the CE marketplace framework used Q-learning to adjust prices in real-time, stabilizing supply-demand fluctuations—evidence of adaptive policy refinement through continuous feedback.

Scalability of AI Techniques in SoS

Several case studies validated the scalability of AI techniques when designed with modularity and reward abstraction. The hierarchical MARL framework in semiconductor manufacturing scaled across sub-agents and main agents by decomposing global rewards into layered incentive structures. In the smart city domain, the modular design of transportation, energy, and emergency agents enabled parallel learning and policy differentiation without centralized bottlenecks.

However, computational scalability remained a limiting factor in some settings. For example, the healthcare and smart city MARL frameworks required coordination overhead and computational tuning, particularly when multiple agents and high-dimensional states were involved. The Circular Economy case also highlighted computational challenges with deep learning models like LSTM, reinforcing the need for model pruning and hardware-aware deployment strategies in large-scale systems.

Resilience and Responsiveness in Dynamic Environments

Dynamic environments—including variable demand in CE marketplaces, patient surges in healthcare, and fluctuating urban traffic—tested the frameworks' responsiveness. Across all domains, the most resilient architectures exhibited:

Reward-driven alignment: Agents using dynamic or hierarchical rewards consistently responded to shifting conditions.

Policy flexibility: Models like PPO and Q-learning adapted agent behavior to uncertain environments.

Minimal reliance on central control: Distributed decision-making allowed agents to re-optimize locally when global plans became suboptimal.

For example, in the healthcare simulation, the Virtual SoS maintained cumulative rewards under minimal resource utilization, demonstrating resilience in low-infrastructure scenarios. In the PCB defect case study, rapid classification using Naïve Bayes enabled near real-time quality control using unstructured human-generated reports.

Together, these patterns reinforce that SoS designs integrating flexible ML agents with adaptive reward structures are well-suited for dynamic, data-constrained, and distributed operational environments.

10.4 Trade-offs and Design Considerations

The case studies presented in this dissertation each required balancing multiple trade-offs during the design and implementation of ML-enabled System-of-Systems (SoS) frameworks. These trade-offs—between accuracy and interpretability, efficiency and generalizability, and local versus global optimization—recurred across domains and underscore the complexity of engineering ML-based coordination systems. This section synthesizes the lessons learned around these three dimensions.

Accuracy vs. Interpretability

High-performance models often sacrifice transparency in favor of predictive power. For example, in the PCB quality control case study, Naïve Bayes classifiers were chosen over more complex deep learning models due to their simplicity and interpretability. While deep neural networks or ensemble methods could potentially increase accuracy, they introduce opacity

that is often unacceptable in regulated or human-in-the-loop environments. The trade-off here favored interpretability to support explainable AI for technician-assisted decision-making.

In contrast, the CE marketplace and healthcare case studies employed more complex models—such as LSTM, Gradient Boosting, and MARL—to optimize dynamic pricing and resource allocation. Although these models yielded superior predictive accuracy and adaptability, their decision processes were difficult to trace, particularly in high-stakes or real-time contexts. To mitigate this, future systems may require explainability tools such as SHAP or LIME to bridge the gap between model complexity and stakeholder trust.

Efficiency vs. Generalizability

The need for computational efficiency was particularly salient in real-time or resource-constrained environments. In semiconductor manufacturing (Case Study 1), the AI framework was designed for rapid inference, completing predictions in milliseconds after training. This constraint prioritized model types such as Random Forest and XGBoost, which offer fast predictions while maintaining acceptable performance.

However, this focus on efficiency limited the generalizability of the models across broader scenarios. For instance, defect detection models trained on structured tabular data may not transfer well to image-based or multimodal defect types. Similarly, simplified PPO training in the smart city simulation was capped at five epochs per agent, a choice driven by convergence speed and reproducibility. While effective in context, this limits the model's potential to generalize across more complex or unseen urban dynamics.

Conversely, the MARL frameworks for healthcare and smart cities were designed with generalizability in mind, accommodating heterogeneous agents and coordination structures. This increased the framework's domain adaptability, but introduced greater computational demands and a steeper learning curve for convergence. The trade-off here lies in striking a balance between deployment readiness and model flexibility.

Local Optimization vs. Global Alignment

Perhaps the most consistent and domain-transcending trade-off involved aligning local subsystem or agent behavior with global SoS-level objectives. In 3D-IC manufacturing, for example, optimizing TSV depth or defect thresholds in isolation risked undermining yield improvements downstream. The unified optimization loop resolved this by adjusting subsystem parameters dynamically based on their impact on global yield.

Similarly, in the MARL-based smart city framework, agents operated with local rewards but were augmented with global coordination terms to drive emergent behavior. This balance was essential in preventing siloed decisions from degrading overall system efficiency. The healthcare SoS simulations further demonstrated that decentralized agents could achieve strong global performance—provided reward signals were structured to incentivize cooperation.

Yet, the circular economy case study exposed the difficulty of achieving global alignment in systems governed by market dynamics. Price stabilization via Q-learning optimized local agent profitability, but required fine-tuning of reward functions to avoid destabilizing long-term supply-demand balance. This underscores the need for iterative design of reward and control structures in multi-agent environments, particularly when agents have partially aligned or competing incentives.

In sum, trade-offs are not merely implementation constraints but active design levers in the engineering of ML-SoS systems. Understanding and managing these trade-offs—between model complexity and interpretability, performance and adaptability, autonomy and coordination—is essential for deploying robust, scalable, and ethically sound intelligent systems.

10.5 Technical Limitations and Data Constraints

Despite the promising outcomes demonstrated across domains, the deployment of ML-based System-of-Systems (SoS) frameworks remains constrained by several technical and data-related limitations. This section outlines the most significant challenges encountered in

the case studies, grouped into three recurring themes: reliance on synthetic versus real-world data, increasing model complexity and computational burden, and the scalability limits of multi-agent systems.

Synthetic vs. Real-World Data

Several case studies in this dissertation—including healthcare (Chapter 7) and the circular economy marketplace (Chapter 9)—relied on synthetically generated datasets due to privacy restrictions, lack of access to proprietary data, or the need to simulate rare or edge-case conditions. While synthetic data facilitates controlled experimentation and reproducibility, it may not fully capture the noise, heterogeneity, or unpredictable interactions characteristic of real-world environments.

For example, in the AIoMT-enabled healthcare SoS simulation, agent behaviors were modeled on synthetic patient arrival patterns and device data streams. While the results validated the robustness of decentralized coordination architectures, the absence of real patient variability limited external validity. Similarly, the CE case study used synthetic resource flows and pricing data to test forecasting and Q-learning strategies. Although useful for evaluating model behavior under controlled volatility, these findings require validation with live platform data to confirm robustness in production environments.

Future research must prioritize the integration of real-world datasets—either through partnerships with industry stakeholders or federated data sharing approaches—to ensure model generalizability and operational realism.

Model Complexity and Computational Cost

As demonstrated in the semiconductor and smart city case studies, model complexity increases substantially when coordinating multiple agents, learning across time-series dependencies, or integrating deep learning components. While advanced models such as LSTMs and PPO-based reinforcement learners offer performance benefits, they also introduce high training and inference costs, especially when executed across multiple decision nodes.

For instance, training the adaptive CNN-based TSV optimizer and LSTM-based fault detection models in the 3D-IC case study remained computationally feasible (15–22 minutes on a standard CPU), but scaling such models to additional process stages or real-time settings would likely require GPU acceleration or distributed computing. In smart city simulations, training PPO agents beyond five epochs yielded diminishing performance improvements relative to computational overhead, necessitating early stopping strategies.

These findings highlight a broader tension between algorithmic sophistication and deployment feasibility. As system complexity grows, practical considerations such as hardware availability, training time, and inference latency become critical bottlenecks. Future work should explore efficient training techniques, such as knowledge distillation, model pruning, or transfer learning, to maintain performance while reducing resource demands.

Scalability Challenges in Multi-Agent Systems

Multi-Agent Reinforcement Learning (MARL) frameworks—central to the healthcare, smart city, and semiconductor SoS case studies—introduce scalability challenges as the number of agents and environment dimensions increase. Coordination overhead, communication bottlenecks, and policy convergence instability all grow with system size.

In the smart city case study, although the reward-augmented MARL framework demonstrated improvements in coordination and adaptability, its effectiveness depended on stable communication and shared observability—assumptions that may not hold in real-world deployments with unreliable infrastructure or asynchronous updates. Similarly, the healthcare simulations revealed that agent-level variability across different SoS configurations (e.g., Directed vs. Acknowledged systems) required fine-grained tuning of reward functions and exploration policies to prevent policy divergence.

Moreover, the addition of new agents or subsystems often necessitates retraining or architectural reconfiguration. This limits plug-and-play extensibility—a desirable property in modular SoS environments. While hierarchical reward structures (as used in Chapters 5 and

7) partially mitigate this by abstracting coordination across layers, full scalability requires mechanisms such as dynamic agent discovery, curriculum learning, and policy reuse.

In summary, the technical limitations observed across these case studies are not unique but emblematic of broader challenges in deploying ML-based SoS architectures at scale. Addressing these issues—through data realism, computational efficiency, and scalable agent coordination—remains critical to advancing the practical integration of intelligent decision-making in real-world complex systems.

10.6 Human Factors, Explainability, and Ethics

The integration of Machine Learning (ML) in System-of-Systems (SoS) environments not only presents technical and operational challenges, but also introduces critical human, ethical, and interpretability concerns. As ML-driven decisions increasingly affect stakeholders in domains such as healthcare, manufacturing, and urban infrastructure, ensuring transparency, accountability, and fairness becomes essential. This section discusses how human factors were incorporated or addressed across the case studies, identifies explainability limitations, and reflects on ethical trade-offs relevant to SoS-ML applications.

Human-in-the-Loop Models

Several case studies underscored the importance of human-in-the-loop (HITL) frameworks, particularly in domains where expert judgment, safety, or contextual interpretation are paramount. The PCB defect detection case study (Chapter 6) demonstrated this by integrating technician-written reports into the machine learning pipeline through natural language processing and Naïve Bayes classification. The structured predictions derived from unstructured technician notes highlight the value of retaining human oversight in automated quality control pipelines.

Similarly, in the semiconductor SME case study (Chapter 5), predictive models were deliberately chosen for their interpretability, such as decision trees, to ensure usability by

non-expert operators. These models offered actionable insights without requiring specialized data science knowledge, thus promoting adoption in low-resource settings. Across both cases, the inclusion of human-centric design supported better decision support and organizational trust in ML systems.

The broader implication is that human-in-the-loop design is not only technically advantageous—helping mitigate false positives or contextual edge cases—but also essential for increasing operational acceptance in environments where full automation may be infeasible or undesirable.

Transparency and Trust in ML-Driven SoS

Trust in ML systems depends heavily on transparency, particularly in SoS contexts where decisions have cascading impacts across multiple subsystems. The explainability of model behavior is crucial for diagnosing failure points, debugging learning behavior, and validating compliance with regulatory standards—especially in healthcare and smart cities.

In practice, simpler models such as Naïve Bayes (Chapter 6) and Random Forests (Chapter 5) were preferred in certain domains due to their ease of interpretation, even if more complex models offered marginally better performance. Conversely, black-box models such as LSTMs and neural networks were adopted in time-series fault detection and forecasting (Chapters 5 and 9), but only after performance justifications outweighed interpretability concerns.

To enhance transparency, techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are increasingly relevant. These methods, though not yet fully adopted in the case studies, represent valuable tools for explaining model outputs, particularly in real-time systems where auditability is vital.

Building trust in ML-based SoS therefore depends on balancing complexity and explainability, and ensuring that operators, regulators, and users can understand, validate, and override ML-driven decisions when necessary.

Fairness, Bias, and Ethical Trade-offs

Ethical considerations were especially salient in the circular economy (CE) and healthcare case studies. In CE marketplaces (Chapter 9), dynamic pricing models driven by reinforcement learning offered real-time adaptability but risked fairness violations if algorithmic biases led to disproportionate access or resource allocation. Likewise, the use of synthetic training data posed risks of introducing skewed representations that might affect minority user groups in actual deployments.

In healthcare simulations (Chapter 7), the design of reward functions and agent policies directly influenced patient triage and resource prioritization. These decisions, if deployed in real systems, could raise questions about algorithmic accountability and medical equity. For example, agents trained on synthetic reward structures may underrepresent edge cases or vulnerable populations, potentially exacerbating disparities.

Addressing these concerns involves several strategies:

- Incorporating fairness constraints into reinforcement learning reward structures.
- Periodic auditing of model decisions using explainable AI techniques.
- Simulation of edge-case scenarios during training to identify potential bias.
- Designing fallback mechanisms that defer to human judgment in ethically sensitive situations.

Moreover, these ethical trade-offs extend beyond technical fixes. As ML systems increasingly participate in decision-making, researchers and practitioners must engage with stakeholders, ethicists, and regulators to co-design systems that align with societal norms and legal frameworks.

In sum, the deployment of ML-SoS frameworks in real-world domains must be guided not only by performance metrics, but also by the ability to explain, justify, and adapt decisions in ways that are human-centered and ethically sound. As this dissertation has shown, balancing automation with oversight, complexity with clarity, and optimization with fairness is critical to the responsible evolution of ML in SoS environments.

10.7 Future Directions for SoS Engineering

This dissertation contributes to the advancement of System-of-Systems (SoS) engineering by demonstrating how machine learning (ML), reinforcement learning, and hybrid optimization techniques can be systematically integrated into decentralized, dynamic environments. The case studies and conceptual frameworks explored throughout this work highlight practical pathways for designing, deploying, and scaling intelligent SoS architectures across diverse application domains. This section reflects on the broader implications for future SoS design methodologies, decision-making infrastructures, and research agendas.

Redefining SoS Coordination Through Learning Architectures

Traditional SoS frameworks often rely on fixed communication protocols, pre-defined coordination mechanisms, and static optimization policies. This dissertation introduces a shift toward adaptive coordination, where agents learn optimal behaviors over time using reinforcement learning and feedback-driven reward structures. Across healthcare, manufacturing, smart cities, and circular economy systems, learning-based agents proved capable of aligning local actions with global objectives—despite partial observability, uncertainty, and heterogeneity in system structure.

The integration of hierarchical reward modeling (Chapters 5 and 7) offers a generalizable technique for balancing subsystem autonomy with overarching goals, ensuring that constituent systems contribute meaningfully to SoS performance. These hierarchical constructs provide blueprints for future SoS frameworks to embed learning directly into decision layers, enabling flexible reconfiguration without the need for complete system redesign.

Toward Simulation-Driven SoS Design and Policy Testing

The dissertation also emphasizes the role of simulation as a core design tool in evolving SoS engineering. Every case study in this work relies on simulation environments to test coordination logic, optimize policies, and validate subsystem behaviors under realistic con-

straints. This approach enables early identification of bottlenecks, coordination failures, and unintended consequences, particularly in domains where real-world experimentation is risky or infeasible.

Future SoS development efforts can benefit from simulation-based methodologies as testbeds for policy innovation, allowing engineers and planners to explore reward trade-offs, fairness criteria, resilience strategies, and edge-case behaviors before deployment. The integration of agent-based modeling, reinforcement learning, and statistical evaluation methods provides a scalable and reproducible framework for such experimentation.

Embedding Adaptability and Responsiveness in Infrastructure Design

A recurring theme across all domains studied is the importance of real-time adaptability. Whether in dynamic resource allocation for hospitals, traffic-energy coordination in smart cities, or demand forecasting in circular marketplaces, the ability of agents to learn, adapt, and respond under uncertainty was central to performance.

These findings suggest that future infrastructure systems—whether digital, physical, or organizational—should be designed not as static systems, but as learning-enabled ecosystems. Embedding ML models within the operational loop (e.g., digital twins, real-time analytics pipelines, or AIoMT networks) ensures continuous responsiveness to changing inputs, allowing systems to evolve with their environment.

Generalizability Across Domains and Architectural Flexibility

One of the key implications of this dissertation is the demonstration that the proposed ML-SoS coordination frameworks are domain-agnostic in structure, even while being domain-specific in instantiation. From manufacturing to healthcare to digital marketplaces, the same foundational principles—distributed decision-making, reward alignment, learning-based optimization—proved effective across vastly different application areas.

This generalizability implies that SoS architectures of the future should prioritize modularity and abstraction, allowing new agents, data sources, or objectives to be added with minimal disruption. The framework designs outlined in this dissertation provide templates for constructing plug-and-play SoS components, supported by domain-specific simulation and learning layers.

Shaping the Future of SoS Methodologies

Taken together, these contributions point to several high-level implications for SoS methodology and research:

- **Shift toward learning-centric SoS design:** Move beyond rule-based coordination to learning-enabled systems that adapt over time.
- **Reward shaping as a design primitive:** Use reward engineering to encode high-level objectives, fairness constraints, and multi-agent alignment goals.
- **Simulation-first development:** Employ synthetic environments as a precursor to real-world deployment, enabling rapid iteration and risk-free policy testing.
- **Cross-domain synthesis and reuse:** Develop frameworks with reusable components that can generalize across smart infrastructure, manufacturing, healthcare, and sustainability.
- **Human-ML collaboration as a design principle:** Incorporate explainability, oversight, and interaction as core elements of SoS design—not as afterthoughts.

Conclusion

In summary, this dissertation expands the design space for System-of-Systems engineering by demonstrating how ML frameworks can manage complexity, improve coordination, and support decision-making in highly dynamic environments. Future SoS systems must embrace learning, modularity, and human-centered design to remain viable in increasingly interconnected and volatile operational landscapes. The work presented here provides a foundation for such systems—both theoretically and practically—and offers a roadmap for

future researchers and practitioners aiming to build adaptive, resilient, and ethically aligned SoS architectures.

10.8 Chapter Summary

This chapter synthesized insights from seven domain-specific case studies to highlight generalizable principles, methodological trade-offs, and practical implications of integrating Machine Learning (ML) into System-of-Systems (SoS) environments. By comparing diverse domains—including semiconductor manufacturing, healthcare, smart cities, and circular economy marketplaces—the chapter identified both the potential and limitations of ML-driven coordination architectures in complex, distributed settings.

Key insights included:

- **Coordination Models:** Hierarchical and decentralized architectures were consistently more effective than purely centralized models, especially in dynamic, resource-constrained environments such as healthcare and smart cities.
- **Cross-Domain Patterns:** Learning convergence, policy adaptability, and subsystem cooperation emerged as recurring patterns across domains. Scalability and responsiveness were enhanced by modular agent design and reward augmentation strategies.
- **Design Trade-offs:** Trade-offs between accuracy and interpretability, computational efficiency and generalizability, and local versus global optimization shaped model selection and deployment strategies in each case study.
- **Technical Limitations:** Challenges related to synthetic versus real-world data, model complexity, and multi-agent scalability underscored the need for continued methodological refinement and empirical validation.
- **Human and Ethical Considerations:** Explainability, human-in-the-loop integration, and fairness emerged as central concerns in the deployment of ML-SoS systems, particularly in high-stakes or public-facing domains.

- **Future Directions:** The findings support the evolution of SoS design toward more adaptive, learning-enabled frameworks that integrate modular ML architectures with domain-specific knowledge and stakeholder feedback.

Taken together, these findings position the ML-SoS integration strategy as a flexible and scalable foundation for future system engineering initiatives. The diversity of case studies validates the framework's adaptability across domains, while the observed challenges highlight critical areas for ongoing research and refinement.

The concluding chapter of the dissertation builds on these insights by summarizing the research contributions, revisiting the research questions, and outlining opportunities for future theoretical advancement and real-world implementation.

Chapter 11

Conclusion

This concluding chapter reflects on the overarching contributions, findings, and implications of the research presented in this dissertation. It synthesizes insights across diverse application domains, revisits the research questions, and discusses how the proposed frameworks advance the integration of Machine Learning (ML) and System-of-Systems (SoS) engineering. In addition to highlighting key outcomes, this chapter outlines the theoretical, methodological, and practical contributions to the fields of ML, systems engineering, and domain-specific practice. It concludes by identifying open questions and presenting a roadmap for future research in ML-enabled SoS design and deployment.

11.1 Research Questions and Key Findings

This dissertation investigated how Machine Learning (ML) can be effectively integrated with System-of-Systems (SoS) engineering principles to improve coordination, adaptability, and decision support in complex and decentralized domains. Through the development and evaluation of ML-SoS frameworks across seven case studies, the research demonstrated how local intelligence and global coherence can be achieved simultaneously through reward-aligned learning agents, predictive models, and optimization techniques.

Restatement of Research Objectives

The central objectives of this research were fourfold: (1) to design ML-based architectures that enable scalable coordination and autonomy in SoS environments, (2) to operationalize these architectures through simulation and agent-based modeling across diverse domains, (3) to evaluate their performance using empirical and synthetic datasets, and (4) to generalize theoretical and methodological insights across application contexts. These goals were pursued using a combination of supervised learning, reinforcement learning, system dynamics

modeling, and natural language processing, embedded within a unifying SoS coordination framework.

Case Study Outcomes and Insights

The dissertation presented seven case studies, each grounded in a domain where decentralized decision-making and multi-subsystem optimization are critical:

- **3D-IC Manufacturing:** An SoS-driven ML framework integrating Random Forest, CNN, and LSTM models improved global yield from 77.41% to 80.41% by aligning local defect detection, TSV optimization, and fault prediction subsystems.
- **Hierarchical MARL for Semiconductor Coordination:** A three-level reward architecture enabled decentralized agents to align with system-wide objectives, showcasing stable convergence, conflict resolution, and resource adaptability in a simulated fab environment.
- **SME Manufacturing Optimization:** A hybrid framework combining system dynamics, linear programming, and ML analytics enhanced cost-efficiency, inventory control, and production agility in low-volume, resource-constrained environments.
- **NLP for Quality Control:** Text analytics and Naïve Bayes classification were applied to technician-generated reports, successfully transforming unstructured data into actionable insights for PCB defect diagnosis and resolution.
- **AIoMT-Enabled Healthcare SoS:** A MARL framework simulated coordination across hospitals, telemedicine, and wearable systems under four SoS configurations. Decentralized models outperformed centralized ones in cumulative rewards, cooperation, and adaptability.
- **Smart City Infrastructure Optimization:** A reward-augmented MARL framework improved energy efficiency, transport scheduling, and emergency responsiveness by 12.5–25% through cooperation among heterogeneous agents under dynamic conditions.

- **Circular Economy Marketplaces:** Forecasting and Q-learning models improved price stability and demand-supply alignment in digital CE platforms, supporting sustainability and resource efficiency.

Each case study validated aspects of the proposed ML-SoS architecture—whether through subsystem alignment, agent coordination, model interpretability, or policy convergence—demonstrating its adaptability and relevance across distinct operational contexts.

Cross-Domain Integration and Generalization

Beyond domain-specific findings, this dissertation established a generalizable framework for applying ML in complex SoS settings. Common architectural elements—such as hierarchical reward structures, coordination-sensitive metrics, and decentralized learning agents—proved effective across diverse applications. Insights from one domain (e.g., conflict resolution in healthcare) translated meaningfully to others (e.g., agent coordination in smart cities), affirming the cross-cutting value of integrated ML-SoS methodologies.

The results collectively highlight that scalable, learning-driven coordination is feasible even in environments with heterogeneous agents, incomplete information, and dynamic goals. By embedding ML techniques within structured SoS designs, the dissertation contributes to both foundational theory and practical system engineering.

11.2 Contributions to ML, SE, and Practice

This dissertation contributes to the advancement of theory, methodology, and application at the intersection of Machine Learning (ML), Systems Engineering, and domain-specific practices. By integrating ML architectures within a System-of-Systems (SoS) framework, the research addresses longstanding challenges in distributed coordination, local-global reward alignment, and intelligent decision support in complex environments.

Advancements in Machine Learning for System-of-Systems

A key contribution lies in extending ML methodologies—particularly reinforcement learning, supervised modeling, and predictive analytics—into the structural and operational complexities of SoS environments. The following advancements were demonstrated:

- **Hierarchical Reward Design:** The introduction of a three-tier reward system (global, main-agent, sub-agent) enables policy learning that is both autonomous and globally coherent. This addresses known limitations in MARL regarding local optima and reward misalignment.
- **Multi-Agent Coordination with Dynamic Feedback:** The research formalized feedback-adjusted learning, allowing agents to recalibrate based on evolving global performance metrics. This innovation supports adaptive cooperation in environments with stochastic dynamics and inter-agent dependencies.
- **Cross-Modal ML Integration:** The dissertation demonstrated the effective combination of Naïve Bayes, Random Forests, LSTMs, CNNs, and Q-learning models within larger coordination structures. This hybridization illustrates how diverse ML techniques can contribute to decision-making across temporal, spatial, and semantic dimensions.

These contributions push the boundaries of traditional ML applications by embedding learning agents within structured, interdependent, and hierarchical environments—characteristics that are increasingly typical of real-world digital systems.

Integration of ML with Systems Engineering Methodologies

From a systems engineering perspective, the research offers methodological innovations that bridge classical systems design principles with data-driven intelligence:

- **System-of-Systems Modeling with Agent Abstractions:** Each case study models complex environments as a coordinated network of agents, enabling modular system

representation and scalable optimization strategies that extend beyond monolithic control.

- **Performance Evaluation Across Scales:** The framework incorporates quantitative performance metrics (e.g., yield, coordination scores, reward convergence) and qualitative indicators (e.g., cooperation dynamics, conflict resolution efficacy) across agent, subsystem, and system levels.
- **Operationalization via Simulation-Based Design:** A structured simulation architecture was developed to implement and test SoS-ML frameworks under variable assumptions, enabling systematic evaluation before real-world deployment.

These methodological contributions position ML as a formal and integrative component of systems engineering, capable of supporting lifecycle-aware design, policy-driven coordination, and real-time decision support.

Domain-Specific Innovations in Semiconductor, Healthcare, Smart Cities, and CE

By operationalizing the ML-SoS framework across seven applied case studies, the research generated domain-specific contributions with practical relevance:

- **Semiconductor Manufacturing:** Novel integration of supervised ML (Random Forest, CNN, LSTM) for early-stage defect mitigation, TSV optimization, and electrical fault detection improved global yield by over 3% in a simulated 3D-IC fab. A hierarchical MARL model further demonstrated decentralized process coordination in constituent subsystems.
- **Healthcare Systems:** The application of MARL in AIoMT-enabled care networks revealed the performance advantages of decentralized SoS configurations (e.g., Acknowledged and Collaborative) in managing hospital triage, remote monitoring, and patient transport under resource constraints.

- **Smart Cities:** A reward-augmented MARL architecture improved system-wide coordination among energy, mobility, and safety agents, demonstrating emergent behavior, adaptability, and alignment with policy objectives in dynamic urban environments.
- **Circular Economy Marketplaces:** ML-based forecasting and reinforcement learning enabled adaptive pricing and demand-supply matching in digital CE platforms. The framework supported sustainable resource allocation, emphasizing fairness, explainability, and environmental resilience.
- **Healthcare Systems:** The application of MARL in AIoMT-enabled care networks revealed the performance advantages of decentralized SoS configurations (e.g., Acknowledged and Collaborative) in managing hospital triage, remote monitoring, and patient transport under resource constraints.
- **Smart Cities:** A reward-augmented MARL architecture improved system-wide coordination among energy, mobility, and safety agents, demonstrating emergent behavior, adaptability, and alignment with policy objectives in dynamic urban environments.
- **Circular Economy Marketplaces:** ML-based forecasting and reinforcement learning enabled adaptive pricing and demand-supply matching in digital CE platforms. The framework supported sustainable resource allocation, emphasizing fairness, explainability, and environmental resilience.

Each of these innovations demonstrates how ML-SoS coordination frameworks can be adapted to domain-specific requirements while maintaining a coherent architectural foundation. The cumulative effect is a cross-sectoral model for ML adoption in distributed, dynamic, and critical systems.

11.3 Broader Implications for ML in Complex Systems

The integration of Machine Learning (ML) into critical infrastructure and complex System-of-Systems (SoS) architectures presents transformative opportunities and nuanced challenges. This dissertation demonstrates that when designed with structural awareness, ML

frameworks can deliver scalable, adaptive, and resilient coordination across heterogeneous systems. However, successful real-world implementation also requires addressing ethical, human-centered, and policy considerations that extend beyond algorithmic performance.

Scalability, Adaptability, and Resilience

The proposed ML-SoS frameworks show promising scalability, enabling their application across diverse domains—semiconductor fabs, hospitals, smart cities, and digital marketplaces. Key attributes contributing to scalability and resilience include:

- **Modular Architecture:** The use of agent-based abstractions allows the framework to scale from a few to hundreds of subsystems or entities, facilitating deployment in both centralized and distributed settings.
- **Dynamic Reward Structures:** The hierarchical reward mechanism supports real-time recalibration based on global system states, allowing agents to adapt their behavior under shifting priorities or constraints.
- **Robust Learning Under Uncertainty:** Techniques such as MARL, ensemble learning, and feedback-aligned optimization enhance resilience against noise, partial observability, and changing operational environments.
- **Cross-Domain Generalization:** Case studies demonstrate that the architectural principles of the ML-SoS model—local autonomy, coordination logic, and layered evaluation—are transferable to various infrastructure and policy contexts.

Together, these features support robust decision-making in volatile, uncertain, complex, and ambiguous (VUCA) systems, a defining characteristic of modern cyber-physical and socio-technical environments.

Ethics, Explainability, and Human-Centered Design

Deploying ML in safety-critical and high-impact systems introduces ethical and human-centric challenges that require deliberate attention:

- **Explainability and Trust:** Interpretability remains essential, especially in manufacturing and healthcare, where ML-driven recommendations must be transparent to operators and stakeholders. The use of interpretable models (e.g., Naïve Bayes, Random Forests) and visualization techniques (e.g., SHAP, t-SNE) supports auditability and trust.
- **Fairness and Bias Mitigation:** In domains like circular economy marketplaces or smart city infrastructure, algorithmic decisions can influence access to resources and services. Strategies such as data balancing, ensemble voting, and fairness-aware training were used to reduce systemic bias and promote equity.
- **Human-in-the-Loop Integration:** Several case studies underscore the importance of combining automated learning with human judgment. For example, in the PCB defect analysis, technician narratives were embedded into ML workflows, ensuring the interpretive value of unstructured human inputs is preserved.
- **Safety and Failsafe Mechanisms:** MARL deployments in healthcare and urban systems must incorporate failover protocols, ethical alignment layers, or override mechanisms to prevent unsafe outcomes during model uncertainty or unexpected conditions.

These ethical considerations not only safeguard system integrity but also shape public perception and adoption, especially in regulated or public-interest sectors.

Policy and Operational Implications for Real-World Deployment

Translating ML-SoS frameworks into operational infrastructures requires engagement with institutional, regulatory, and organizational systems:

- **Data Governance and Interoperability:** Cross-organizational deployment—such as between hospitals and telemedicine platforms, or across smart city subsystems—requires standardized data formats, secure sharing protocols, and interoperable APIs.

- **Policy Alignment and Regulatory Compliance:** ML frameworks must be aligned with local policies and legal requirements, particularly in healthcare (e.g., patient privacy, triage fairness) and environmental systems (e.g., sustainability regulations).
- **Operational Readiness and Change Management:** Effective adoption demands organizational capacity building—training, stakeholder alignment, and phased integration with legacy systems. Simplified user interfaces, modular rollouts, and performance benchmarking can lower adoption barriers.
- **Public-Private Partnerships:** Large-scale deployments of smart systems, healthcare networks, or CE platforms benefit from shared governance models, where public policy goals are pursued alongside private sector innovation.

These implications extend the dissertation's impact beyond academic modeling to inform future strategies for infrastructure modernization, responsible AI deployment, and digitally enhanced decision-making ecosystems.

11.4 Open Questions and Roadmap for Future Work

While this dissertation has established a robust framework for integrating Machine Learning (ML) into System-of-Systems (SoS) coordination, several open questions and future research directions remain. These avenues highlight the need for continued exploration into real-world applicability, architectural innovation, and standardization for scalable, ethical, and effective ML-enabled SoS solutions.

Empirical Validation and Real-Time Integration

Most of the evaluation in this dissertation was conducted using simulated environments with synthetic or semi-synthetic data. While this enabled controlled experimentation and comparative analysis, real-world deployment scenarios may exhibit unanticipated complexities, including sensor drift, policy feedback loops, and unpredictable human behaviors.

Future work must focus on validating these ML-SoS frameworks in operational settings, including smart manufacturing plants, hospital networks, or urban infrastructure systems.

Additionally, integrating real-time data pipelines and low-latency inference mechanisms remains a crucial next step. The development of edge-compatible, fault-tolerant ML agents capable of responding to time-sensitive scenarios—such as energy grid surges or emergency medical triage—is essential for deploying SoS architectures at scale.

Hybrid AI Architectures and Multi-Objective Optimization

Several promising directions exist for expanding the architectural sophistication of ML-SoS frameworks. Hybrid architectures that combine reinforcement learning, supervised learning, symbolic reasoning, and optimization techniques could offer enhanced adaptability and generalization across domains. For example, embedding meta-learning or curriculum learning within SoS agents may accelerate policy convergence and improve resilience in non-stationary environments.

Moreover, the case studies primarily focused on single-objective formulations (e.g., yield maximization, cost minimization). Future research should prioritize multi-objective optimization strategies, particularly those balancing performance, fairness, explainability, and sustainability. Pareto-efficient learning algorithms and reward shaping mechanisms may play a pivotal role in advancing this line of work.

Cross-Sectoral Frameworks and Interoperability Standards

As ML-SoS frameworks gain adoption across sectors—semiconductors, healthcare, smart cities, and circular economies—there is a pressing need for interoperable design principles and cross-sector coordination standards. These may include ontologies for agent roles, shared communication protocols, trust management systems, and APIs for modular integration.

Collaborative research involving policymakers, domain experts, and AI system designers will be essential to ensure that these frameworks are both technically robust and socially

aligned. The development of reference implementations and open-source toolkits could also accelerate adoption and ensure reproducibility across applications.

Long-Term Vision for ML-Enabled System-of-Systems Design

Looking forward, the integration of ML into SoS should not only aim to optimize system performance, but also to reshape the design philosophy of complex infrastructures. This includes designing systems that are not only reactive but anticipatory—capable of self-assessment, model updating, and strategic realignment in response to external shocks.

The vision of "self-governing" SoS composed of intelligent, adaptive agents opens new opportunities in sustainability, resilience, and social equity—but it also raises new challenges in governance, transparency, and oversight. Future research must engage with these broader sociotechnical questions, ensuring that the evolution of ML-SoS design remains inclusive, explainable, and aligned with human values.

11.5 Closing Remarks

This dissertation has explored the intersection of Machine Learning (ML) and System-of-Systems (SoS) engineering, presenting a unified framework for coordinating autonomous, intelligent subsystems across a diverse range of domains. From semiconductor manufacturing and quality control to healthcare, smart cities, and circular economy marketplaces, the research demonstrated that ML—when embedded within scalable SoS architectures—can enhance adaptability, efficiency, and decision quality in complex, dynamic environments.

Throughout the research journey, each case study illuminated distinct technical challenges, modeling strategies, and domain-specific constraints. Yet, common patterns emerged: the importance of reward alignment in agent-based systems, the trade-offs between interpretability and performance, and the critical role of hierarchical coordination in managing subsystem interactions. These insights informed the development of generalizable design

principles and highlighted the transformative potential of ML-SoS integration beyond any single application.

Beyond the empirical results and architectural contributions, this dissertation reflects a broader aspiration: to enable systems that are not only intelligent, but collaborative—capable of learning, adapting, and working in concert toward shared objectives in real time. The future of infrastructure, manufacturing, and public services increasingly depends on such systems, where distributed intelligence and collective optimization are prerequisites for resilience and sustainability.

As the field continues to evolve, new challenges will arise in areas such as human-centered AI, ethical alignment, policy translation, and real-time deployment. But the foundations laid here—both in theory and implementation—offer a strong starting point for researchers and practitioners seeking to build the next generation of adaptive, intelligent, and responsible SoS.

Ultimately, this work aims to contribute not only to technical progress but also to a broader understanding of how ML can be responsibly integrated into the fabric of complex societal systems. As we design for autonomy, we must also design for accountability. As we optimize for performance, we must also optimize for purpose.

The road ahead is one of continued experimentation, iteration, and interdisciplinary collaboration. The promise of ML-enabled SoS lies not just in smarter systems, but in systems that are more aligned—with each other, with human needs, and with the uncertain futures they are built to navigate.

Bibliography

[1] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998, doi: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.

[2] D. DeLaurentis, "Understanding transportation as a system-of-systems design problem," in *43rd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2005, doi: 10.2514/6.2005-123.

[3] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*, K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, Eds. Springer International Publishing, pp. 321–384, doi: 10.1007/978-3-030-60990-0_12.

[4] A. Sheikh and E. K. P. Chong, "Advancing aiomt-enabled healthcare system-of-systems using multi-agent reinforcement learning," *IEEE Access*, pp. 1–1, 2025, doi: 10.1109/ACCESS.2025.3596921.

[5] A. A. Sheikh and E. K. P. Chong, "Multi-agent reinforcement learning framework for optimizing smart cities as system of systems," *Systems Engineering*, e70006. doi: 10.1002/sys.70006.

[6] C. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "1. systems of systems engineering: Basic concepts, model-based techniques, and research directions," *ACM Computing Surveys*, 2015, doi: 10.1145/2794381.

[7] P. Gu, Z. Chen, Y. Zhang, K. Xie, C. Zhao, F. Ye, and Y. Tao, "2. x-sem: A modeling and simulation-based system engineering methodology," *Journal of Manufacturing Systems*, 2024, doi: 10.1016/j.jmsy.2024.01.013.

[8] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008, doi: 10.1109/TSMCC.2007.913919.

[9] A. K. Raz, M. Bhuyian, J. L. Bricio-Neto, C. d. Santos, and D. T. Maxwell, "Conceptual, mathematical, and analytical foundations for mission engineering and system of systems analysis," *IEEE Systems Journal*, pp. 1 – 11, publisher: Institute of Electrical and Electronics Engineers. doi: 10.1109/jsyst.2024.3409231.

[10] F. Petitdemange, I. Borne, and J. Buisson, "Assisting the evolutionary development of SoS with reconfiguration patterns," *European Conference on Software Architecture*, p. 9, publisher: ACM. doi: 10.1145/2993412.3004845.

[11] C. Shankar and R. H. Campbell, "Managing pervasive systems using role-based obligation policies," *IEEE International Conference on Pervasive Computing and Communications*, pp. 373–377, publisher: IEEE. doi: 10.1109/PERCOMW.2006.77.

[12] W. K. Edwards, "Policies and roles in collaborative applications," *Conference on Computer Supported Cooperative Work*, pp. 11 – 20, publisher: ACM. doi: 10.1145/240080.240175.

[13] P. Gu, Z. Chen, Y. Zhang, K. Xie, C. Zhao, F. Ye, and Y. Tao, "X-SEM: A modeling and simulation-based system engineering methodology," *Journal of Manufacturing Systems*, doi: 10.1016/j.jmsy.2024.01.013.

[14] R. Bemthuis, M.-E. Iacob, and P. J. M. Havinga, "A design of the resilient enterprise: A reference architecture for emergent behaviors control," *Sensors*, vol. 20, no. 22, p. 6672, publisher: Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/S20226672.

[15] A. Sheikh and E. K. P. Chong, "Artificial intelligence-driven optimization for three-dimensional integrated circuit manufacturing: A system of systems framework," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, pp. 1–1, 2025, doi: 10.1109/TCPMT.2025.3549707.

[16] P.-N. Hsu, K.-C. Shie, K.-P. Chen, J.-C. Tu, C.-C. Wu, N.-T. Tsou, Y.-C. Lo, N.-Y. Chen, Y.-F. Hsieh, M. Wu, C. Chen, and K.-N. Tu, "Artificial intelligence deep learning for 3D IC reliability prediction," *Scientific Reports*, vol. 12, no. 1, p. 6711, 2022, doi: 10.1038/s41598-022-08179-z.

[17] Y. Hou, C. Lu, W. Abbas, M. Seid Ibrahim, M. Waseem, H. Hung Lee, and K.-H. Loo, "Precursor prediction and early warning of power mosfet failure using machine learning with model uncertainty considered," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 12, no. 6, pp. 5762–5776, 2024, doi: 10.1109/JESTPE.2024.3476980.

[18] A. A. Sheikh, E. K. P. Chong, and S. J. Simske, "Enhancing defect detection in circuit board assembly using ai and text analytics for component failure classification," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 14, no. 10, pp. 1881–1890, 2024, doi: 10.1109/TCPMT.2024.3453597.

[19] M. Nandagopal, K. Seerangan, T. Govindaraju, N. E. Abi, B. Balusamy, and S. Selvarajan, "A deep auto-optimized collaborative learning (dacl) model for disease prognosis using ai-iomt systems," *Dental science reports*, vol. 14, 2024, doi: 10.1038/s41598-024-59846-2.

[20] A. A. Sheikh, S. J. Simske, and E. K. P. Chong, "Evaluating artificial intelligence models for resource allocation in circular economy digital marketplace," *Sustainability*, vol. 16, no. 23, 2024, paper 10601. doi: 10.3390/su162310601.

[21] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, 1998, pp. 41–48, url: https://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-006.pdf.

[22] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, Mar 2002, url: https://dl.acm.org/doi/10.1145/505282.505283.

[23] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999, url: https://mitpress.mit.edu/books/foundations-statistical-natural-language-processing.

[24] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2008, url: https://web.stanford.edu/~jurafsky/slp3/.

[25] A. Rammal, K. Ezukwoke, A. Hoayek, and M. Batton-Hubert, "Unsupervised variable selection using a genetic algorithm: An application to textual data," in *2022 International Conference on Smart Systems and Power Management (IC2SPM)*, pp. 11–19, doi: 10.1109/IC2SPM56638.2022.9989008.

[26] X. Wu, H. Wang, D. Wei, and M. Shi, "ANFIS with natural language processing and gray relational analysis based cloud computing framework for real time energy efficient resource allocation," *Computer Communications*, vol. 150, pp. 122–130, doi: 10.1016/j.comcom.2019.11.015.

[27] R. S. Sutton and A. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning. The MIT Press.

[28] A. Louati, H. Louati, E. Kariri, W. Neifar, M. K. Hassan, M. H. H. Khairi, M. A. Farahat, and H. M. El-Hoseny, "3. sustainable smart cities through multi-agent reinforcement learning-based cooperative autonomous vehicles," *Sustainability*, 2024, doi: 10.3390/su16051779.

[29] Y. Li, Y. Zhang, X. Li, and C. Sun, "1. regional multi-agent cooperative reinforcement learning for city-level traffic grid signal control," *IEEE/CAA Journal of Automatica Sinica*, 2024, doi: 10.1109/jas.2024.124365.

[30] A. A. Sheikh and E. K. P. Chong, "Optimizing semiconductor manufacturing for small and medium enterprises: A system-dynamics and machine learning approach," *Qeios*, 2025, doi: 10.32388/EOQ6MJ.2.

[31] R. Sepehrzad, A. S. G. Langeroudi, A. Khodadadi, S. Adinehpour, A. Al-Durra, and A. Anvari-Moghaddam, "4. an applied deep reinforcement learning approach to control active networked microgrids in smart cities with multi-level participation of battery energy storage system and electric vehicles," *Sustainable Cities and Society*, 2024, doi: 10.1016/j.scs.2024.105352.

[32] A. A. Kutty, T. G. Wakjira, M. Kucukvar, G. M. Abdella, and N. C. Onat, "Urban resilience and livability performance of european smart cities: A novel machine learning approach," *Journal of Cleaner Production*, vol. 378, p. 134203, 2022, doi: 10.1016/j.jclepro.2022.134203.

[33] Y. Sha, Z. He, H. Gutierrez, J. Du, W. Yang, and X. Lu, "The intelligent detection method for flip chips using CBN-s-net algorithm with SAM images," *Journal of Manufacturing Processes*, vol. 83, pp. 60–67, 2022, doi: https://doi.org/10.1016/j.jmapro.2022.08.058.

[34] P. Paulachan, J. Siegert, I. Wiesler, and R. Brunner, "An end-to-end convolutional neural network for automated failure localisation and characterisation

of 3D interconnects," *Scientific Reports*, vol. 13, no. 1, p. 9376, 2023, doi: 10.1038/s41598-023-35048-0.

[35] Y. Xu, K. Feng, X. Yan, X. Sheng, B. Sun, Z. Liu, and R. Yan, "Cross-modal fusion convolutional neural networks with online soft-label training strategy for mechanical fault diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 1, pp. 73–84, 2024, doi: 10.1109/TII.2023.3256400.

[36] J. E. Choi, D. H. Seol, C.-Y. Kim, and S. J. Hong, "Generative adversarial network-based fault detection in semiconductor equipment with class-imbalanced data," *Sensors*, vol. 23, no. 4, 2023, article ID: 1889. doi: 10.3390/s23041889.

[37] J. Wang, P. Gao, J. Zhang, C. Lu, and B. Shen, "Knowledge augmented broad learning system for computer vision based mixed-type defect detection in semiconductor manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 81, p. 102513, doi: 10.1016/j.rcim.2022.102513.

[38] T. Schlosser, M. Friedrich, F. Beuth, and D. Kowerko, "Improving automated visual fault inspection for semiconductor manufacturing using a hybrid multistage system of deep neural networks," *Journal of Intelligent Manufacturing*, vol. 33, no. 4, pp. 1099–1123, 2022, doi: 10.1007/s10845-021-01906-9.

[39] F. Duan, Z. Lv, S. Chen, and H. Chen, "A short review of through-silicon via (tsv) interconnects: Metrology and analysis," *Applied Sciences*, 2023, doi: 10.3390/app13148301.

[40] D. Karnam, Y.-L. Lo, and C.-H. Yang, "Simulation study and parameter optimization of laser TSV using artificial neural networks," *Journal of Materials Research and Technology*, vol. 25, pp. 3712–3727, 2023, doi: 10.1016/j.jmrt.2023.06.199.

[41] Q. Zheng, Y. Zhang, H. Tian, and L. He, "A cooperative adaptive genetic algorithm for reentrant hybrid flow shop scheduling with sequence-dependent setup time and

limited buffers," *Complex & Intelligent Systems*, vol. 10, no. 1, pp. 781–809, 2024, doi: 10.1007/s40747-023-01147-8.

[42] A. Größler, "System dynamics modelling as an inductive and deductive endeavour. comment on the paper by schwaninger and grösser," *Systems Research and Behavioral Science*, vol. 25, no. 4, pp. 467–470, 2008, doi: 10.1002/SRES.908.

[43] E. Malbon and J. Parkhurst, "System dynamics modelling and the use of evidence to inform policymaking," *Policy Studies*, vol. 44, no. 4, pp. 454–472, 2022, doi: 10.1080/01442872.2022.2080814.

[44] C. Han and Q. Zhang, "Optimization of supply chain efficiency management based on machine learning and neural network," *Neural Computing and Applications*, vol. 33, no. 5, pp. 1419–1433, 2021, doi: 10.1007/S00521-020-05023-1.

[45] H. D. Perez, S. Amaran, E. Erisen, J. M. Wassick, and I. E. Grossmann, "Optimization of extended business processes in digital supply chains using mathematical programming," *Computers & Chemical Engineering*, vol. 152, 2021, doi: 10.1016/J.COMPCHEMENG.2021.107323.

[46] D. Jiang, W. Lin, and N. Raghavan, "A novel framework for semiconductor manufacturing final test yield classification using machine learning techniques," *IEEE Access*, vol. 8, pp. 197 885–197 895, 2020, doi: 10.1109/ACCESS.2020.3034680.

[47] A. Rammal, K. Ezukwoke, A. Hoayek, and M. Batton-Hubert, "Root cause prediction for failures in semiconductor industry, a genetic algorithm–machine learning approach," *Dental Science Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-30769-8.

[48] C.-F. Chien and C.-C. Chen, "Adaptive parametric yield enhancement via collinear multivariate analytics for semiconductor intelligent manufacturing," *Applied Soft Computing*, vol. 108, 2021, doi: 10.1016/J.ASOC.2021.107385.

[49] J. Mondschein, J. W. Welburn, and D. Gonzales, *Securing the Microelectronics Supply Chain: Four Policy Issues for the U.S. Department of Defense to Consider*. RAND Corporation, 2022, doi: 10.7249/pea1394-1.

[50] M. Crawford, T. Telesco, C. Nelson, and B. Botwin, "Defense industrial base assessment: U.s. integrated circuit design and fabrication capability," *Department of Commerce's Office of Technology Evaluation*, 2009, survey and data collection on U.S. IC design and fabrication capabilities.

[51] M. A. Mak, "Trusted defense microelectronics: Future access and capabilities are uncertain," *GAO Reports*, 2015.

[52] J. Wang, S. Gao, Z. Tang, D. Tan, B. Cao, and J. Fan, "A context-aware recommendation system for improving manufacturing process modeling," *Journal of Intelligent Manufacturing*, vol. 34, no. 3, pp. 1347–1368, doi: 10.1007/s10845-021-01854-4.

[53] K.-J. Kim, K.-J. Kim, C.-H. Jun, I.-G. Chong, and G.-Y. Song, "Variable selection under missing values and unlabeled data in semiconductor processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 1, pp. 121–128, conference Name: IEEE Transactions on Semiconductor Manufacturing. doi: 10.1109/TSM.2018.2881286.

[54] "Deep analysis of power equipment defect based on semantic framework text mining technology," *CSEE Journal of Power and Energy Systems*, doi: 10.17775/CSEEJPES.2019.00210.

[55] C. C. Ng, A. K. B. Nazaruddin, Y. K. Lee, X. Wang, Y. Liu, C. S. Chan, L. Jin, Y. Sun, and L. Fan, "ICDAR 2021 competition on integrated circuit text spotting and aesthetic assessment," in *Document Analysis and Recognition – ICDAR 2021*,

ser. Lecture Notes in Computer Science, J. Lladós, D. Lopresti, and S. Uchida, Eds. Springer International Publishing, pp. 663–677, doi: 10.1007/978-3-030-86337-1_44.

[56] Z. Kong, C. Yue, Y. Shi, J. Yu, C. Xie, and L. Xie, "Entity extraction of electrical equipment malfunction text by a hybrid natural language processing algorithm," *IEEE Access*, vol. 9, pp. 40 216–40 226, conference Name: IEEE Access. doi: 10.1109/ACCESS.2021.3063354.

[57] Z. Wang, K. Ezukwoke, A. Hoayek, M. Batton-Hubert, and X. Boucher, "NLP based on GCVAE for intelligent fault analysis in semiconductor industry," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, doi: 10.1109/ETFA52439.2022.9921524.

[58] M. Kolat, B. Kovari, T. Becsi, and S. Aradi, "5. multi-agent reinforcement learning for traffic signal control: A cooperative approach," *Sustainability*, 2023, doi: 10.3390/su15043479.

[59] H. Roberts, J. Zhang, B. Bariach, J. Cowls, B. Gilburt, P. Juneja, A. Tsamados, M. Ziosi, M. Taddeo, and L. Floridi, "Artificial intelligence in support of the circular economy: Ethical considerations and a path forward," *AI & Society*, Nov. 2022, doi: 10.1007/s00146-022-01596-8.

[60] J. Walzberg, J.-M. Frayret, A. L. Eberle, A. Carpenter, and G. Heath, "Agent-based modeling and simulation for the circular economy: Lessons learned and path forward," *Journal of Industrial Ecology*, vol. 27, no. 5, pp. 1227–1238, 2023, publisher: International Society for Industrial Ecology. doi: 10.1111/jiec.13423.

[61] H. Onyeaka, P. Tamasiga, U. M. Nwauzoma, T. Miri, U. C. Juliet, O. Nwaiwu, and A. A. Akinsemolu, "Using artificial intelligence to tackle food waste and enhance the circular economy: Maximising resource efficiency and minimising environmental

impact—a review," *Sustainability*, vol. 15, no. 13, p. 10482, 01 2023, publisher: Multidisciplinary Digital Publishing Institute. doi: 10.3390/su151310482.

[62] M. S. Pathan, E. Richardson, E. Galvan, and P. Mooney, "The role of artificial intelligence within circular economy activities—a view from ireland," *Sustainability*, vol. 15, no. 12, p. 9451, 01 2023, publisher: Multidisciplinary Digital Publishing Institute. doi: 10.3390/su15129451.

[63] M. P. Lamela, J. Rodriguez-Molina, M. Martinez-Nunez, and J. Garbajosa, "A blockchain-based decentralized marketplace for trustworthy trade in developing countries," *IEEE Access*, vol. 10, pp. 79 100–79 123, 2022, publisher: Institute of Electrical and Electronics Engineers (IEEE).

[64] L. Mikkelsen, K. Mortensen, H. Rasmussen, H.-P. Schwefel, and T. Madsen, "Realization and evaluation of marketplace functionalities using ethereum blockchain," in *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*. IEEE, Dec. 2018.

[65] F. Hu, Y. Deng, and A. H. Aghvami, "5. scalable multi-agent reinforcement learning for dynamic coordinated multipoint clustering," *IEEE Transactions on Communications*, 2022, doi: 10.1109/TCOMM.2022.3220870.

[66] Y. Zhang, Z. Yu, J. Zhang, L. Wang, T. H. Luan, B. Guo, and C. Yuen, "1. learning decentralized traffic signal controllers with multi-agent graph reinforcement learning," *IEEE Transactions on Mobile Computing*, 2023, doi: 10.1109/tmc.2023.3332081.

[67] F. Galtier, F. Bousquet, M. Antona, and P. Bommel, "Markets as communication systems," *Journal of Evolutionary Economics*, vol. 22, no. 1, pp. 161–201, 01 01, 2012, doi: 10.1007/s00191-011-0225-5.

[68] A. Sheikh, E. K. P. Chong, and S. J. Simske, "Enhancing defect detection in circuit board assembly using AI and text analytics for component failure classification,"

IEEE Transactions on Components, Packaging and Manufacturing Technology, pp. 1–1, doi: 10.1109/TCPMT.2024.3453597.

[69] I. Sakai, N. Sakurai, and T. Ohiwa, "High rate deep si etching for through-silicon via applications," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 29, no. 2, 2011, paper 021009. doi: 10.1116/1.3543635.

[70] S. L. Burkett, M. B. Jordan, R. P. Schmitt, L. A. Menk, and A. E. Hollowell, "Tutorial on forming through-silicon vias," *Journal of Vacuum Science & Technology A*, vol. 38, no. 3, p. 031202, 2020, doi: 10.1116/6.0000026.

[71] J. H. Lau, "TSV manufacturing yield and hidden costs for 3D IC integration," in *2010 Proceedings 60th Electronic Components and Technology Conference (ECTC)*. IEEE, 2010, pp. 1031–1042, doi: 10.1109/ECTC.2010.5490828.

[72] "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[73] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[74] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[75] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[76] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, doi: 10.1162/neco.1997.9.8.1735.

[77] Haibo He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.

[78] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.

[79] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *ArXiv*, vol. abs/2010.16061, 2011, url: https://api.semanticscholar.org/CorpusID:3770261.

[80] S. Theodoridis, "Chapter 8 - parameter learning: a convex analytic path," in *Machine Learning (Second Edition)*, second edition ed., S. Theodoridis, Ed. Academic Press, 2020, pp. 351–425, doi: https://doi.org/10.1016/B978-0-12-818803-3.00017-9.

[81] Y. Huangfu, S. Habibi, and A. Wassyng, "System failure detection using deep learning models integrating timestamps with nonuniform intervals," *IEEE Access*, vol. 10, pp. 17 629–17 640, 2022, doi: 10.1109/ACCESS.2022.3150342.

[82] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. The MIT press, 2016.

[83] V. G. Kulkarni, *Modeling and Analysis of Stochastic Systems*, 2nd ed., ser. Chapman & Hall/CRC Texts in Statistical Science. Boca Raton, FL, USA: CRC Press, 2009, url: https://www.taylorfrancis.com/books/mono/10.1201/9781315367910/modeling-analysis-stochastic-systems-vidyadhar-kulkarni.

[84] S. P. Boyd and L. Vandenberghe, *Convex optimization*, version 29 ed. Cambridge University Press, 2023.

[85] T. Ni, Z. Yang, H. Chang, X. Zhang, L. Lu, A. Yan, Z. Huang, and X. Wen, "A novel TDMA-based fault tolerance technique for the TSVs in 3D-ICs using honeycomb

topology," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 724–734, 2021, doi: 10.1109/TETC.2020.2969237.

[86] S. Zhou, S. Yao, T. Shen, and Q. Wang, "A novel end-to-end deep learning framework for chip packaging defect detection," *Sensors*, vol. 24, no. 17, 2024, article ID: 5837. doi: 10.3390/s24175837.

[87] W. Li, C. Hao, S. He, C. Qiu, H. Liu, Y. Xu, B. Li, X. Tan, and F. Peng, "Multi-agent reinforcement learning method for cutting parameters optimization based on simulation and experiment dual drive environment," *Mechanical Systems and Signal Processing*, 2024, doi: 10.1016/j.ymssp.2024.111473.

[88] P. Do, V.-T. Nguyen, A. Voisin, B. Iung, and W. F. Neto, "Multi-agent deep reinforcement learning-based maintenance optimization for multi-dependent component systems," *Expert Systems With Applications*, 2023, doi: 10.1016/j.eswa.2024.123144.

[89] E. Canzini, M. Auledas-Noguera, S. Pope, and A. Tiwari, "Decision making for multi-robot fixture planning using multi-agent reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, 2023, doi: 10.1109/tase.2024.3424677.

[90] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning series. The MIT Press.

[91] L. Buşoniu, R. Babuška, and B. D. Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications - 1*, ser. Studies in Computational Intelligence, D. Srinivasan and L. C. Jain, Eds. Berlin, Heidelberg: Springer, 2010, vol. 310, pp. 183–221, doi: 10.1007/978-3-642-14435-6_7.

[92] F. Keshavarz-Ghorbani and S. H. R. Pasandideh, "A lagrangian relaxation algorithm for optimizing a bi-objective agro-supply chain model considering co2 emissions," *Annals of Operations Research*, pp. 1–31, 2021, doi: 10.1007/S10479-021-03936-1.

[93] M. Mikolajková, H. Saxén, and F. Pettersson, "Mixed integer linear programming optimization of gas supply to a local market," *Industrial & Engineering Chemistry Research*, vol. 57, no. 17, pp. 5951–5965, 2018, doi: 10.1021/ACS.IECR.7B04197.

[94] R. J. Kuo and Y. S. Han, "A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem – a case study on supply chain model," *Applied Mathematical Modelling*, vol. 35, no. 8, pp. 3905–3917, 2011, doi: 10.1016/J.APM.2011.02.008.

[95] R. G. García-Cáceres, M. E. Martínez-Avella, and F. Palacios-Gómez, "Tactical optimization of the oil palm agribusiness supply chain," *Applied Mathematical Modelling*, vol. 39, no. 20, pp. 6375–6395, 2015, doi: 10.1016/J.APM.2015.01.031.

[96] M. C. May, J. Neidhöfer, T. Körner, L. Schäfer, and G. Lanza, "Applying natural language processing in manufacturing," *Procedia CIRP*, vol. 115, pp. 184–189, 2022, 10th CIRP Global Web Conference – Material Aspects of Manufacturing Processes. doi: https://doi.org/10.1016/j.procir.2022.10.071.

[97] *Extraction of Manufacturing Rules From Unstructured Text Using a Semantic Framework*, ser. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 1B: 35th Computers and Information in Engineering Conference, 08 2015, doi: 10.1115/DETC2015-47556.

[98] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Prentice Hall.

[99] M. Vans and S. Simske, *Functional Applications of Text Analytics Systems*, 1st ed. River Publishers, doi: 10.1201/9781003338222.

[100] M. J. Denny and A. Spirling, "Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it," *Political Analysis*, vol. 26, no. 2, pp. 168–189, 2018.

[101] C. P. Chai, "Comparison of text preprocessing methods," *Natural Language Engineering*, vol. 29, no. 3, pp. 509–553, doi: 10.1017/S1351324922000213.

[102] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. Springer, 2006.

[103] T. M. Mitchell, *Machine Learning*, ser. McGraw-Hill series in computer science. McGraw-Hill.

[104] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the *EM* algorithm," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 39, no. 1, pp. 1–22, doi: 10.1111/j.2517-6161.1977.tb01600.x.

[105] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions, 2E*, ser. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., doi: 10.1002/9780470191613.

[106] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, OCLC: ocn190786122.

[107] Q. Qi, D. J. Hessen, T. Deoskar, and P. G. M. Van Der Heijden, "A comparison of latent semantic analysis and correspondence analysis of document-term matrices," *Natural Language Engineering*, pp. 1–31, doi: 10.1017/S1351324923000244.

[108] B. E. Weller, N. K. Bowen, and S. J. Faubert, "Latent class analysis: A guide to best practice," *Journal of Black Psychology*, vol. 46, no. 4, pp. 287–311, doi: 10.1177/0095798420930932.

[109] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992, doi: 10.1007/BF00992698.

[110] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed., ser. Wiley Series in Probability and Statistics. Wiley, doi: 10.1002/9780470316887.

[111] D. P. Bertsekas, *Dynamic programming and optimal control. volume 1*, fourth edition ed. Athena Scientific.

[112] J. Wu and Y. Liu, "A modified multi-agent proximal policy optimization algorithm for multi-objective dynamic partial-re-entrant hybrid flow shop scheduling problem," *Engineering Applications of Artificial Intelligence*, vol. 140, p. 109688, doi: https://doi.org/10.1016/j.engappai.2024.109688.

[113] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, doi: 10.1613/jair.301.

[114] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[115] E. Grigoroudis and Y. A. Phillis, "Modeling healthcare system-of-systems: A mathematical programming approach," *IEEE Systems Journal*, vol. 7, no. 4, pp. 571–580, 2013, doi: 10.1109/JSYST.2013.2251984.

[116] Y. Hata, S. Kobashi, and H. Nakajima, "Human health care system of systems," *IEEE Systems Journal*, vol. 3, no. 2, pp. 231–238, 2009, doi: 10.1109/JSYST.2009.2017389.

[117] S. A. Alowais, S. S. Alghamdi, N. Alsuhebany, T. Alqahtani, A. Alshaya, S. N. Almohareb, A. Aldairem, M. Alrashed, K. B. Saleh, H. A. Badreldin, M. S. A. Yami, S. A. Harbi, and A. M. Albekairy, "Revolutionizing healthcare: the role of artificial intelligence in clinical practice," 2023, doi: 10.1186/s12909-023-04698-z.

[118] C. Li, J. Wang, S. Wangâ€Ž, and Y. Zhang, "A review of iot applications in healthcare," *Neurocomputing*, 2024, doi: 10.1016/j.neucom.2023.127017.

[119] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications - 1*, D. Srinivasan and L. C. Jain, Eds. Springer Berlin Heidelberg, vol. 310, pp. 183–221, series Title: Studies in Computational Intelligence. doi: 10.1007/978-3-642-14435-6_7.

[120] Y. Zhao, Z. Yang, Z. Wang, and J. Lee, "Local optimization achieves global optimality in multi-agent reinforcement learning," vol. 202. ML Research Press, 2023, pp. 42 200–42 226, publisher Copyright: © 2023 Proceedings of Machine Learning Research. All rights reserved.; 40th International Conference on Machine Learning, ICML 2023 ; Conference date: 23-07-2023 Through 29-07-2023.

[121] G. Jin, Y. Liang, Y. Fang, Z. Shao, J. Huang, J. Zhang, and Y. Zheng, "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 10, pp. 5388–5408, 2024, doi: 10.1109/TKDE.2023.3333824.

[122] T. G. Wakjira and M. S. Alam, "Peak and ultimate stress-strain model of confined ultra-high-performance concrete (UHPC) using hybrid machine learning model with conditional tabular generative adversarial network," *Applied Soft Computing*, vol. 154, p. 111353, 03 2024, doi: 10.1016/j.asoc.2024.111353.

[123] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, fifth edition ed., ser. Wiley Series in Probability and Statistics. Wiley, 2016.

[124] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Melbourne, Australia: OTexts, 2018.

[125] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994.

[126] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*, second edition ed., ser. Wiley Series in Probability and Statistics. Wiley, 2015.

[127] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 01 2018, doi: 10.1080/00031305.2017.1380080.

[128] B. V. Vishwas and A. B. Patel, *Hands-On Time Series Analysis with Python: From Basics to Bleeding Edge Techniques*. Apress, 2020.

[129] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, publisher: Institute of Mathematical Statistics. doi: 10.1214/aos/1013203451.

[130] L. Floridi and M. Taddeo, "What is data ethics?" *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2083, p. 20160360, 2016, doi: 10.1098/rsta.2016.0360.

[131] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019, doi: 10.1126/science.aax2342.

[132] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, article 115 (35 pages), Jul. 2021, doi: 10.1145/3457607.

[133] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and machine learning: limitations and opportunities*. The MIT Press, 2023.

[134] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, Aug. 2014, doi: 10.1561/0400000042.

[135] M. Veale and R. Binns, "Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data," *Big Data & Society*, vol. 4, no. 2, p. 205395171774353, 2017, doi: 10.1177/2053951717743530.

[136] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019, doi: 10.1038/s42256-019-0048-x.

[137] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, p. 31–57, Jun. 2018, doi: 10.1145/3236386.3241340.

[138] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 399–410, doi: 10.1109/DSAA.2016.49.

[139] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012, url: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[140] K. Deb, *Multi-objective optimization using evolutionary algorithms*, 1st ed., ser. Wiley-Interscience series in systems and optimization. John Wiley & Sons.

[141] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1126–1135.

[142] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1135–1143.