THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Advancing Edge Intelligence: Federated and Reinforcement Learning for Smarter Embedded Systems

Hongyi Zhang

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

Advancing Edge Intelligence: Federated and Reinforcement Learning for Smarter Embedded Systems

Hongyi Zhang

ISBN 978-91-8103-143-0
Series number 5601
Doktorsavhandlingar vid Chalmers tekniska högskola.
ISSN 0346-718X

Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Printed by Chalmers Digitaltryck
Gothenburg, Sweden, December 2024

To my family

Abstract

Context: The rapid growth of embedded devices and edge computing has brought new opportunities for creating intelligent systems. However, these systems face challenges such as limited computational power and the need to protect user privacy. As a result, there is a need for machine learning methods that can scale effectively, maintain privacy, and adapt to changing conditions in embedded applications.
Objective: This thesis focuses on improving the performance of machine learning models in embedded systems by using federated learning and reinforcement learning. The main goal is to develop methods that allow edge devices to work together without sharing raw data, which helps maintain privacy. Another goal is to make these systems more adaptable to dynamic environments, so they can perform better under changing conditions. Additionally, the research seeks to improve the efficiency of communication and computation across devices.
Method: The research uses a mix of case studies, simulations and real-world experiments. Federated learning is applied to allow edge devices to train models without centralizing the data, keeping sensitive information local. Reinforcement learning is used to help devices learn how to make better decisions by interacting with their environment. These two methods is tested in different scenarios to evaluate improvements in model accuracy, resource use, and adaptability.
Results: The results of this thesis highlight significant advancements in federated learning (FL) and reinforcement learning (RL) for embedded systems. A comprehensive literature review identified six key challenges and open research questions in FL, emphasizing the need for efficient communication, scalability, and privacy preservation. Case studies in telecommunications and automotive applications demonstrated that FL, particularly with asynchronous aggregation protocols, improves model performance, reduces communication overhead, and speeds up training in real-time, dynamic environments. Novel algorithms, such as AF-DNDF and deep RL approaches, further enhanced decision-making capabilities and adaptability in applications like autonomous driving and UAV base station deployment for disaster scenarios. The development of frameworks like EdgeFL provided practical solutions to overcome FL's implementation challenges, offering scalable, low-effort alternatives. Overall, the integration of FL and RL into embedded systems resulted in im-

proved model accuracy, resource utilization, and adaptability, making these approaches highly suitable for real-world industrial use cases.
Conclusion: This research advances the field of edge intelligence by providing a practical approach to deploying machine learning models that are scalable, privacy-focused, and adaptive in embedded systems. The work demonstrates clear improvements in performance and offers a foundation for future research, which could explore more complex learning approaches and apply these techniques to a wider range of embedded systems.

Keywords: Federated Learning, Reinforcement Learning, Machine Learning, Software Engineering

List of Publications

This thesis is based on the following publications:

[A] **Zhang H.**, Bosch J. and Olsson H.H., “Engineering Federated Learning Systems: A Literature Review”. In *2020 International Conference on Software Business* (pp. 210-218). Springer, Cham, 2020.

[B] **Zhang H.**, Dakkak A., Mattos D.I., Bosch J. and Olsson H.H., “Towards Federated Learning: A Case Study in the Telecommunication Domain”. In *International Conference on Software Business* (pp. 238-253). Springer, Cham, 2021.

[C] **Zhang H.**, Bosch J. and Olsson H.H., “Federated learning systems: Architecture alternatives”. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 385-394). IEEE.

[D] **Zhang H.**, Bosch J. and Olsson H.H., “Real-time end-to-end federated learning: An automotive case study”. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 459-468). IEEE.

[E] **Zhang H.**, Bosch J., Olsson H.H. and Koppisetty, A.C., “AF-DNDF: Asynchronous Federated Learning of Deep Neural Decision Forests”. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 308-315). IEEE.

[F] **Zhang H.**, Li J., Qi Z., Lin X., Aronsson A., Bosch J., and Olsson H. H., “Autonomous navigation and configuration of integrated access backhauling for UAV base station using reinforcement learning”. In *2022 IEEE Future Networks World Forum (FNWF)* (pp. 184-189). IEEE..

[G] **Zhang H.**, Li J., Qi Z., Lin X., Aronsson A., Bosch J., and Olsson H. H., “Deep Reinforcement Learning in a Dynamic Environment: A Case Study in the Telecommunication Industry”. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 68-75). IEEE..

[H] **Zhang H.**, Qi Z., Li J., Aronsson A., Bosch J., and Olsson H. H., “5G network on wings: A deep reinforcement learning approach to the UAV-based integrated access and backhaul". In *IEEE Transactions on Machine Learning in Communications and Networking)*, vol. 2, pp. 1109-1126. IEEE..

[I] **Zhang H.**, Bosch J., and Olsson H. H., "Enabling Efficient and Low-Effort Decentralized Federated Learning with the Edgefl Framework". In *Information and software technology*, p.107600..

Acknowledgments

First and foremost, I would like to thank my main supervisor, Jan Bosch, for his patience, enthusiasm, and unwavering support during my Ph.D. studies. His assistance was invaluable throughout the research process, particularly his insightful feedback and recommendations, which encouraged me to reinforce my thoughts and elevate my work to a higher level. I could not have asked for a better advisor and mentor during my Ph.D. studies.

Second, I would like to express my gratitude to my co-supervisor, Helena Holmström Olsson. I want to express my gratitude to her for her important assistance in developing the study technique and making suggestions for strengthening my research. Her suggestions on how to collaborate with companies, analyze data, and be available on a constant basis have helped to strengthen the foundation for my empirical study.

I would like to thank everyone at Chalmers University who helped with this research, especially my former examiner Ivica Crnkovic. His patience, advice, and support will live on in my recollection forever.

I would also like to thank my new examiner, Robert Feldt, for his guidance and suggestions on improving my research and this thesis.

In addition, I would like to thank my wife, Lida, my parents, and my kittens, Guapi and Leo, for their unending support and encouragement during my studies.

Last but not least, I would like to thank the Software Center and company participants, colleagues for giving me the opportunity to work more directly in the industries during this project. Without their help, this research would not be feasible.

Contents

CHAPTER 1

Introduction

The rapid increase in data generated by humans and machines has surpassed our capacity to process and extract meaningful insights. Artificial intelligence (AI) plays a pivotal role in takling this challenge by enabling machines to learn from data and make complex decisions. AI now drives critical advances in data-driven decision-making and underpins all forms of computer learning [1]. For example, in a simple game like tic-tac-toe, with its 255,168 possible moves, humans can still intuitively figure out how to avoid losing [2]. However, in more complex games like checkers, which has nearly 500 trillion possible positions, only a few experts can excel. With vast amounts of data, computers, through AI, can efficiently compute these possibilities and make the best decisions. This showcases how AI can process complex combinations and optimize outcomes, making it a powerful tool for enhancing various aspects of human life and business processes [3][4].

In industrial applications, AI is increasingly being integrated to enhance the intelligence of products and improve user experiences [5]. Rather than existing as standalone applications, AI technologies are often embedded within products and services, similar to how virtual assistants like Siri enhance smartphones [6]. From security systems to financial analytics, AI-driven solutions leverage massive datasets to make smarter predictions and recommendations, continuously learning from new information to refine their accuracy. The adaptability of AI is particularly noteworthy; machine learning algorithms allow systems to improve over time based on user data [7]. For example, just as an AI algorithm can learn to play chess, it can also learn to recommend personalized content to users based on their preferences. The continuous feedback loop provided by machine learning ensures that AI systems evolve to offer more relevant and optimized solutions.

However, despite the significant progress in AI, the integration of these technologies into embedded systems and edge computing presents several challenges [8][9]. One of the primary issues is the limited computational resources and energy efficiency of embedded devices. Edge devices, which often operate in decentralized environments, must process data locally, requiring algorithms that are both lightweight and efficient [10][11][12]. These devices also face strict constraints regarding memory, processing power, and real-time response, all while handling increasingly complex tasks such as AI inference and model updates. Additionally, security and privacy are major concerns, especially when sensitive user data is processed at the edge. Federated Learning , for example, addresses privacy by enabling decentralized model training without sharing raw data, but it introduces challenges related to communication overhead, model accuracy, and heterogeneity across devices [13][14][14].

This thesis contributes to overcoming some of these challenges by exploring the integration of Federated Learning and Reinforcement Learning into edge and embedded systems. Through a comprehensive analysis of existing literature and empirical case studies in collaboration with industry, the research highlights several key contributions. The thesis presents novel FL and RL algorithms that address issues such as communication latency and model aggregation in decentralized systems. These algorithms, validated in both telecommunications and automotive applications, demonstrate the potential to significantly reduce communication overhead while maintaining high model accuracy in dynamic, resource-constrained environments. Additionally, Reinforcement Learning is applied to enhance the decision-making capabilities of embedded systems, allowing them to better adapt to real-time changes in their environments, such as autonomous navigation and resource optimization.

The development of asynchronous Federated Learning protocols and frameworks like EdgeFL enables more scalable and efficient deployment of AI on edge devices, further enhancing their capacity to handle complex tasks in heterogeneous hardware settings. By providing a practical framework for implementing AI in embedded systems, this thesis addresses both the theoretical and applied aspects of integrating Federated Learning and Reinforcement Learning into real-world industrial contexts. The results demonstrate substantial improvements in model accuracy, resource utilization, and system adaptability, making these approaches well-suited for future AI-driven embedded and edge applications.

1.1 Structure of the thesis

The thesis is structured as follows:

Chapter 2 provides the background necessary to understand the context of this research. It begins with an exploration of AI and its evolution, discussing key advancements in machine learning techniques and their application in various domains. The chapter then dives into edge computing and embedded systems, explaining their role in processing data close to its source, thus reducing latency and improving responsiveness. It outlines the current challenges faced by embedded systems, including limited computational resources, energy constraints, and the need for real-time processing capabilities. Privacy concerns are also addressed, particularly in scenarios where sensitive data is generated and processed locally. The chapter further discusses the significance of integrating Federated Learning and Reinforcement Learning into these systems, providing an overview of how these techniques can enhance adaptability and decision-making in dynamic environments. This foundation prepares the reader for the detailed methodologies and contributions presented in the subsequent chapters.

Chapter 3 details the research methods employed throughout this thesis. It outlines the research goals and motivations behind each approach used to address the identified challenges. These methods include both theoretical explorations and practical applications of Federated Learning and Reinforcement Learning in edge computing environments.

Chapter 4 presents the core objectives of the thesis, showing how each of the primary research questions (RQ1, RQ2, and RQ3) is addressed. It provides an overview of the research contributions, specifically focusing on how the methods proposed contribute to improving the scalability, performance, and adaptability of Federated Learning and Reinforcement Learning in embedded systems.

The main body of the thesis, comprising Chapters 5 through 13, is based on the included papers A through I.

- Chapter 5 provides a comprehensive overview of the state-of-the-art in Federated Learning systems, based on the literature review presented in Paper A. It categorizes various FL deployments, identifying their challenges and presenting open research questions. The chapter highlights six key results regarding the performance, privacy, and communication efficiency of existing FL systems.
- Chapter 6 dives into the real-world obstacles and limitations that prevent industries from fully adopting Federated Learning, drawing on insights from Paper B. This chapter discusses the challenges faced by companies in implementing FL, such as integration complexity, limited computational resources, and privacy concerns. It summarizes the case studies that identify key issues and provides criteria for designing reliable FL systems.
- Chapter 7 addresses the various architectural alternatives for Federated Learning, as explored in Paper C. This chapter compares four architecture designs—centralized, hierarchical, regional, and decentralized—using empirical performance data. It discusses the trade-offs between communication latency, model accuracy, and scalability, offering guidance on selecting the appropriate architecture based on system requirements and use cases.
- Chapter 8 introduces a real-time asynchronous Federated Learning method, as described in Paper D. This chapter focuses on the challenges of synchronous model aggregation in FL and proposes an alternative approach suitable for dynamic and heterogeneous hardware environments. It includes an automotive case study that demonstrates the effectiveness of asynchronous FL in improving edge model performance while reducing communication overhead.
- Chapter 9 builds on the work in Paper E, combining deep neural decision forests (DNDF) with asynchronous Federated Learning to create a novel algorithm called AF-DNDF. This chapter presents experimental

results from the automotive industry, showing how AF-DNDF enhances local edge model quality without sacrificing accuracy, making it highly efficient for edge computing tasks such as road object recognition.

- Chapter 10 extends the discussion of Reinforcement Learning in edge environments, as presented in Paper F. This chapter focuses on the use of deep Reinforcement Learning to autonomously navigate and optimize the configuration of UAV-based base stations. The proposed algorithm demonstrates how Reinforcement Learning can be applied in dynamic environments to improve mission-critical communication services.

- Chapter 11 covers the integration of AI in UAV-based networks, drawing on Paper G. It explores the use of Reinforcement Learning to optimize the 3D placement of UAV base stations, enabling them to provide reliable connectivity in disaster scenarios. This chapter highlights the potential of using AI to dynamically adapt to changing conditions in real-time and maintain consistent service quality.

- Chapter 12 presents the EdgeFL framework, as introduced in Paper H, which offers a low-effort and efficient solution for decentralized Federated Learning. This chapter discusses the challenges of scalability and centralization in FL and demonstrates how EdgeFL overcomes these issues through decentralized aggregation protocols. The chapter also covers the framework's impact on improving learning efficiency and model evolution in edge devices.

- Chapter 13 concludes the main body of the thesis with a detailed discussion of the results and contributions from the included papers. This chapter reflects on how the novel methods presented throughout the thesis address key challenges in edge intelligence, improving the deployment and performance of AI in embedded systems.

Chapter 13 summarizes the main findings from the research, reflecting on the novel methods proposed and their impact on the deployment of AI in embedded systems. The chapter also discusses potential direction for future research, including further exploration of decentralized learning architectures, optimizing Reinforcement Learning for dynamic environments, and scaling the proposed methods to larger industrial applications.

CHAPTER 2

Background

In recent years, artificial intelligence (AI) has transformed various sectors, driving innovation and efficiency. This chapter explores the evolution of AI, the role of machine learning, the significance of edge computing, and the challenges faced by embedded systems. By understanding these foundational concepts, we can better appreciate the contributions of Federated Learning (FL) and Reinforcement Learning (RL) in enhancing the capabilities of intelligent systems. Section 2.1 presents the evolution of artificial intelligence, tracing its development from early symbolic reasoning to modern machine learning techniques. It highlights how the availability of large datasets and powerful computational resources have driven advancements in AI, enabling applications across various domains. Section 2.2 discusses the role of machine learning as a core component of AI, emphasizing its ability to identify patterns and make predictions based on data. It outlines how machine learning enhances existing products and services, transforming them into intelligent systems that improve user experiences. Section 2.3 explores edge computing and embedded systems, explaining their significance in processing data closer to its source to reduce latency and bandwidth usage. It describes how embedded systems, with their specialized functions and resource constraints, play a crucial role in the implementation of edge computing solutions. Section 2.4 highlights the current challenges faced by embedded systems, including limited computational resources, energy efficiency concerns, and the need for real-time data processing. It emphasizes the importance of finding effective solutions to address these challenges, particularly in dynamic environments. Section 2.5 discusses the significance of Federated Learning and Reinforcement Learning as innovative approaches to overcoming the challenges in embedded systems. It explains how Federated Learning enables decentralized training while preserving privacy and how Reinforcement Learning enhances adaptability, contributing to the development of smarter and more resilient systems.

2.1 Evolution of Artificial Intelligence

Artificial intelligence (AI) fundamentally seeks to replicate and simulate human-like cognitive functions in machines. This journey began in the mid-20th century, marked by pioneering research in symbolic reasoning and rule-based systems. Early AI efforts focused on creating algorithms that could mimic logical reasoning and solve problems using predefined rules [15][16]. However, these approaches were often limited by their reliance on explicit programming and a lack of adaptability to new, unforeseen situations.

As technology advanced, the emergence of machine learning, a vital subset of AI, significantly transformed the landscape of intelligent systems. Unlike traditional AI methods, machine learning enables systems to learn from data and improve their performance autonomously without the need for detailed instructions for each specific task [17][18]. This shift has been driven by the exponential growth of available data and the development of powerful computational resources, including GPUs and cloud computing platforms. Consequently, machine learning algorithms can now process vast amounts of information, allowing them to identify patterns and make predictions with increasing accuracy [19].

Deep learning, a specialized branch of machine learning, has gained particular prominence in recent years due to its remarkable success in complex tasks such as image recognition, natural language processing, and game playing. Deep learning leverages artificial neural networks with many layers, enabling models to learn hierarchical representations of data [20][21]. This has led to breakthroughs in various fields, including healthcare—where AI aids in disease diagnosis and treatment planning—finance, where algorithms enhance trading strategies, and transportation, which benefits from advancements in autonomous vehicles [22].

Deep learning, a specialized branch of machine learning, has gained significant prominence in recent years due to its remarkable success in solving complex tasks such as image recognition, natural language processing, and game playing. Deep learning models leverage artificial neural networks with many layers, allowing them to learn hierarchical representations of data [20][21]. This has led to groundbreaking advancements across various domains.

In healthcare, deep learning is revolutionizing disease diagnosis and treatment planning. For example, convolutional neural networks (CNNs) are employed in medical imaging for detecting conditions such as cancer, diabetic retinopathy, and cardiovascular diseases with high accuracy [23]. In genomics, deep learning models help identify disease-related genetic variants, improving personalized medicine approaches.

In finance, deep learning is enhancing algorithmic trading by identifying patterns in vast amounts of market data, helping predict price movements and optimize investment strategies. Fraud detection systems also utilize deep learning to analyze transactions in real-time, identifying suspicious activities and minimizing risks for financial institutions [24].

In transportation, deep learning plays a critical role in autonomous vehicle development. Self-driving cars rely on deep learning models for object detection, path planning, and decision-making, enabling safe navigation in complex environments. Companies like Tesla, Waymo, and Uber have integrated deep learning technologies to advance the capabilities of autonomous driving systems, pushing the boundaries of innovation in this sector [25].

In manufacturing, deep learning is enhancing quality control and predictive maintenance. By analyzing sensor data, deep learning models can predict when equipment is likely to fail, allowing companies to perform maintenance before breakdowns occur, reducing downtime and costs. In addition, visual inspection systems powered by deep learning are being used to detect defects in products during the manufacturing process with higher precision than traditional methods [7][26].

These examples illustrate how deep learning has become a transformative force in numerous industries, driving innovation and improving efficiency in a wide range of applications. The advancements in AI have paved the way for numerous applications that enhance efficiency and decision-making across diverse industries. As AI continues to evolve, it holds the promise of further transforming how humans interact with technology, improving quality of life, and driving innovation in various sectors.

2.2 The Role of Machine Learning

Machine learning serves as the backbone of modern artificial intelligence applications, providing the tools and methodologies necessary for machines to learn from data and improve over time. It encompasses a diverse array of algorithms and models that enable systems to identify patterns, make predictions, and execute complex tasks across various domains [27]. The rise of big data has been pivotal in this evolution, as the vast quantities of information generated in our digital age can be harnessed to train more sophisticated and accurate models.

In industrial applications, machine learning significantly enhances the intelligence of products and services. Rather than functioning as isolated solutions, AI capabilities are increasingly being integrated into existing systems, enriching user experiences and operational efficiencies. For instance, virtual assistants like Siri and Alexa utilize machine learning algorithms to provide personalized responses and recommendations based on user interactions, preferences, and context [6]. These systems continuously learn and adapt to improve their performance, demonstrating the dynamic capabilities of machine learning.

Moreover, machine learning is transforming sectors such as manufacturing, healthcare, and finance. In manufacturing, predictive maintenance powered by machine learning algorithms allows companies to foresee equipment failures, minimizing downtime and reducing costs [7][28]. In healthcare, machine learning models analyze patient data to assist in diagnosing diseases, personalizing treatment plans, and predicting patient outcomes [29]. In finance, algorithms evaluate vast datasets to detect fraudulent activities and optimize trading strategies [30].

The versatility and adaptability of machine learning make it a critical component in the development of intelligent systems. As organizations increasingly adopt these technologies, the potential for innovation and improved decision-making grows, leading to enhanced efficiency and competitiveness in the market.

2.3 Edge Computing and Embedded Systems

Edge computing represents a paradigm shift in how data is processed, emphasizing the importance of conducting data analysis closer to its source instead of relying on centralized data centers [31]. This approach significantly reduces latency and bandwidth consumption, making it especially suitable for applications that require real-time processing and immediate responsiveness. By processing data at the edge of the network, organizations can achieve faster decision-making, enhance user experiences, and improve overall system efficiency [32].

Embedded systems are specialized computing devices designed to perform specific functions within larger systems. They play a crucial role in the context of edge computing, as these systems are often deployed in various environments, such as industrial settings, smart homes, and healthcare facilities. Due to their specialized nature, embedded systems typically have limited computational resources, which necessitates efficient design and implementation strategies [33].

The integration of artificial intelligence within embedded systems has the potential to enhance performance and enable smarter decision-making [34]. For example, in industrial automation, embedded AI can optimize machine operations by analyzing real-time data from sensors and adjusting processes accordingly. In smart home applications, AI-powered devices can learn user preferences and adapt their functions, providing personalized experiences [35].

However, deploying AI in resource-constrained environments presents unique challenges [14]. One major concern is computational power; traditional machine learning models may require substantial processing capabilities that embedded systems often lack. Energy efficiency is another critical issue, as many embedded devices operate on limited power sources, necessitating algorithms that can deliver high performance without excessive energy consumption. Additionally, the ability to process data in real-time is essential, particularly in applications where immediate feedback is crucial, such as autonomous vehicles or healthcare monitoring systems [36].

Addressing these challenges is vital for the successful integration of AI in edge computing and embedded systems. Ongoing research and development efforts are focused on creating lightweight algorithms, optimizing data processing techniques, and enhancing hardware capabilities to ensure that embedded systems can leverage the power of AI effectively.

2.4 Current Challenges in Embedded Edge Systems

Despite the significant advancements in artificial intelligence (AI) and edge computing, several key challenges persist in the effective deployment of embedded systems, particularly when these systems are expected to handle real-time, dynamic environments. Embedded systems, which are typically resource-constrained and operate in environments with stringent performance demands, are integral to sectors like automotive, healthcare, telecommunications, and smart manufacturing [37][38]. However, the nature of these environments brings several challenges that must be addressed for AI at the edge to realize its full potential.

Latency remains one of the most critical issues for embedded systems, especially for applications that demand instantaneous decision-making. Systems like autonomous vehicles, drone navigation, and real-time industrial monitoring require near-instant responses to dynamic, evolving conditions [39][40]. Any delay in processing sensor data or making decisions can result in safety risks or suboptimal system performance. The need for low-latency processing is complicated by the fact that many machine learning models—especially deep learning models—are computationally expensive. Balancing the trade-off between fast decision-making and processing complex models locally presents a challenge that requires developing optimized algorithms and hardware acceleration techniques such as edge TPUs and GPUs [31][41]. These technologies help in mitigating latency but also require intelligent software design to maximize efficiency.

Another pressing challenge is communication costs, which significantly affect embedded edge systems [42]. These systems often rely on transmitting data between remote devices and centralized data centers, introducing bandwidth constraints, increased energy consumption, and potential network reliability issues. Particularly in settings like remote health monitoring or agricultural IoT networks, where reliable connectivity may be intermittent or expensive, minimizing data transmission becomes crucial [43]. To address this, researchers have been exploring methods such as data compression, feature extraction, and selective communication—where only the most relevant data is transmitted to centralized servers—to alleviate communication burdens while preserving data integrity and model accuracy [44]. Federated Learning, which reduces communication by keeping data processing localized to the device, is another promising avenue that aligns with this objective [45].

The performance of machine learning models is also tightly constrained by the computational resources available on embedded devices. Many of these devices operate with limited processing power, memory, and energy capacity, which makes it difficult to deploy traditional, large-scale AI models [36][35]. As a result, there is a growing need for the development of lightweight models and model compression techniques such as quantization, pruning, and knowledge distillation [46]. These methods reduce the size and complexity of machine learning models, making them suitable for deployment on devices with restricted computational budgets while maintaining acceptable levels of performance. For instance, TinyML has emerged as a field focused on bringing the power of AI to small, resource-constrained devices by optimizing models for minimal power consumption and rapid inference [47].

Another challenge facing embedded systems is maintenance, particularly for systems deployed in remote or difficult-to-access locations such as weather stations, oil rigs, or satellite systems [48]. These devices must operate reliably over extended periods, often with minimal human intervention. Regular software updates, security patches, and performance monitoring are essential to ensuring these systems remain functional and secure [49]. However, performing these tasks remotely can be both costly and challenging. To address this, over-the-air (OTA) updates have become a critical feature, allowing developers to update the software of embedded devices without physical access. Furthermore, self-diagnosis and self-healing capabilities—where the system can autonomously detect issues and take corrective actions—are gaining importance in mission-critical systems [50]. Such solutions are essential for enhancing the reliability and longevity of embedded systems, especially in contexts where system downtime could have serious consequences.

In summary, addressing the challenges of latency, communication costs, performance limitations, and maintenance difficulties is paramount for the successful integration of AI into embedded edge systems. By developing optimized algorithms, reducing data transmission needs, enhancing computational efficiency, and implementing effective remote maintenance strategies, ongoing research is helping to unlock the full potential of embedded systems across diverse application domains. These advancements will enable embedded systems to play a crucial role in future intelligent infrastructures, from smart cities and autonomous transportation to industrial automation and remote healthcare.

2.5 The Significance of Federated Learning and Reinforcement Learning

Federated Learning and Reinforcement Learning offer promising strategies to tackle the unique challenges posed by embedded systems in the context of edge computing. These approaches address key limitations such as data privacy, communication costs, and system adaptability, making them well-suited for real-world applications across various industries [51][52].

Federated Learning allows decentralized model training across multiple edge devices, enabling each device to learn from local data without transferring it to a central server [53]. This is especially crucial in applications where data privacy and security are paramount, such as healthcare, finance, and telecommunications [54]. By keeping raw data on the local devices and only sharing model updates (such as gradients or weight adjustments), FL substantially reduces the risk of data breaches and helps organizations comply with regulatory requirements like GDPR [55][56]. Additionally, this decentralized architecture mitigates the issues associated with centralized data processing, where bottlenecks or single points of failure can arise. By distributing the learning process across edge devices, FL enhances system scalability and fault tolerance [57].

Moreover, communication overhead is a significant concern in edge computing, particularly when devices are deployed in resource-constrained environments with limited bandwidth or intermittent connectivity. FL addresses this by drastically reducing the amount of data that needs to be transmitted between devices and central servers, as only model parameters (which are typically much smaller than raw datasets) are exchanged [57]. This reduction in communication not only lowers network congestion but also helps in extending battery life for devices that operate on limited power, such as IoT sensors or wearables [54]. In scenarios like smart cities or autonomous vehicle fleets, where devices are constantly generating data, the ability to minimize communication while still improving model accuracy is a critical advantage [58].

On the other hand, Reinforcement Learning enhances the adaptability and decision-making capabilities of embedded systems, particularly in environments where conditions are dynamic and unpredictable [59]. Unlike traditional machine learning algorithms that rely on static datasets, RL involves continuous interaction with the environment, allowing the system to learn from real-time feedback. This makes RL particularly valuable in applications like autonomous driving, robotic process automation, or smart industrial control systems, where decisions must be made quickly and adjusted on-the-fly as new information becomes available [60]. For instance, an RL agent in an autonomous vehicle can learn to navigate changing road conditions, avoid obstacles, and adjust to traffic patterns—all while optimizing for fuel efficiency or safety [61].

The synergy between FL and RL presents a hybrid approach that combines the strengths of both methods, making it highly effective for developing smarter and more resilient embedded systems. For example, in collaborative environments, such as a fleet of autonomous drones or industrial IoT networks, FL enables each device to learn from local data without compromising privacy, while RL empowers devices to make decisions autonomously based on the specific conditions they encounter [62]. This hybrid approach leads to improved learning efficiency, reduced data transmission costs, and enhanced responsiveness of the system. In telecommunications or smart energy grids, where devices must operate cooperatively and adapt to changing loads or demands, combining FL's decentralized learning with RL's ability to dynamically adjust to environmental feedback offers significant performance gains. Both methods can be key enablers for advancing the capabilities of embedded systems within edge computing [63][64]. By leveraging FL's privacy-preserving model training and RL's adaptive decision-making, the systems can become not only more efficient and scalable but also more secure and responsive to real-world challenges.

2.6 Summary

In summary, this chapter has outlined the evolution of AI and its foundational role in machine learning, as well as the significance of edge computing and embedded systems. The challenges these systems face highlight the urgent need for innovative solutions. Federated Learning and Reinforcement Learning are key to addressing these challenges, paving the way for the development of advanced embedded systems that can effectively respond to real-world demands. The subsequent chapters of this thesis will further explore these methodologies and their practical applications.

CHAPTER 3

Research Methodology and Design

The methodology chapter outlines the research design and approaches employed to investigate the integration of Federated Learning and Reinforcement Learning within embedded systems and edge computing environments. This chapter aims to provide a comprehensive overview of the methods utilized to address the research questions, emphasizing the engineering and deployment of FL, the exploration of architectural frameworks, and the enhancement of adaptability through RL. By combining theoretical insights with empirical research [65][66], this methodology seeks to bridge the gap between advanced machine learning techniques and practical applications in resource-constrained environments. The subsequent sections will detail the specific methodologies used in each paper, illustrating how they collectively contribute to advancing the field of edge intelligence.

3.1 Research Questions

In this section, we outline the primary research questions guiding this study. These questions are designed to address critical aspects of integrating Federated Learning and Reinforcement Learning within embedded systems and edge computing environments. By focusing on the deployment of FL and RL methods, this research aims to enhance data privacy, optimize system performance, and improve the adaptability of embedded systems in dynamic settings. Each question represents specific challenges and opportunities that arise in the application of these advanced machine learning techniques, providing a comprehensive framework for our investigation:

- RQ1. How can Federated Learning be effectively engineered and deployed in embedded systems to enhance data privacy, reduce communication costs, and ensure model accuracy?
- RQ2. What architectural frameworks for Federated Learning can optimize scalability and performance in edge computing environments, and how do these architectures influence model training efficiency?
- RQ3. How can Federated Learning and Reinforcement Learning methods enhance the adaptability and efficiency of embedded systems in dynamic environments?

The first research question (RQ1) addresses the need for effective strategies to implement ML in embedded systems. It is crucial as it addresses the challenges of implementing FL in embedded systems, which often have limited computational resources and stringent requirements for data handling. The question emphasizes the importance of balancing three key aspects: data privacy, communication costs, and model accuracy. One possible solution is FL. FL allows for decentralized training, which helps maintain data privacy by keeping raw data localized rather than transmitting it to a central server. However, this approach must be carefully designed to minimize the communication costs associated with sharing model updates. Additionally, ensuring model accuracy is vital, as inaccuracies can lead to poor decision-making in embedded applications. By investigating these factors, we aim to develop effective strategies that practitioners can implement when deploying FL in resource-constrained environments.

The second research question (RQ2) focuses on the architectural aspects of FL systems. It seeks to explore the various designs and frameworks that can enhance the scalability and performance of FL in edge computing contexts. Scalability is essential as embedded systems often operate in diverse and dynamic environments, requiring architectures that can adapt to varying loads and conditions. Performance optimization is equally important, as efficient training can significantly impact the responsiveness and effectiveness of embedded applications. This question aims to identify specific architectural features and design choices that can lead to improved training efficiency, allowing FL systems to better serve the demands of edge computing while maintaining performance standards.

The third research question (RQ3) examines the synergy between FL and RL, focusing on their potential to enhance the adaptability and efficiency of embedded systems. In dynamic environments, systems must quickly adjust to changes and uncertainties, making adaptability a critical factor for success. This research explores how FL and RL can synergize to create smarter and more resilient embedded systems. The goal is to leverage the strengths of both methodologies, including FL's ability to preserve privacy and reduce data transfer, alongside RL's capacity for learning optimal actions through interaction with the environment. Understanding this synergy can lead to innovative solutions that effectively address real-world challenges faced by embedded systems across various application domains.

To address these research questions, we employ a multi-faceted approach that integrates both theoretical and practical perspectives [67][65]. For RQ1, we begin with a comprehensive literature review to explore existing frameworks, strategies, and case studies related to the implementation of Federated Learning in embedded systems. This review is complemented by case study interviews with industry stakeholders, including engineers, product managers, and researchers. These interviews provide us with firsthand insights into the practical challenges and successes encountered in real-world deployments of FL, allowing us to capture a diverse range of experiences and best practices. This dual approach ensures that our findings are grounded in established research while being informed by the realities of industry application. For RQ2 and RQ3, we focus on the development and validation of innovative algorithms and architectural frameworks tailored to enhance the scalability and performance of FL and Reinforcement Learning in edge computing environments. This involves designing novel architectures that optimize training efficiency and adapt to the unique constraints of embedded systems. We then apply these innovations in real-world scenarios through partnerships with companies that operate in the embedded systems domain. Collaborating with these organizations allows us to test our proposed solutions in dynamic settings, gaining valuable insights into the practical challenges and requirements of implementing FL and RL.

By working closely with industry partners, we ensure that our research is not only theoretically robust but also highly relevant to contemporary applications. This collaborative approach enables us to refine our algorithms and frameworks based on real-world feedback, ensuring they meet the operational demands and constraints of embedded systems. Ultimately, this comprehensive methodology aims to bridge the gap between theory and practice, enhancing the adaptability and efficiency of embedded systems in various application domains.

3.2 Design Science

This thesis adopts a design science approach, which is well-suited for addressing the research objectives by focusing on the creation and evaluation of artifacts—such as models, frameworks, or methodologies—that solve identified problems within the field of Federated Learning, Reinforcement Learning and edge intelligence [68] [69]. Design science is inherently a problem-solving research paradigm, making it an ideal fit for this study, as the core goal is to generate innovative solutions that enhance the performance, scalability, and adaptability of machine learning models in embedded systems.

The essence of design science lies in its ability to bridge the gap between theoretical exploration and practical application [70] [69]. In this research, the artifacts developed, such as architectural frameworks and machine learning models, are designed to address real-world challenges related to data privacy, computational constraints, and communication efficiency in distributed edge environments. By leveraging design science, this study focuses on constructing and refining these solutions iteratively, ensuring that they are not only theoretically sound but also practically viable.

The potential of design science in this context is particularly relevant due to its iterative nature, where the creation, evaluation, and refinement of solutions are deeply embedded in the research process [71]. This methodology aligns closely with the research objectives of improving Federated Learning systems and enhancing decision-making in dynamic environments through Reinforcement Learning. Specifically, design science allows for the systematic exploration of different architectures, model optimization techniques, and the deployment of these models in resource-constrained environments [69].

Additionally, the evaluation component of design science ensures that the proposed solutions are rigorously tested in both simulated environments and real-world case studies. By incorporating extensive feedback loops—through workshops, collaborations with industry experts, and validation via simulations—this approach guarantees that the resulting artifacts are not only theoretically grounded but also practical and effective in real industrial scenarios [72].

Design science was selected as the research methodology for this thesis because it provides a structured framework to develop, implement, and evaluate innovative artifacts. Its iterative nature, emphasis on solving real-world problems, and focus on the evaluation of practical outcomes align seamlessly with the goals of enhancing Federated Learning and Reinforcement Learning in embedded systems. This research follows the design science framework, which is structured around six key concepts as outlined by Peffers et al. [72]. These concepts form a comprehensive approach to addressing complex problems in a systematic manner:

1. Problem Identification and Motivation: The research begins by identifying specific challenges in embedded system, particularly those related to privacy concerns, communication costs, and computational constraints in resource-limited edge environments. The motivation for this work stems from the need to address real-world issues in industries such as automotive, telecommunications, and healthcare, where secure and efficient edge learning is critical. Engaging with industry stakeholders and conducting a comprehensive literature review allowed us to clearly pinpoint the gaps, such as the scalability of FL architectures and the adaptability of Reinforcement Learning models in embedded systems. This phase ensures the research is tightly focused on solving practical problems with significant impact.

2. Definition of Objectives for a Solution: After identifying the challenges, the next step involved defining clear and measurable objectives for the research. These objectives included reducing communication latency, enhancing model accuracy while maintaining privacy, and developing lightweight models, frameworks, algorithms for resource-constrained devices. These objectives were directly aligned with both the academic

goals of advancing FL and RL methods and the practical needs of industry partners, ensuring that the research outputs would be relevant for real-world applications. This step ensured that each artifact developed was grounded in addressing a specific problem with tangible benefits.

3. Solution Design: The design phase involved developing artifacts such as asynchronous Federated Learning frameworks, architectural alternatives for FL systems, and Reinforcement Learning algorithms optimized for edge computing environments. These solutions were tailored to address the identified challenges, with a focus on reducing resource consumption and improving scalability and privacy. The design process was iterative, incorporating feedback from early prototypes and simulations to refine the models. For instance, a key design contribution was the development of an asynchronous Federated Learning method that improved model accuracy while minimizing communication overhead in heterogeneous hardware environments, which was demonstrated through an automotive case study.

4. Demonstration: Once the solutions were designed, they were applied in practical scenarios, including real-world case studies and simulations. For instance, the asynchronous FL method was tested in a real-time automotive scenario, where edge devices needed to quickly process sensor data while communicating with a central server. The demonstration phase showed that the designed solutions not only worked in theory but also offered tangible improvements in performance. Demonstrating the effectiveness of the algorithms in dynamic environments, such as UAV-based networks and edge computing systems, reinforced the validity and applicability of the proposed methods.

5. Evaluation: Evaluation was carried out using a combination of empirical testing, quantitative metrics, and industry feedback. Key performance indicators such as latency reduction, model accuracy, and resource efficiency were measured to assess how well the artifacts achieved the objectives. In one evaluation, the AF-DNDF algorithm showed significant improvement in both local edge model quality and resource efficiency in real-world scenarios like road object recognition. The evaluation phase was critical in identifying areas for further refinement and in validating the practical utility of the research outcomes. This ensured that the pro-

posed solutions were not only innovative but also robust and effective in meeting industry needs.

6. Communication: The final phase involved the dissemination of findings through academic publications, workshops, and industry collaborations. By presenting the results to both academia and industry, the research contributes to ongoing advancements in the field, while also informing practitioners of new methodologies that could be applied to solve real-world problems. This communication ensures that the research outcomes are accessible and can be further refined or built upon by other researchers and practitioners.

As noted by Wohlin et al. [73], the essence of design science lies in problem-solving through the development of artefacts aimed at improving specific situations. These key concepts can be distilled into three overarching activities:

1. Problem Identification or Conceptualization: This involves recognizing the issues that require solutions, synthesizing information from various sources to articulate the core challenges.
2. Solution or Artefact Design and Implementation: This activity encompasses the creative and technical processes involved in developing and implementing artefacts that address the identified problems.
3. Evaluation or Validation: This final activity focuses on assessing the effectiveness of the solutions through rigorous testing and analysis, ensuring that they meet the objectives defined in earlier phases.

By applying this structured framework, this research aims to create practical solutions that not only advance theoretical knowledge but also provide tangible benefits to practitioners in the field of embedded systems and Federated and Reinforcement Learning.

3.2.1 Data Collection and Evaluation Methods

The research applied a range of data collection and evaluation methods, including literature reviews, semi-structured interviews, empirical case studies, and simulation experiments, each designed to investigate different aspects of FL and RL in embedded systems [74]. The literature review provided a theoretical foundation, helping to frame the research questions and identify key challenges in FL and RL workflows. Semi-structured interviews with industry experts further contextualized these challenges, while empirical case studies documented real-world applications. Finally, simulation experiments evaluated the effectiveness of developed algorithms and frameworks, providing data on performance in controlled settings.

3.2.2 Data Analysis of Case Studies and Simulations

Data analysis varied by study type, with specific methods applied to different data sources. For empirical case studies, qualitative analysis of interview transcripts and observational data were conducted to capture industry challenges and current practices. Thematic analysis was employed to distill insights from interviews, while case study data was summarized to highlight obstacles in FL and RL adoption and the corresponding needs in embedded systems [75].

For simulation experiments, quantitative analysis assessed the performance and adaptability of developed FL and RL algorithms under various scenarios. By applying statistical methods to measure improvements and validate algorithmic efficiency, the simulations provided quantitative backing for proposed solutions, allowing us to draw comparisons between baseline and advanced configurations [73].

3.2.3 Process and Contributions by Individual Papers

Each individual paper contributes to one or more of the aforementioned activities. Through literature reviews and case studies, combined with semi-structured interviews, Papers A and B identify the challenges of Federated Learning, establish the motivation for this research, and define the objectives. Papers C, I propose architectures and frameworks that serve as solutions to the challenges raised in earlier phases, which are then validated using empirical datasets through case studies and simulation experiments. Papers D, E, F, G, and H focus on advancing the application of Federated Learning and Reinforcement Learning in embedded systems, providing further insights and methodologies that enhance adaptability and performance in dynamic environments. Additionally, we maintained close collaboration with companies within the Software Center, organizing seminars and meetings to deepen our understanding of the challenges faced by practitioners, exchange progress updates, and gather insights from industry experts. Following the presentation of our results, we engaged in discussions about the implications of the new algorithms and frameworks, further refining and enhancing our research.

In summary, this study used a multi-method approach to address FL and RL challenges in embedded systems, combining theoretical and practical insights from case studies, interviews, and simulation data. Design science methodology was instrumental in structuring the research, as it integrates problem identification, solution development, and iterative validation. This approach aligns with the research's goal of developing artefacts that solve practical problems in real-world contexts, making it ideal for advancing FL and RL applications within the embedded systems domain.

3.3 Research Methods

The research techniques employed in this study, including both data collection and evaluation methods, are essential for gathering empirical data necessary to analyze actions within real-world contexts [74]. By utilizing methods such as literature reviews and case studies, this thesis effectively addresses practical problems and offers a comprehensive understanding of the behaviors and influences associated with various choices made by researchers and practitioners. Literature reviews provide a solid theoretical foundation, allowing us to identify existing gaps in knowledge and articulate the significance of our research questions. Case studies, on the other hand, facilitate an in-depth exploration of specific instances within the embedded systems domain, revealing the complexities and nuances of implementing Federated Learning and Reinforcement Learning. Together, these techniques enable the collection of rich qualitative and quantitative data, fostering insights that inform the development of innovative solutions and enhancing the overall robustness of the research findings.

3.3.1 Literature Review

A literature review, according to [76][77], provides knowledge that includes substantive findings as well as theoretical and methodological contributions to a specific subject. These works are described in relation to the subject matter under consideration by discovering and utilizing books, scholarly articles, and any other materials pertinent to the specific issue, field of research, or theory. The purpose of the literature review is to lay the groundwork for the investigation, highlight the research value of the chosen topic, and provide the research basis for writing the dissertation [78]. The literature review must provide a thorough examination of the breadth and depth of existing research on the dissertation topic, as well as the findings and results obtained in order to identify gaps in previous research or areas of insufficiency and highlight points of focus and innovation in the researcher's own research on the topic [79].

In the context of this research, the literature review is crucial for several reasons. Firstly, it allows for a comprehensive understanding of the theoretical foundations and practical applications of Federated Learning and Reinforcement Learning within embedded systems. By examining existing studies, we can identify successful methodologies, prevalent challenges, and the technological landscape surrounding these approaches. This understanding is essential for framing our research questions and objectives, ensuring they are relevant and grounded in current knowledge.

In our study, we use a literature review to better understand the concept of Federated Learning and the limitations and constraints that exist in the current Federated Learning systems reported in the literature. The data retrieved from each study was primarily focused on the area of study and the technical problems solved by using Federated Learning. We searched papers from various high-ranked journals and conferences to give a current literature overview of Federated Learning in the software engineering research domain. As a result, it can provide a comprehensive picture of the most recent methodologies used, as well as the limitations and obstacles in today's Federated Learning systems, which can lead our future research path. The technique also validates the significance of our research inquiries by situating them within the broader discourse on FL and RL. By referencing established findings, we can articulate the importance of our study and demonstrate how it builds upon or diverges from previous work. This establishes a strong rationale for our research and reinforces its relevance to the ongoing developments in these fields.

In summary, the literature review is an important element of our research methodology. It not only lays a solid theoretical foundation but also identifies gaps and trends that inform our research questions. Through this comprehensive exploration, we aim to advance the understanding of how FL and RL can be effectively integrated into embedded systems, ultimately enhancing their adaptability and performance in dynamic environments.

3.3.2 Case Study

A case study is a qualitative research method that provides an in-depth analysis of a specific phenomenon within its real-world context [80][81]. This method is especially useful for exploring complex issues where the interplay between the phenomenon and its environment is critical to understanding the outcomes. As Yin [82] notes, case studies can serve both exploratory and explanatory purposes, making them versatile tools for researchers seeking to uncover the underlying mechanisms of particular events or situations.

The strength of case studies lies in their ability to capture rich, contextual data through various means, including interviews, observations, and document analysis [83]. This multi-faceted approach allows researchers to gather nuanced insights that may not be readily apparent through quantitative methods. By focusing on specific instances or cases, researchers can identify patterns, discern underlying factors, and explore the complexities of interactions involved in the phenomenon of interest [80].

In our study, we adopted a multi-case study approach across both the automotive and telecommunication domains to investigate the practical applications of Federated Learning and Reinforcement Learning in embedded systems. This method enables us to engage directly with multiple case companies in each domain that have implemented these technologies, allowing us to explore the real-world challenges and benefits they experience. By examining diverse organizational contexts, we can gain a comprehensive understanding of FL and RL, including the obstacles encountered during deployment and the strategies employed to overcome them. We adopted the case study method with semi-structured interviews and simulation experiments to thoroughly investigate the challenges inherent in current machine learning workflows within the embedded systems domain. Our goal was to not only identify the limitations of existing systems but also to assess the benefits that Federated Learning and Reinforcement Learning can offer. By employing this multi-faceted approach, we aimed to uncover the primary constraints and limitations faced by practitioners, and subsequently propose actionable solutions tailored to real-world applications of FL and RL.

The exploratory nature of the case study method enables us to dive deeply into the complexities involved in implementing Federated Learning within embedded systems [80]. This approach provides a rich context for understanding these technologies, the specific challenges encountered by organizations, and the strategies they employ to navigate these obstacles. Through detailed observations and interactions with stakeholders, we gain valuable insights into the nuances of deploying FL and RL in dynamic environments.

In addition to case studies, we conducted semi-structured individual interviews to elicit detailed feedback from industry experts and practitioners. This qualitative data collection technique allowed us to gather diverse perspectives and experiences, enriching our understanding of the practical implications of Federated Learning and Reinforcement Learning. We complemented our qualitative findings with simulation experiments, which provided a controlled environment to test and validate our proposed solutions. By simulating various scenarios and conditions, we were able to evaluate the performance of our algorithms and frameworks, ensuring they meet the requirements of embedded systems.

The combination of these research techniques—case studies, interviews, and simulations—creates a comprehensive framework for understanding the current landscape of machine learning in embedded systems. In the following sections, we will elaborate on the specific methodologies employed for the interviews and simulation experiments, detailing how these techniques contribute to addressing our research questions and advancing the field.

1) Interviews: The interview technique employed in this research is a semi-structured format, which strikes a balance between guided inquiry and the flexibility to explore emerging topics [84][85]. Semi-structured interviews are characterized by a combination of predetermined questions and open-ended prompts, allowing interviewees to provide in-depth responses while also giving the interviewer the freedom to probe further into relevant areas of interest that may arise during the conversation [86][87].

This method is particularly advantageous in our study, as it facilitates rich, qualitative data collection from industry experts and practitioners in the embedded systems field. By utilizing semi-structured interviews, we can explore participants' experiences, challenges, and insights regarding the implementation of Federated Learning and Reinforcement Learning in real-world contexts. This approach encourages interviewees to share their perspectives on the limitations of current machine learning workflows, the potential benefits of FL and RL, and the specific constraints they face in their organizations.

The semi-structured format allows for a deep exploration of topics such as data privacy concerns, communication costs, performance issues, and the adaptability of embedded systems in dynamic environments. It fosters a conversational atmosphere, encouraging participants to discuss their thoughts and experiences candidly, which can lead to unexpected insights that might not emerge from more rigid interview formats [84].

To ensure consistency and reliability, we developed an interview guide that outlines key themes and questions related to our research objectives for research paper A and B during problem identification phase. However, we also remained open to diverging from the guide when the discussion guaranteed it [85]. Each interview was conducted in a manner that encouraged dialogue and elaboration, allowing us to gather nuanced information that enriches our understanding of the challenges and opportunities in the domain of FL and RL.

The qualitative data collected through these interviews complement our case study findings and simulation results, providing a holistic view of the complexities involved in implementing Federated Learning and Reinforcement Learning in embedded edge systems[87].

2) Simulation Experiments: In our research, simulation experiments play a crucial role in validating the proposed algorithms and frameworks related to Federated Learning and Reinforcement Learning within embedded systems. Simulation is a powerful technique that allows researchers to model and analyze complex systems in a controlled environment, enabling the exploration of various scenarios and conditions that may be impractical or impossible to test in real-world settings [88].

Simulation experiments provide a means to evaluate the performance, scalability, and adaptability of our proposed solutions without the constraints and uncertainties that come with live deployment [89][90]. By creating a virtual representation of the embedded systems environment, we can manipulate variables, test different configurations, and observe the behavior of the FL and RL algorithms under diverse conditions. This iterative process allows us to refine our models, identify potential weaknesses, and optimize performance before implementation in real-world applications [91].

In this study, the simulation experiments focus on key aspects such as communication costs, model accuracy, latency, and the ability to adapt to dynamic

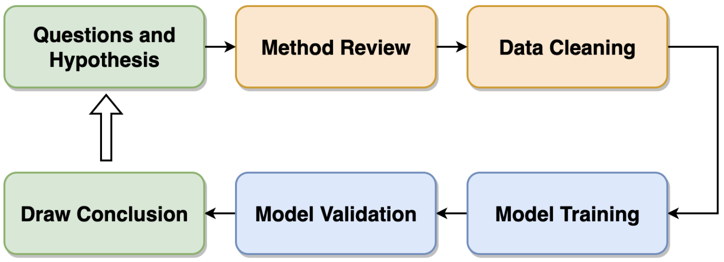


Figure 3.1: The simulation process (cycle)

changes in the environment. By simulating various scenarios, we can assess how well our solutions perform when faced with real-world challenges, such as limited computational resources, varying data distributions, and different operational contexts.

Moreover, simulations provide a platform for conducting “what-if” analyses, helping us understand the potential impact of different design choices and algorithmic adjustments [88]. This flexibility is particularly valuable in the context of embedded systems, where constraints such as power consumption and processing capability must be carefully balanced.

Through the insights gained from simulation experiments, we are able to inform our practical implementations and enhance our understanding of how Federated Learning and Reinforcement Learning can be effectively applied in the embedded systems domain. In the following sections, we will outline the specific methodologies employed in our simulations, including the setup, parameters, and metrics used for evaluation. In this section, we present the simulation techniques (Figure 3.1) used in this thesis, which includes questions and hypothesis raising, methods review, model training (hyper-parameter seeking), model validation, and conclusion drafting [92].

1. Questions and Hypothesis: In this initial step, we articulate the research questions and hypotheses that align with the objectives of our study. These questions serve as a foundation for guiding the selection of technologies and the design of validation cases for the simulations.

2. Methods Review: Following the formulation of research questions and the preparation of data, we conduct a review of various commonly used

machine learning algorithms, methods, and frameworks pertinent to our inquiry. This evaluation considers the diverse purposes these models serve, as different architectures are optimized for specific types of data, such as text or images. We also review aggregation protocols tailored to our case scenarios.

3. Data Cleaning: This phase involves implementing data cleaning strategies to eliminate low-variance features, constant values, and outliers. A key aspect of data preparation is splitting the dataset, typically distributing it evenly across edge devices. For each local edge device, approximately 70% of the data will be designated for training the model, while the remaining 30% will be reserved for evaluation. This separation is crucial, as using the same data for both training and evaluation would compromise the fairness of the model's performance assessment in real-world applications.

4. Model Training: During this step, we focus on training the models. To ensure the models achieve the desired quality for edge clients, we explore various hyperparameters using a random search strategy. Throughout the training process, the quality of the local edge models is continuously enhanced through the aggregation of models from multiple sources.

5. Model Validation: Once the model training is complete, we validate the model to assess its effectiveness in realistic conditions. A subset of the evaluation dataset is employed to test the model's proficiency in scenarios that differ from its training data. If the model's performance does not meet expectations, we revisit the hyperparameter settings. Evaluation is critical in business contexts, allowing data scientists to confirm that their objectives are being met. If results are subpar, earlier steps must be scrutinized to identify the causes of underperformance. Inadequate evaluation may result in the model failing to meet essential business requirements.

6. Draw Conclusion: The final step involves analyzing the performance metrics and drawing conclusions to address the original research questions. This analysis helps determine whether the initial research goals have been met and informs the next steps for future research directions.

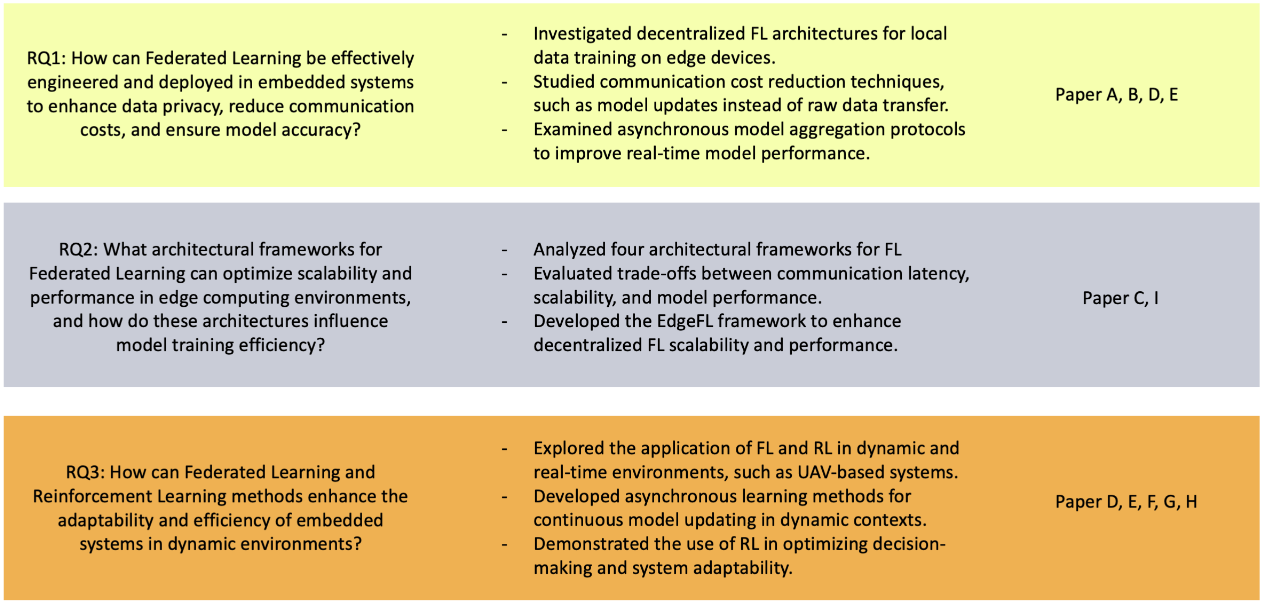
We chose this method to develop new algorithms and frameworks aimed at addressing the challenges related to Federated Learning and Reinforcement Learning faced by companies in the embedded systems domain. Additionally, we validated these proposed approaches using empirical scenarios and datasets. This methodology enables us to acquire in-depth practical knowledge about deploying Federated Learning in real-world contexts.

3.4 Research Design

This section outlines the planning of our research, focusing on the identification of challenges within Federated Learning and Reinforcement Learning systems, as well as the barriers organizations face in integrating these components into their embedded systems. Aligned with our research questions, we first examine how FL and RL can be effectively engineered to enhance data privacy, reduce communication costs, and ensure model accuracy (RQ1). Next, we investigate architectural frameworks that can optimize scalability and performance in edge computing environments, and assess how these frameworks influence model training efficiency (RQ2). Lastly, we explore how the integration of FL and RL methods can improve the adaptability and efficiency of embedded systems in dynamic environments (RQ3). The research is structured into two main phases: Problem Identification and Solution Proposal and Validation. An overview of the research activities within these phases is presented in Table 3.1 and a mapping of research questions, activities, and papers is illustrated in Figure 3.2.

3.4.1 Problem Identification

The first phase focuses on identifying challenges and gathering insights regarding the current Federated Learning and Reinforcement Learning systems. To begin, a comprehensive literature review was conducted to establish a foundational understanding of existing research and applications in this domain. This review helped us formulate our first and second research questions, offering clarity on the current state of FL and RL systems and their applications. We posed critical questions such as, "What are the primary advantages of implementing machine learning methods in edge embedded systems?" and "What are the main obstacles and limitations of existing systems?"



RQ1: How can Federated Learning be effectively engineered and deployed in embedded systems to enhance data privacy, reduce communication costs, and ensure model accuracy?	- Investigated decentralized FL architectures for local data training on edge devices. - Studied communication cost reduction techniques, such as model updates instead of raw data transfer. - Examined asynchronous model aggregation protocols to improve real-time model performance.	Paper A, B, D, E
RQ2: What architectural frameworks for Federated Learning can optimize scalability and performance in edge computing environments, and how do these architectures influence model training efficiency?	- Analyzed four architectural frameworks for FL - Evaluated trade-offs between communication latency, scalability, and model performance. - Developed the EdgeFL framework to enhance decentralized FL scalability and performance.	Paper C, I
RQ3: How can Federated Learning and Reinforcement Learning methods enhance the adaptability and efficiency of embedded systems in dynamic environments?	- Explored the application of FL and RL in dynamic and real-time environments, such as UAV-based systems. - Developed asynchronous learning methods for continuous model updating in dynamic contexts. - Demonstrated the use of RL in optimizing decision-making and system adaptability.	Paper D, E, F, G, H

Figure 3.2: Mapping of Research Questions, Activities, and Papers

Table 3.1: Overview of the research activities

Research Phase	Research Questions	Research Objectives	Data Collection and Evaluation Methods	Case Companies	Participants Role
Problem Identification	How can Federated Learning be effectively engineered and deployed in embedded systems to enhance data privacy, reduce communication costs, and ensure model accuracy?	To identify the challenges of the traditional machine learning workflow and the benefits of Federated Learning	Literature Review, Case Study with semi -structured interviews	Ericsson	Head of Automation and AI, AI Systems Developer, Data and Analytic Manager, etc
Solution Proposal and Validation	What architectural frameworks for Federated Learning can optimize scalability and performance in edge computing environments, and how do these architectures influence model training efficiency?	To propose solutions and validate algorithm or frameworks on empirical datasets	Case Studies with simulation experiments	Volvo Cars, Scania, Ericsson	Senior Data Scientist, Analytic System Architect, Data engineer, Software Developer, etc
	How can Federated Learning and Reinforcement Learning methods enhance the adaptability and efficiency of embedded systems in dynamic environments?				

To dive deeper into the barriers hindering industrial implementation of FL and RL components, we selected Ericsson as our exploratory case company. An interview-based case study was conducted involving ten experienced engineers from Ericsson, which is actively seeking ways to ensure consistent service quality for its large-scale and distributed customer base. This company was chosen due to its extensive experience and its exploration of the potential applications of both FL and RL.

During the study, we identified the specific challenges companies face when deploying FL and RL in an industrial setting. Data was primarily collected through semi-structured interviews with practitioners at Ericsson who are involved in designing or utilizing machine learning applications, particularly those engaged in data engineering or with expertise in both FL and RL. Drawing from their professional experiences, these practitioners provided insights into the issues they encounter with traditional machine learning workflows and shared their perspectives on how FL and RL could serve as solutions for future intelligent industrial applications.

Additionally, we examined the potential benefits and challenges of integrating FL and RL, alongside exploring how organizations in the embedded systems field could transition from conventional machine learning to incorporate these advanced methods.

3.4.2 Solution Proposal and Validation

In the second phase, we built upon the conclusions and insights gathered from the first phase to develop comprehensive solutions, analyses, and validations aimed at assisting companies in integrating Federated Learning and Reinforcement Learning components into their systems. This stage was essential for translating theoretical understanding into practical applications that address the challenges identified in the previous phase.

We engaged in a thorough examination of various architectural designs that could be employed in real-world industrial applications, focusing on how these architectures can effectively mitigate the challenges associated with implementing FL and RL. Collaborating closely with industry leaders like Volvo Cars and Scania allowed us to leverage their expertise and insights, ensuring our solutions were grounded in practical realities. We organized weekly workshops to facilitate the exchange of knowledge, sharing our latest findings and gathering valuable feedback from senior data scientists, AI system developers, and architects. This iterative process fostered an environment of collaboration, ensuring that our work was relevant and directly applicable to industry needs.

In our analysis, we assessed the performance of different FL architectures, summarizing their strengths and weaknesses. This involved detailed comparisons of how various designs could be deployed across different industrial scenarios. By identifying architecture alternatives that align with specific operational requirements, we provided actionable insights that organizations could use to tailor their approaches to FL and RL integration.

To tackle real-time deployment challenges, we developed an innovative asynchronous aggregation protocol. This protocol is particularly beneficial in environments characterized by diverse hardware settings, as it allows for flexibility and adaptability in the deployment of FL systems. It enhances the robustness of the learning process by ensuring that model updates can be aggregated efficiently, even when devices have varying computational capabilities or connectivity issues.

Furthermore, we explored methodologies to combine the asynchronous Federated Learning algorithm with several established machine learning techniques. This approach not only increased the efficiency of model learning but also minimized communication overhead, a critical concern in federated settings where bandwidth may be limited. By refining the sharing method, we were able to optimize the performance of FL systems while reducing the resource demands on edge devices.

All proposed solutions underwent rigorous validation using empirical datasets, focusing particularly on high-impact use cases in the autonomous driving sector. This domain was chosen due to its complex requirements and the critical need for robust, scalable machine learning solutions. Through our validations, we demonstrated the practicality and effectiveness of our strategies, providing clear evidence of how FL and RL can drive significant advancements in embedded systems.

The second phase of our research not only aimed to address theoretical gaps but also to create practical solutions that can be readily implemented in real-world contexts, ultimately contributing to the advancement of Federated Learning and Reinforcement Learning applications in the embedded systems domain.

Table 3.2: Overview of activities during the case study combined with the interviews

Duration	Activities	Details	Reflection
1 week	Presentation of the concept of Federated Learning and Reinforcement Learning during the company workshop	Feedback from the various team in Ericsson	Participants confirmed the potential of Federated Learning and the challenges they encountered
1 week	Initiating meeting	Discussion with managers in Ericsson and Steering committee at Software Center	Project approval
1 week	Question Design	Meeting with key participants at the researcher side regarding the questions of Federated Learning and Reinforcement Learning that were interested by the researchers	Come up with the question list
6 months	Schedule meetings with different teams to explore the questions	10 semi-structured meetings at Ericsson	Identify the challenges, solutions, and improvements that need to be implemented if employing Federated Learning in a real-world context.
2 months	Weekly presentation to update the progress and collect feedback	8 Follow-up meeting with Ericsson Participants	Analysing the data collected and sharing the ideas regarding the solution of Federated Learning
1 week	Conduct a reporting workshop to give a final presentation of the result	Follow-up workshop at Ericsson	Identification of the current results and the potential improvements for future research

Table 3.3: Overview of activities during the case study combined with the simulation experiments

Duration	Activities	Details	Reflection
1 week	Presentation of Federated Learning and Reinforcement Learning results during the company workshop	Feedback from the various teams in Volvo Cars, Scania, Ericsson	Participants confirmed the concept of Federated Learning, Reinforcement Learning and potential applications in their fields and the challenges they encountered
1 week	Initiating meeting	Discussion with managers in Volvo Cars, Scania and Steering committee at Software Center	Project approval
1 week	Project plan meeting with the key participants from Volvo Cars and Scania	Identification of the problems they encountered and possible solutions, framework, architecture	Come up with a project plan and possible outcomes
2 week	Data and case description meeting	Description of the data and cases that were going on in the companies	Decision on the simulation data, cases and platforms
3-4 months	Weekly presentation to update the progress and collect feedback	16 Follow-up meetings with Volvo Cars, Scania, Ericsson Participants	Analysing the simulation results and share the algorithm, frameworks regarding the solution of Federated Learning
1 week	Conduct a reporting workshop to give a final presentation of the result	Follow-up workshop at Volvo Cars, Scania, Ericsson	Identification of the current results and the potential improvements for future research

3.5 Industrial Collaboration

This research was carried out through a close partnership between academia and industry, facilitated by the Software Center [93]. The Software Center is a collaborative research initiative comprising 15 enterprises and 5 academic institutions, dedicated to enhancing the digitalization capabilities of Europe's software-intensive industries. Companies within the embedded systems sector are particularly interested in effectively integrating AI, specifically machine learning and deep learning, into their systems to accelerate digitization and enhance service quality.

Throughout the project, the Software Center organized various seminars and workshops aimed at bridging the gap between researchers and industry practitioners, thereby deepening our understanding of the challenges faced by companies in the embedded systems domain. Our research involved collaboration with several organizations from the Software Center, including Ericsson, Scania, and Volvo, to explore the difficulties they encounter when utilizing machine learning methods and how Federated Learning could address these challenges.

Through the case study approach, we gained insights into deploying Federated Learning and Reinforcement Learning components in real-world contexts, along with the specific considerations engineers must take into account. In the problem identification phase, we worked closely with Ericsson to pinpoint the challenges faced by companies, as well as the limitations and constraints associated with Federated Learning. This involved conducting ten semi-structured interviews with a diverse group of professionals, including four experienced machine learning engineers, three data experts, two project managers, and one analytics architect. During these interviews, we discussed the problems, potential solutions, and necessary enhancements for the practical implementation of Federated Learning.

To further enrich our understanding, we organized eight follow-up meetings with participants from Ericsson to analyze the data collected and discuss insights regarding the Federated Learning solution. The culmination of this phase was a reporting workshop where we presented our findings.

In the subsequent solution development and validation phase, we collaborated with Volvo Cars and Scania to validate our proposed solutions, algorithms, and frameworks through real-world automotive case studies. Throughout this phase, we held 16 follow-up meetings with representatives from Volvo

Cars and Scania, which included six machine learning experts and two project managers. These meetings facilitated discussions on simulation results and the sharing of algorithms and frameworks for the Federated Learning solutions. The final stage involved a reporting workshop to present the data, highlight current findings, and recommend improvements for future research.

The activities undertaken during the case studies, interviews, and simulation experiments in collaboration with these companies are detailed in Tables 3.3 and 3.2.

3.6 Threats to Validity

3.6.1 Construct validity

Construct validity pertains to the extent to which the empirical results reflect the intended constructs of the study. While the findings presented in this thesis are inherently subjective, as they are based on the experiences of selected participants, the depth of insights is bolstered by the extensive expertise of these professionals. Consequently, the conclusions drawn are primarily applicable to the specific domain and scenarios explored in this research [94]. However, the ideas and outcomes may also hold relevance beyond the immediate context. The authors and participants involved possess significant experience in machine learning, Federated Learning, and data engineering, which strengthens the credibility of the findings. Additionally, in instances where industry-specific jargon or specialized terminology might be unclear, the authors, drawing on their dual expertise in academia and industry, were able to clarify and contextualize these concepts. Therefore, the risk of construct validity threats is minimized.

3.6.2 Conclusion validity

Conclusion validity concerns the accuracy of the inferences drawn from the data regarding the relationships between variables [95], [96]. To safeguard against threats to conclusion validity, several strategies were implemented throughout the study. First, potential biases arising from researchers' preconceived notions or similar findings in existing literature were actively mitigated. In the second phase of research, data were gathered from multiple sources, including documentation and simulations, in addition to the interviews and literature reviews. This triangulation of data formats helped to reduce bias that could arise from reliance on a single data collection method. Furthermore, the Software Center facilitated workshops and seminars where the findings were presented and discussed with participants from the case companies. These sessions were instrumental in validating the results, ensuring that interpretations were accurate and comprehensive. Overall, feedback from participants overwhelmingly confirmed the findings.

3.6.3 Internal validity

Internal validity refers to the degree to which the study's design and methodology support a causal relationship between the observed variables, minimizing the influence of confounding factors [96]. To enhance internal validity in this research, several key measures were adopted. First, participant selection was carefully controlled, ensuring that all individuals involved had extensive, relevant experience in machine learning, Federated Learning, and embedded systems. This selective approach aimed to reduce variability stemming from differences in participant expertise, thereby strengthening the reliability of the findings. Additionally, data collection methods, such as semi-structured interviews and simulation experiments, were standardized to prevent inconsistencies that might otherwise affect the results.

The study's iterative approach further supported internal validity by refining the research design at each phase based on feedback from workshops and validation sessions held with case company stakeholders. These sessions provided an opportunity to clarify ambiguous findings, address potential misconceptions, and confirm that conclusions accurately reflected the participants' experiences. Moreover, data triangulation from multiple sources, documentation, simulations, and participant feedback, helped ensure that interpretations were well-founded and that potential confounding factors were systematically addressed. Thus, these combined measures helped to limit threats to internal validity and strengthen the study's internal coherence.

3.6.4 External validity

External validity relates to the extent to which the findings can be generalized beyond the specific context of the study. This research involved close collaboration with multiple companies, and data were collected from participants across various teams and domains, with simulations conducted on diverse industrial cases. Terminology used within the companies was standardized, and the implementation details were thoroughly documented [97]. Nonetheless, given the focus on embedded systems, caution must be exercised in generalizing these results to the broader industry. That said, the authors believe that there are significant parallels between the case studies presented and other organizations operating in regulated fields, suggesting potential applicability of the findings in similar contexts.

3.7 Summary

In this chapter, we outlined the research methodology employed to explore the role of Federated Learning and Reinforcement Learning in enhancing the adaptability, scalability, and privacy of embedded systems in edge computing environments. The research was structured around three core objectives: (1) Engineering and deploying FL for improved privacy and communication efficiency, (2) Developing scalable architectures for FL, and (3) Enhancing system adaptability and efficiency using approaches with FL and RL.

To address these objectives, we utilized a mixed-methods approach, combining literature reviews, case studies, and empirical experiments. We conducted nine studies that serve as the foundation of this thesis, with each paper contributing uniquely to the research questions posed. The first set of papers focused on the design and deployment of Federated Learning systems in real-world contexts, providing insights into how FL can be engineered to enhance privacy and reduce data transmission costs. The second set explored the architectural frameworks required to scale FL in dynamic environments, assessing their impact on performance and model training efficiency. The final set examined how FL and RL can work in terms of enabling adaptive, efficient systems capable of learning and responding to real-time environmental changes. A mapping between the research objectives, research questions, and the papers reviewed was established, ensuring that each study fills a specific role in addressing the challenges within embedded systems domain.

CHAPTER 4

Contributions of this thesis

In this chapter, we describe the key objectives that form the core contributions of this thesis, showing how they address the primary research questions. Following this, we provide a summary of each included publication, outlining the methods employed and the main results achieved.

4.1 Overview

Objective 1: Engineering and Deploying Federated Learning for Enhanced Privacy and Efficiency

The first key objective of this thesis is to design and implement Federated Learning systems in embedded environments to address key challenges such as data privacy, communication efficiency, and model accuracy. This objective responds to RQ1, which focuses on how FL can be effectively engineered and deployed to meet these critical requirements.

Papers A and B provide a foundational understanding of how FL systems can be structured and adapted for real-world applications. Paper A offers a comprehensive review of the engineering challenges and solutions in deploying FL, highlighting its advantages in maintaining data privacy by keeping sensitive information local to the device. Paper B takes a case-study approach within the telecommunications domain, where semi-structured interviews reveal the practical concerns and opportunities industries face when transitioning to FL systems.

To further address the issue of model accuracy and communication efficiency, Papers D and E introduce asynchronous model aggregation methods. These methods allow local models to be updated and aggregated at different times, thereby reducing communication overhead and allowing the models to better adapt to the local data. By implementing these strategies, FL can significantly reduce data transmission and storage costs while maintaining high levels of model accuracy. This makes FL a powerful tool for embedded systems where data privacy and limited resources are key constraints.

Through this objective, the thesis shows that FL can be effectively engineered to address the critical concerns of data privacy, communication efficiency, and model performance in embedded systems, making it a viable solution for real-world applications with stringent privacy and resource requirements.

Objective 2: Developing Scalable Federated Learning Architectures

The second key objective is to develop scalable architectural frameworks for Federated Learning that optimize both performance and resource management in edge computing environments. This objective corresponds to RQ2, which seeks to understand how different FL architectures can improve scalability, performance, and model training efficiency in real-world scenarios.

Paper C provides a comparative analysis of several architectural frameworks, including centralized, decentralized, regional, and hierarchical approaches. Each of these architectures offers different trade-offs in terms of scalability, communication overhead, and fault tolerance. The decentralized architecture, for example, eliminates the need for a central server, thereby reducing bottlenecks and improving fault tolerance. This approach enhances the scalability of the system by allowing more edge devices to participate in model training without overwhelming a single point of failure.

Paper I introduces the EdgeFL framework, which builds on these architectural principles by proposing a decentralized approach that also minimizes the coordination overhead associated with FL. EdgeFL allows more efficient coordination between edge devices, further improving the scalability and performance of FL in resource-constrained environments. Together, these papers emphasize the importance of choosing the right architecture to balance communication costs, training efficiency, and system scalability.

By focusing on scalable architectural designs, this thesis contributes to the growing field of FL, providing insights into how different frameworks can be implemented to handle the demands of large-scale, real-world edge computing environments. This makes FL a more practical and efficient tool for industries that rely on distributed systems.

Objective 3: Enhancing System Adaptability and Efficiency with Federated and Reinforcement Learning

The third key objective is to explore how the integration of Federated Learning and Reinforcement Learning can enhance the adaptability and efficiency of embedded systems operating in dynamic environments. This directly addresses RQ3, which focuses on how these two learning approaches can be used to improve system performance in scenarios where environmental conditions are constantly changing.

Papers F, G, and H propose the application of RL in dynamic, real-time environments, particularly in the telecommunications and autonomous systems sectors. For example, Paper G illustrates how deep Reinforcement Learning can be applied to optimize network configurations in rapidly changing environments, ensuring that the system adapts to shifting demands in real time. This adaptability is critical in scenarios such as autonomous vehicles or telecommunications systems, where conditions can change suddenly, and systems must respond immediately to maintain performance.

Paper H extends this concept by applying RL to UAV-based communication networks, demonstrating how RL can autonomously configure system operations to maintain optimal performance. Meanwhile, Paper F presents a case study on how deep RL methods can enable dynamic decision-making in unpredictable environments, such as emergencies or high-traffic situations.

Paper D, E also contributes to this objective by showing how asynchronous FL techniques allow for real-time updates and more flexible system behavior in response to changing conditions. Asynchronous learning helps embedded systems adapt more quickly to local data changes without requiring constant coordination with a central server.

This thesis demonstrates how these two methods can significantly enhance the adaptability and operational efficiency of embedded systems. FL ensures that systems can learn from distributed data sources without compromising privacy, while RL enables the systems to make autonomous decisions in real time. This integrated approach creates a powerful framework for building intelligent systems capable of responding to complex, dynamic environments. A mapping of research objectives, research questions, and output papers is illustrated in Figure 4.1.

4.2 Included publications

4.2.1 Paper A: Engineering Federated Learning Systems: A Literature Review

4.2.1.1 Summary

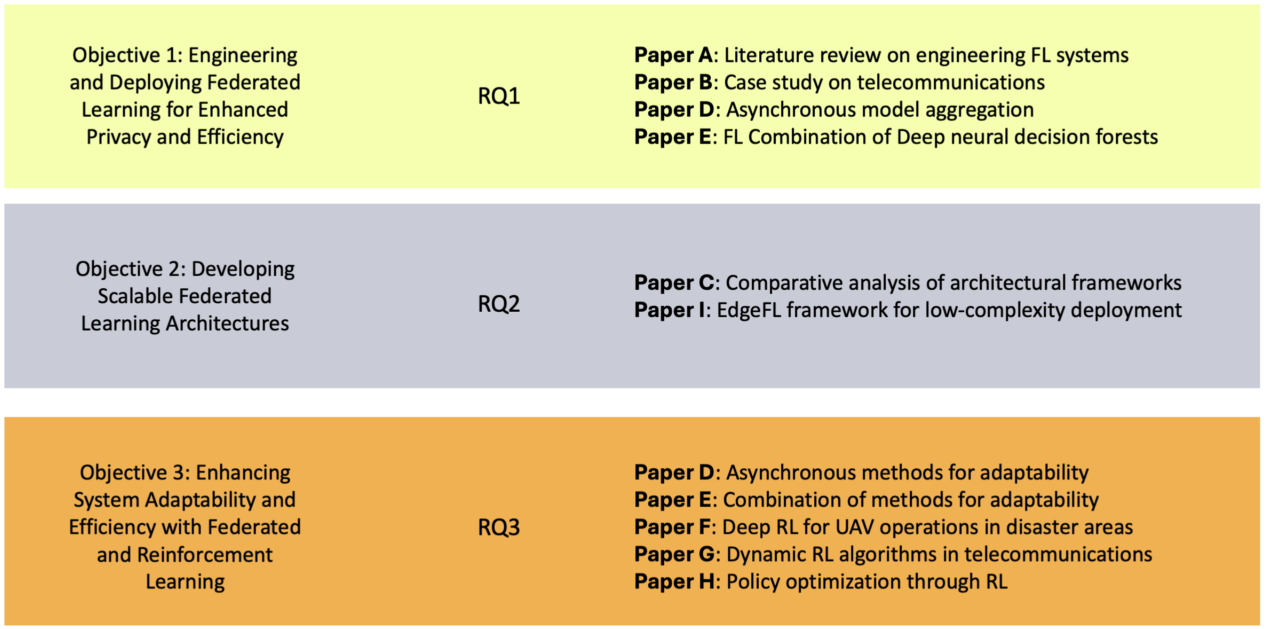
This paper provides a comprehensive review of Federated Learning systems, focusing on their role in distributed edge environments. It emphasizes how FL enhances data privacy by keeping user data localized and reduces communication costs by training models directly on edge devices. The paper categorizes existing FL implementations across various domains and highlights key challenges such as system scalability, model accuracy, and resource limitations. It concludes by proposing several open research questions in the field.

4.2.1.2 Research Method

A literature review is conducted, analyzing numerous case studies and academic papers on Federated Learning. The review spans multiple fields, from telecommunications to healthcare, examining FL's deployment across different sectors. This methodology involved categorizing solutions based on architecture, privacy mechanisms, and communication strategies. Additionally, a comparative analysis of FL's advantages and limitations was performed.

4.2.1.3 Main Results

The literature review identified common challenges faced by FL systems, such as model heterogeneity and data distribution discrepancies. It also pointed out the need for more standardized benchmarks in FL research. Key contributions include a detailed categorization of FL use cases and a proposal for future



Objective 1: Engineering and Deploying Federated Learning for Enhanced Privacy and Efficiency	RQ1	**Paper A**: Literature review on engineering FL systems **Paper B**: Case study on telecommunications **Paper D**: Asynchronous model aggregation **Paper E**: FL Combination of Deep neural decision forests
Objective 2: Developing Scalable Federated Learning Architectures	RQ2	**Paper C**: Comparative analysis of architectural frameworks **Paper I**: EdgeFL framework for low-complexity deployment
Objective 3: Enhancing System Adaptability and Efficiency with Federated and Reinforcement Learning	RQ3	**Paper D**: Asynchronous methods for adaptability **Paper E**: Combination of methods for adaptability **Paper F**: Deep RL for UAV operations in disaster areas **Paper G**: Dynamic RL algorithms in telecommunications **Paper H**: Policy optimization through RL

Figure 4.1: Mapping of Research Objectives, Research Questions and Papers

research directions, including the development of more efficient aggregation algorithms and frameworks tailored to specific industries.

4.2.2 Paper B: Towards Federated Learning: A Case Study in the Telecommunication Domain

4.2.2.1 Summary

This paper presents a case study investigating the adoption of Federated Learning in the telecommunications industry. The study explores the reasons why businesses see FL as promising and identifies the key services that an effective FL system must offer. It also highlights the difficulties industries encounter in transitioning to FL and suggests practical guidelines for implementing reliable FL systems.

4.2.2.2 Research Method

The paper utilizes a semi-structured interview methodology to gather insights from various companies in the telecommunications industry. Interviews were conducted with stakeholders involved in decision-making processes related to the adoption of FL. The qualitative data gathered from these interviews were analyzed to identify common trends, barriers, and motivations behind FL implementation. The research focused on understanding how businesses perceive FL's potential and what obstacles they face in adopting it.

4.2.2.3 Main Results

The case study revealed that companies are attracted to Federated Learning due to its potential to reduce data privacy concerns and improve efficiency by processing data locally on edge devices. However, many struggle with integrating FL components into existing systems due to technical complexities and the lack of standardized frameworks. The study identified five key criteria necessary for successfully deploying FL in industrial settings, including scalability, model accuracy, and system compatibility.

4.2.3 Paper C: Federated Learning Systems: Architecture Alternatives

4.2.3.1 Summary

This paper explores various architectural alternatives for Federated Learning systems, comparing centralized, hierarchical, regional, and fully decentralized approaches. It discusses the trade-offs involved in each architecture and provides insights into how system performance, scalability, and fault tolerance can be optimized depending on the chosen framework.

4.2.3.2 Research Method

We performed a comprehensive analysis of different FL system architectures, using simulation-based models to compare their performance under varying conditions, such as communication latency, edge device capabilities, and data distribution. The architectures were tested in environments that mimic real-world constraints like limited bandwidth and device heterogeneity.

4.2.3.3 Main Results

The study found that decentralized FL architectures tend to provide better scalability and fault tolerance compared to centralized and hierarchical alternatives. However, there are trade-offs in terms of communication latency and model accuracy. For example, while fully decentralized systems reduce bottlenecks and single points of failure, they sometimes suffer from delayed convergence. The results suggest that hybrid architectures, which balance decentralization with some form of hierarchy, may provide the best of both worlds for most edge computing environments.

4.2.4 Paper D: Real-Time End-to-End Federated Learning: An Automotive Case Study

4.2.4.1 Summary

This paper examines the application of real-time end-to-end Federated Learning in the automotive sector, focusing on the use of connected vehicles for autonomous driving systems. The study showcases how FL can enable vehicles to continuously learn and adapt to new driving conditions without compromising data privacy, by keeping data localized to each vehicle.

4.2.4.2 Research Method

The methodology involved implementing a Federated Learning system in an automotive testbed. The system was deployed in autonomous vehicles where each vehicle trained its local model using real-time sensor data. The models were then asynchronously aggregated without interrupting vehicle operation. Metrics for accuracy and latency were collected to compare with a traditional centralized model.

4.2.4.3 Main Results

The results demonstrated that FL could effectively enable vehicles to learn autonomously while significantly reducing the need for continuous data uploads to the cloud. The global model, regularly updated with contributions from individual vehicles, was able to maintain high accuracy in detecting objects and navigating under changing conditions. However, the study also highlighted challenges related to network latency and model synchronization, suggesting the need for future research into optimizing communication protocols for real-time applications.

4.2.5 Paper E: Asynchronous Federated Learning of Deep Neural Decision Forests

4.2.5.1 Summary

This paper focuses on the development of an asynchronous Federated Learning model using Deep Neural Decision Forests (DNDFs). It proposes a solution to improve learning efficiency by allowing participants in the FL system to update their local models asynchronously, rather than waiting for synchronized updates from all nodes.

4.2.5.2 Research Method

The proposed algorithm was implemented in a real-world automotive system, where decision trees were used to classify sensor data. The learning process was distributed across several autonomous vehicles, each training its model locally. Asynchronous updates were aggregated in a federated manner. Metrics on communication overhead, classification accuracy, and computational efficiency were collected and analyzed.

4.2.5.3 Main Results

The asynchronous approach demonstrated a significant improvement in communication efficiency, as devices no longer had to wait for all participants to synchronize updates. This led to faster convergence times without a notable loss in model accuracy. The results suggest that asynchronous FL models could be particularly beneficial in dynamic environments where devices have varying connectivity or computational power, making them a strong candidate for real-time applications.

4.2.6 Paper F: Autonomous Navigation and Configuration of Integrated Access Backhauling for UAV Base Station Using Reinforcement Learning

4.2.6.1 Summary

This paper explores the use of Reinforcement Learning for autonomous navigation and configuration of Unmanned Aerial Vehicles (UAVs) functioning as base stations in a 5G network. The study demonstrates how RL can enable UAVs to dynamically adapt to changing network conditions and user demands, optimizing their positioning and resource allocation.

4.2.6.2 Research Method

A Reinforcement Learning model was trained using simulated data from UAV deployments. The algorithm was tasked with adjusting UAV positioning and configurations to optimize communication under various network load conditions. The model's performance was validated through simulations that represented emergency communication scenarios.

4.2.6.3 Main Results

The results showed that the RL algorithm allowed UAVs to efficiently manage network traffic and respond to changes in user density and network congestion. The system significantly reduced network latency and improved overall connectivity in the simulation. The study concluded that RL-based approaches are promising for future dynamic and flexible 5G networks, where UAVs can play a crucial role in maintaining high-quality service in challenging environments.

4.2.7 Paper G: Deep Reinforcement Learning in a Dynamic Environment: A Case Study in the Telecommunication Industry

4.2.7.1 Summary

This paper presents a case study that examines the use of Deep Reinforcement Learning in dynamic environments within the telecommunication industry. The study focuses on how DRL can help optimize resource management and service delivery in situations where network conditions and user demands are constantly changing.

4.2.7.2 Research Method

The case study used a simulated telecommunication network with fluctuating traffic loads and user mobility patterns. A DRL algorithm was employed to manage network resources dynamically, with the goal of minimizing service disruption and maximizing resource utilization. The system was evaluated based on performance metrics like latency, throughput, and service quality.

4.2.7.3 Main Results

The study found that DRL significantly improved network performance by enabling the system to adapt in real-time to changes in traffic and user behavior. The DRL model outperformed traditional optimization techniques, achieving lower latency and higher resource efficiency. The paper concludes that DRL can be a key enabler for telecommunication companies looking to build more responsive and adaptable networks.

4.2.8 Paper H: 5G Network on Wings: A Deep Reinforcement Learning Approach to the UAV-Based Integrated Access and Backhaul

4.2.8.1 Summary

This paper introduces a novel application of Deep Reinforcement Learning to UAV-based integrated access and backhaul (IAB) systems in 5G networks. The goal is to demonstrate how DRL can enhance the performance of UAVs in maintaining network coverage and optimizing backhaul capacity in dynamic and challenging environments.

4.2.8.2 Research Method

A DRL algorithm was implemented to allow UAVs to autonomously adjust their positions and resource allocations in a simulated 5G IAB network. The algorithm was tested in scenarios involving fluctuating user demands and varying levels of network congestion. Performance was measured in terms of network coverage, latency, and backhaul capacity.

4.2.8.3 Main Results

The results showed that DRL-enabled UAVs could effectively maintain network connectivity and optimize backhaul resources in real-time, even under challenging conditions. The study found that this approach not only improved network coverage but also reduced latency and enhanced overall service quality. The findings suggest that DRL can be a powerful tool for managing complex 5G networks in the future.

4.2.9 Paper I: Enabling Efficient and Low-Effort Decentralized Federated Learning with the EdgeFL Framework

4.2.9.1 Summary

This paper introduces the EdgeFL framework, a decentralized approach to Federated Learning designed to reduce complexity and improve efficiency in edge computing environments. The framework aims to minimize the technical overhead required to deploy FL while maintaining high model accuracy and scalability.

4.2.9.2 Research Method

EdgeFL was deployed in a distributed network of edge devices. The framework was tested in various edge computing scenarios, where multiple devices with limited computational resources participated in training a global model. The framework's performance was evaluated by measuring communication overhead, model accuracy, and ease of deployment in different edge environments.

4.2.9.3 Main Results

The study found that EdgeFL successfully reduced the complexity of deploying FL in decentralized environments. It achieved high model accuracy with minimal communication overhead, making it ideal for low-power edge devices. The results suggest that EdgeFL could enable wider adoption of FL in industrial and IoT applications by lowering the barriers to entry for businesses.

4.3 Summary

In this chapter, the primary contributions of the thesis are presented, structured around three key objectives. The first objective focuses on engineering Federated Learning systems that enhance data privacy, communication efficiency, and model accuracy in embedded environments. Through papers A, B, D, and E, the thesis demonstrates how FL can be effectively deployed in real-world applications by leveraging asynchronous model aggregation methods to improve system performance and privacy. The second objective, outlined in papers C and I, explores scalable architectural frameworks for FL, comparing different approaches and emphasizing the importance of decentralized architectures in improving scalability and fault tolerance in edge computing environments.

The third objective combines Federated Learning and Reinforcement Learning to improve the adaptability and performance of embedded systems in changing environments. Papers F, G, and H showcase how RL supports autonomous decision-making and real-time system adjustments in fields like telecommunications, while papers D and E highlight the use of FL in autonomous vehicles within the automotive industry. This thesis advances the integration of FL and RL, offering a framework for creating intelligent, efficient, and adaptive embedded systems that can respond to complex, dynamic scenarios.

CHAPTER 5

Engineering Federated Learning Systems: A Literature Review

This chapter has earlier been published as
Engineering Federated Learning Systems: A Literature Review
Zhang H., Bosch J. and Holmström Olsson H.
In *International Conference on Software Business* (pp. 210-218). Springer, Cham.

Nowadays, the development of mobile devices, connected vehicles, and data collection sensors has brought explosive growth of data, which highly power the traditional Machine Learning methods [98]. However, those common methods usually require centralized model training by storing data in a single machine or a central cloud data center, which leads to many problems such as data privacy, computation efficiency [99], etc.

Due to the development of computing and storage capabilities of distributed edge devices, using increased computing power on the edge becomes an applicable solution [100]. In a Federated Learning system, local model training is applied and data created by edge devices do not need to be exchanged. Instead, weight updates are sent to a central aggregation server to generate a global model. The system solves the problem that models in a traditional Machine Learning approach can only be trained and delivered on a single central server. The theory of Federated Learning has been explored in [101][102]. After the concept was first applied by Google in 2017 [103], there have been several Federated Learning architectures, frameworks and solutions proposed to solve real-world issues.

The contribution of this paper is threefold. First, we provide a state-of-art literature review within the area of Federated Learning systems. We identify and categorize existing literature into different application domains according to the problems expressed and solved. Based on the challenges and limitations identified in our literature review, we propose six open research questions for future research. This review can recommend a new option for industries and AI software engineer to solve the problems of traditional AI/ML systems, like expensive training equipment, computation efficiency, data privacy, etc. Furthermore, the difficulties are pointed out in this review when deploying the Federated Learning components into real systems.

This paper is structured as follows. In section 5.1, we describe the research method we applied. In section 5.2, we summarize the results from the literature review. In section 5.3, we outline the challenges of current Federated Learning systems. Finally, we conclude the paper in section 5.4.

5.1 Research Method

This research is conducted following the guidelines presented by Kitchenham [104]. The purpose of our review is to present an overview of contemporary research on the empirical results and solutions regarding Federated Learning that has been reported in the existing literature. In this paper, we address the following research questions:

- **RQ1.** What are the application domains where Federated Learning technique is applied?
- **RQ2.** What are the existing Federated Learning systems as reported in the published literature?
- **RQ3.** What are the main challenges and limitations identified in those reported systems?

5.1.1 Search Process

To provide a state-of-the-art literature review of Federated Learning in the software engineering research domain, we searched papers from several high-ranked journals/conferences. During our search process, in order to include all the papers which are related to our research questions, we started by selecting relevant terms, namely “Federated Learning”, “Distributed Learning”, “Collaborative Learning” to cover all papers which are related to Federated Learning and continued with “Case Study”, “Application”, “Solution” and “Framework” to identify papers that report on empirical study results.

The journals that were included in our search process are top-ranked software engineering and computer science journals such as IEEE Transactions on Software Engineering (TSE), Communications of the ACM (CACM), Machine Learning (JML), etc[105]. In addition, we used the same queries to search for relevant conference papers and literature in the well-known libraries, such as IEEE Xplore Digital Library, ACM Digital Library, Science Direct and Google Scholar.

5.1.2 Inclusion and exclusion criteria

Each paper that matched the search criteria was reviewed by at least one of the authors of this paper. During the selection, we firstly checked the keywords and the abstract to only include papers within Federated Learning field. After that, we searched and analyzed the application scenario in the body of the paper to identify the specific engineering problems solved by applying Federated Learning. We only selected the papers that report on Federated Learning with empirical results, e.g. Federated Learning on user action prediction, wireless systems, health records, etc. In summary, we included the paper where engineering Federated Learning systems are the main topic of the paper.

5.1.3 Results of the Literature Search Process

This section summarizes the results of our literature search process. Although there were about 253 different papers that initially matched the search criteria entered in the search engines of the journals and conferences listed in section 5.1.1, we found only 28 papers satisfying the inclusion criteria we specified. Those papers solve at least one engineering problem and present their empirical findings/results in the abstract or in the body of the paper. Based on problems addressed and solved in each paper, we categorize them into six application domains. In our search results, there are 4 papers ([106][107][108][109]) in telecommunication field, 6 papers ([103][110][111][112][113][114]) relates to mobile applications, 4 papers ([115][116][117][118]) relates to automotive, 5 papers ([119][120][121][122][123]) in IoT and 4 papers ([124][125][126][127]) relates to medical solutions. The rest of the papers ([128][129][130][131][132]) are related to other fields like air quality monitoring, image-based geolocation recognition, etc.

5.2 Existing Federated Learning Systems

In this section, and in accordance with the RQ2, we present the existing Federated Learning systems reported in papers we selected. In the rest of the section, in order to provide clear descriptions, we present each domain in more details.

5.2.0.1 Telecommunication

A typical telecommunication system usually contains numerous components and distributes to different places. In our results, most of the research focuses on constructing an efficient learning framework for federated model training. Wang et al. [106] define an “In-Edge AI” framework which enables intelligent collaboration between devices and the aggregation server to exchange learning parameters for better model training in energy and computation constraint user equipments. Kang et al. [108] introduce reputation metrics for reliable worker selection in mobile networks. The solution enhances system safety while keeping the same prediction accuracy. Yang et al. [109] propose a novel over-the-air computation based approach for fast global model aggregation via exploring the super-position property of the wireless multiple-access channel, which solves the problem of limited communication bandwidth in wireless systems for aggregating the locally computed updates.

5.2.0.2 Mobile Applications

Because of the explosive growth of smartphones and the evolution of the wireless network, a statistical Machine Learning model can significantly improve the mobile applications. However, due to the private data produced by personal-owned mobile devices, data privacy and security is also an essential topic in this domain. In order to apply Machine Learning techniques to human daily life, Yang et al. [103] and Ramaswamy et al. [110] apply Federated Learning techniques on the Google Keyboard platform to improve virtual keyboard search suggestion quality and emoji prediction. Leroy et al. [112] conduct an empirical study for the "Hey Snips" wake word spotting by applying Federated Learning techniques. Ammand et al. [113] implement a federated collaborative filter for personalized recommendation system. Liu et al. [114] propose “FedVision”, an online visual object detection platform, which is the first computer vision application applied Federated Learning technique.

5.2.0.3 Automotive

Automotive is a prospective domain for Federated Learning applications. Samarakoon et al. [115] suggest a distributed approach of joint transmit power and resource allocation which enables low-latency communication in vehicular networks. The proposed method can reduce waiting queue length without additional power consumption and similar model prediction performance compared to a centralized solution. Lu et al. [116] and Saputra et al. [117] evaluate the failure battery and energy demand for the electronic vehicle (EV) on top of Federated Learning. Their approaches show the effectiveness of privacy serving, latency reduction and security protection. Zeng et al. [118] propose a framework for combining Federated Learning algorithm within a UAV swarm. The framework proves that it can reduce the number of communication rounds needed for convergence compared to baseline approaches.

5.2.0.4 IoT

Internet of Things is a distributed platform which contains numerous remote sensors and devices. Different from the wireless system, devices within IoT are power-constrained. In our search results, most research in this domain focuses on data privacy and system efficiency problem. Zhou et al. [119] propose a real-time data processing architecture of the Federated Learning system on top of differential IoT. Zhao et al. [120] design an intelligent system which utilizes customer data to predict client requirements and consumer behaviour with Federated Learning techniques. However, the authors use the blockchain to replace the centralized aggregator in the traditional Federated Learning system in order to enhance security and system robustness. Mills et al. [122] design an advanced FedAvg algorithm which greatly reduces the number of rounds to model convergence in IoT network. Savazzi et al. [123] present a fully distributed or server-less learning approach in a massive IoT network. The proposed distributed learning approach is validated in an IoT scenario where a machine learning model is trained distributively to solve the problem of body detection. Sada et al. [121] give a distributed video analytic architecture based on Federated Learning. It allows real-time distributed object detection and privacy-preserving scheme for model updating.

5.2.0.5 Medical

Federated Learning has propelled to the forefront in investigations of this application domain. Vepakomma et al. [124] propose "splitNN" which enables local and central health entities to collaborate without sharing patient labels. Huang et al. [125][127] present an approach of improving the efficiency of Federated Learning on health records prediction. Brisimi et al. [126] give an approach to a binary supervised classification problem to predict hospitalizations for cardiac events on top of Federated Learning, which demonstrates faster convergence and less communication overhead compared to traditional machine learning approaches.

5.2.0.6 Other

In our research, we also identified some other application scenarios. Sozinov et al. [131] evaluate federated learning for training a human activity recognition classifier which can be applied to recognize human behaviour such as sitting, standing, etc. Sprague et al. [130] gives a groundwork for deploying large-scale federated learning as a tool to automatically learn, and continually update a machine learning model that encodes location. Verma et al. [129] provide strategies and results in building AI models using the concept of federated AI across multiple agencies. Hu et al. [132] propose an inference framework "Federated Region-Learning" to PM2.5 monitoring. The results demonstrate the computational efficiency compared to the centralized training method. Hao et al. [128] evaluate an efficient and privacy-enhanced Federated Learning scheme for industrial AI solution.

5.3 Discussion

In the previous section, we can observe that although Federated Learning is a newly-emerging concept, it has the potential to accelerate the Machine Learning process, utilize the advantage of distributed computing and preserve user privacy. However, there are several challenges and limitations associated with the techniques identified and described in the literature review. One of the biggest problems is the system failure tolerance, the majority of the Federated Learning systems presented in our reviewed paper apply a centralized architecture where edge devices are directly connected to a single central server and exchange model information. As [106] describe, this may make the system face the risk of single-point failure and influence the service availability of the learning system.

Furthermore, system efficiency is still a crucial problem for Federated Learning system. There are some proposed approaches to save computation power and communication resources for Federated Learning systems [109][122][128]. However, the conclusion needs to be further verified in real-world industrial deployments with the largely increased number of edge nodes. Besides, our review also identifies challenges of the methods to separate training devices, since systems reported in our reviewed papers usually utilize all the devices to participate training, which leads to the waste of the computation resources.

In addition, model validation has to be further improved. Especially for those safety-critical systems, such as automotive and medical applications [127][116][126], the quality of the models in all edge devices should be guaranteed.

Besides, due to the increasing number of edge devices, the mechanism of handling devices joining and leaving is one of the limitations in current Federated Learning systems. As [101] presents, the most common way is to simply accept new drop broken connections. This may lead to further problems of system performance such as model performance and model convergence.

Finally, although Federated Learning systems have the advantage of privacy-preserving, systems still have to face the risk of various security issues such as Denial-of-Service, malicious model updates, etc, which is also a major limitation and future direction for Federated Learning systems [120].

According to these challenges, we then propose six open questions for future research:

1. How to guarantee continuous model training and deployment in an industrial Federated Learning system?

2. How to efficiently update model weights and deploy global models?

3. How to split edge device sets for model training and testing?

4. How to guarantee model performance on all edge devices?

5. How to handle devices leaving and joining in different industrial scenarios?

6. How to protect Federated Learning systems from malicious attacks?

5.4 Conclusion

To stay competitive, more and more companies have introduced AI components into their products. However, although machine learning methods can improve software service quality, many companies struggle with how to minimize the system training cost and a reliable way to preserve user data privacy. Due to the model-only exchange and distributed learning features, Federated Learning is one option to solve those challenges. In order to provide concrete knowledge of this kind of learning approach to the industry, in this paper, we provide a literature review of the empirical results of Federated Learning systems presented in the existing literature. Our research reveals that there are several Federated Learning systems used for different application scenarios. Those scenarios are categorized into six different application domains: telecommunication, mobile applications, automotive, IoT, medical, other. Also, we note that the emerging trend of applying Federated Learning to mobile applications and identify several prospective domains. We summarize our findings in this article that works as a support for researchers and companies when selecting the appropriate technique. Furthermore, based on the challenges and limitations of current Federated Learning systems, six open research questions are presented.

In our future work, we plan to expand this review to include closely related, and highly relevant research papers. Also, we plan to validate our findings in the industry and explore the open research questions we propose in this paper.

CHAPTER 6

Towards Federated Learning: A Case Study in the Telecommunication Domain

This chapter has earlier been published as
Towards Federated Learning: A Case Study in the Telecommunication Domain
Zhang H., Dakkak, A., Mattos, D.I., Bosch J. and Holmström Olsson H.
In *International Conference on Software Business* (pp. 238-253). Springer, Cham.

Machine learning has steadily altered the way we live, learn, and work, with significant advances in speech, image, and text recognition, as well as language translation [98]. Large corporations like Google, Facebook, and Apple collect massive amounts of training data from users in order to build large-scale deep learning networks. However, while the utility of deep learning is clear, the training data it employs can have major privacy implications: images and videos of millions of people are collected centrally and stored indefinitely by major organizations, and individuals have little influence over how the data is used. Secondly, images and videos are likely to contain sensitive information such as faces, license plates, computer screens, and other people's conversations [133]. Large companies have a monopoly on ´´big data" and they could reap enormous economic gains as a result.

It is known that as the amount of training data increases, the diversity and performance of the models trained by machine learning will become better [134]. However, in many fields, the sharing of personal data is not allowed by regulations, such as GDPR [135]. Those regulations have put forward clear requirements for privacy provisions, further improving the protection of personal information. Therefore, researchers in related industries can only analyze and mine data sets belonging to their own organizations. If a single organization (e.g. a particular medical clinic) does not have a very large amount of data and includes insufficient diversity, then by performing machine learning on such a dataset, researchers may end up with a less generalized model. In this case, the limitations of data privacy and confidentiality clearly affect the effectiveness of machine learning.

On the other hand, with billions of edge devices connected worldwide, these devices are able to generate large amounts of data. In traditional cloud computing architectures, these data need to be centrally transferred to a cloud infrastructure for processing. The traditional method may increase the network load and cause transmission congestion and delays in the data processing. In order to solve those challenges, a new learning concept, Federated Learning, has emerged. Federated Learning refers to the provision of computing and storage services close to the source of things or data.

Although the concept of Federated Learning has significant benefits, it is sometimes hard for industries and companies to build reliable Federated Learning systems [136]. The contribution of this paper is threefold. We provide a case study in the context of a world-leading company with cutting edge technology and advanced practices. The study identifies the reasons why our case company considers Federated Learning as an applicable technique. Furthermore, based on our results, we summarize the services that a complete Federated Learning system needs to support in industrial scenarios and identify the challenges that industries are attempting to solve when adopting and transitioning to Federated Learning. Finally, we suggest 5 criteria for companies who want to implement reliable Federated Learning systems.

This paper is structured as follows. Section 6.1 presents the background of this study. In section 6.2, we describe the research method we applied as our basic principle when searching and collecting data. In section 6.3, we summarize the results from the interviews. In section 6.4, we outline the challenges and the criteria when realizing Federated Learning components into industrial systems. Finally, we conclude the paper in section 6.5.

6.1 Background

Due to the rapid development of the computation capability of edge devices, the integration of edge devices and machine learning has become more than a hypothesis. Due to its characteristics, Federated Learning is proposed to improve traditional Machine Learning approaches, as it enables edge devices to collaboratively learn a shared Machine Learning model. The theory of Federated Learning has been explored in [101][102]. Its major objective is to learn a global statistical model from numerous edge devices.

With the concept first applied by Google in 2017 [103], there have been several Federated Learning architectures, frameworks and solutions proposed to solve real-world applications. In a Federated Learning system, multiple devices work together in a collaborative manner to train predictive models. Federal learning can be built on edge devices (e.g., smart phones, video surveillance devices, etc.). Each edge node trains the machine learning model locally and independently, and the global model is optimized and merged by a central server (e.g., aggregation server). In the whole federation process, the privacy data does not leave the data owner and does not need to be shared with other nodes, which solves the problems of privacy and data security. In summary, the advantage of applying Federated Learning is conspicuous. Due to the mechanism of model training and data distribution, a Federated Learning system is a privacy-preserving Machine Learning approach. It is capable to utilize local computation resources, ease the computation pressure of the central server and provide rapid model evolution due to the local training fashion

Because of the local training fashion, it is capable of utilizing local compute resources, easing the computation load of central servers and providing rapid model evolution [137].

6.2 Research Methodology

The goals of this study were to explore the benefits for industries implementing Federated Learning and identify the issues that industries are attempting to solve when adopting and transitioning their machine learning components to Federated Learning. These goals were translated into the following research questions:
RQ1. What are the reasons that companies considers Federated Learning as an applicable technique?
RQ2. What kinds of services a Federated Learning system needs to support in the production environments?
RQ3. What are the main challenges and limitations when deploying Federated Learning components into embedded systems?

To answer these research questions, we designed the research in collaboration with Ericsson AB. This study built on a 6-month (Jan 2021 – July 2021) case study and applied a case study approach. [138]. In this paper, we chose a qualitative case study research approach since it allowed us to look at the current situation with a number of people from a given domain and understand a phenomenon in the industrial context in which it arises [139], [140]. In particular, case studies are considered appropriate for examining real-life contexts, such as software development and technique evolution, where controlling the context is not possible [141] and where there is a desire to access the interpretations and expectations of people so that a particular context can be richly understood [138]. Therefore, the high interdependence between the industrial context, the benefits of implementing Federated Learning (RQ1), required services (RQ2) and the faced challenges (RQ3) makes the case study a suitable choice.

6.2.1 Data Collection and Analysis

We collected data primarily through semi-structured interviews with practitioners who design or use machine learning applications, who involve heavily in data engineering, or who are otherwise machine learning experts [81]. The average interview length was around an hour. All of the interviews were recorded, transcribed, and shared via the case company's internal network.

Based on our interview protocol, we first asked participants to provide an overview of their domain and the specifics related to the telecom industry.

Then, we asked participants to provide their view regarding the machine learning projects they were currently involved in, the challenges they faced and the issues that the case company are attempting to solve when adopting and transitioning their machine learning components to Federated Learning. Finally, we asked the participants what they considered the key requirements for building a reliable Federated Learning system.

In addition to the interviews, we collected internal materials to support some of the interviews in addition to conducting interviews. These documents were either shared by participants or were available on the internal network as training, resources, or publications. The use of multiple data sources seeks to present a more comprehensive picture and improve the accuracy of this research [80].

The obtained data were processed using inductive thematic coding technique [142][140]. The authors acquainted themselves with the data by reading and transcribing the interviews in the first phase. During the interviews, at least two authors were present and took notes. After conducting all interviews, the contents were transcribed by the authors. In the second phase, the authors developed the initial set of codes by emphasizing significant observations in relation to the study's specified objectives. The initial set of codes were individually created by three of the authors and then combined later. The authors identified the primary themes in the third phase: machine learning applications, barriers for the industry to implement Federated Learning components and move toward Federated Learning. The authors reviewed these themes in connection to the retrieved codes and the entire data in the fourth phase. The authors defined and named the themes in the context of the appropriate material in the fifth phase. The final part entails the creation of the report, which includes the selection of data and quotes, reflection on current issues and the summary of methods that help companies step towards Federated Learning.

6.2.2 Case Company

In this research, we worked in close collaboration with Ericsson. Ericsson is one of the most well-known ICT (Information and Communication Technology) suppliers to service providers. They help clients get the most out of connectivity by creating game-changing technology and services that are easy to use, adapt, and scale in a fully connected world. Due to large-scale and distributed customers, Ericsson is also seeking a way to deliver reliable service quality to their customers. Since the company has board experience and tries to investigate the possibility of implementing Federated Learning, we chose it as our case company and try to identify the issues for companies to deploy Federated Learning into an industrial context.

6.2.3 Use Cases and Participants

During the research, we studied three different use cases within Ericsson. As we listed in Table 6.1, use case A refers to data collection and analysis. This field is critical for machine learning as well as Federated Learning since the quality and efficiency of the data collection procedure has a huge impact on final model performance [143]. Use case B refers to system architecture design and operation. Since Federated Learning is a distributed system, infrastructure design requires experience and careful consideration. Use case C refers to machine learning project design, development and operation, which is highly relevant to our topic and the experience from those practitioners is valuable for constructing a Federated Learning system. In total, we interviewed 10 participants in 9 interviews. Participants were gathered through industry contacts and were selected based on their relevance to the use cases involved in this study. All participants were experienced architects, senior developers, team leaders and development managers with at least eight years of experience, and most were also very experienced in data engineering and machine learning application development. To maintain confidentiality, we referred to the participants using labels P1 to P9, reflecting the interview numbers. As there are several participants in a single interview, we give the label suffixes, such as P7-1, P7-2. A summary of participants is listed in Table 6.1.

6.2.4 Threats to validity

The findings, like any case study, is primarily applicable to the domain and situations that were studied in this research [94]. The insights and results however, can be applicable and relevant also beyond the specific case at hand. Furthermore, while the empirical results are subjective because they represent the experiences of the chosen individuals, the possible scope of the results and their applicability is enlarged due to the extensive experience of the professional participants.

Table 6.1: Overview of the interviewees

Participant ID	Role	Use Case	Experience (Years)
P1	Global Data Domain Expert	A	30
P2	Data and Analytic Manager	A	25
P3	Analytic System Architect	B	13
P4	Analytic System Architect	B	14
P5	Data and Analytic Technical Driver	A	8
P6	New Products Operations Director	B	25
P7-1	Machine Learning Project Manager	C	30
P7-2	AI Systems Developer	C	18
P8	Customer data collection expert	A	28
P9	Head of Automation and AI	C	17

As for the construct validity, both the authors and participants in this case study have extensive machine learning, Federated Learning, and data engineering experience. Furthermore, if structures with special nomenclature within the industries or in academia were not understood, the authors with experience in both could translate and demonstrate them. As a result, the presence of dangers in the construct validity is not recognized.

In order to prevent threats to the validity of the conclusions, we took a number of steps during the study. The first stage was to eliminate bias created by individuals who had a similar point of view. To assist eliminate any personal prejudice, participants from various roles in different project teams were invited. The second stage was to gather data in the form of documentation to supplement the information gained through interviews, which also helped to avoid bias from being created by just collecting data in one format. Finally, direct participant comment on the developing data was obtained to assist validate the findings.

6.3 Empirical findings: Towards Federated Learning

6.3.1 Benefits of Implementing Federated Learning

As we observed in most of the companies, maintaining qualified service has become more and more expensive with the exponentially increased number of customers. One of the participants stated that their clients prefer to focus more on their own strengths while leaving service monitoring and maintenance to their device supplier.

> *"Customers want to focus on their strengths, such as marketing, bundling, and selling. Ericsson will track SLA (Service Level Agreement) to ensure that this network meets these KPIs and maintains SLA at these levels."* — Interview P6, New Products Operations Director

However, with the trend of this situation, the challenge appeared. In order to maintain and monitor a wide variety of equipment and traffic devices, companies may need to put more resources on products maintenance and troubleshooting, which turns out to be inefficient and inapplicable.

> *"It is not wise or feasible to have more people pumped in to monitor these types of systems as traffic density rises. As a result, AI assistance is required."* — Interview P2, Data and Analytics Manager

Our case company is a pioneer in the use of machine learning techniques in its products. As additional improvements in performance and network optimization are required for new demands from industrial applications, machine learning may help reduce complexity, meet new technologies and case requirements, improve network performance and allow for network automation.

> *"We use machine learning in every aspect of a telecommunication network you can think of, from the different use cases to the functions of creating a mobile network. "* — P7-1, Machine Learning System Project Manager

When it comes to Machine learning, data has become crucial to model performance and service quality. In general, the addition of large amounts of reliable data in industrial applications will significantly improve the learning quality and prediction accuracy of machine learning. As described by the participants:

> *"Customer issue: When it comes to machine learning, the right data is like food to humans."* — P7-1, Machine Learning System Project Manager

Nevertheless, the development of machine learning techniques also raises concerns about data privacy leaks when significant amounts of customer data are transferred. With the improvement of regulations and the importance of privacy protection, more constraints have been recognized:

> *"We also recognize the growing number of constraints on data movement, such as those imposed not only by data sovereignty concerns, correct? So, Norway, data stays in Norway, India said, our data stays here, and so on. Again, more constrained geographies."* — P5, Data and Analytic Technical Driver

In order to tackle those challenges, industries are trying to seeking a way to both avoid large data transmission but continuously provide stable service. One of our participants stated that with commonly applied learning strategies, such as centralized learning, it is almost impossible to make a quick response to large-scaled distributed customers when the characteristic of data has been dramatically changed.

> *"It's nearly impossible to respond quickly to large-scale customer changes without a Federated setting."* — P9, Head of Automation and AI

6.3.2 Transition to Federated Learning

Federated Learning can be a potential solution to those challenges due to its characteristics. Even though our participants agreed on the increasing interest in developing Federated Learning components into an industrial context, there are also issues that prevent companies adopt and transiting traditional learning strategies to Federated Learning.

> *"We need to think about how we can bring data out in a clever way, and I believe federated learning can help. Not only from radio base stations but also from the center."* — P8, Customer Data Collection Expert

One of the problems is a systematic distributed management system. Especially when industries are trying to collect data and monitor service performance from millions of network elements, it will become expensive and painful to discover the error. The situation may become more crucial if an additional function is added to edge devices, such as model training and validation.

> *"We don't have anything in place to quickly identify, for this one network element out of the million missed one file or wasn't available. It's extremely expensive because it requires people to manually check the edge to see what is going on."* — P3, Analytic System Architect

In addition, the system architecture is another important issue that has to be considered. Since in most of the scenarios, centralized data collection architecture is still the major data pipeline and support for current machine learning model development, as described by one of the participants, different levels of closeness to the source can be gradually applied when trying to transit current learning strategy to fully Federated Learning.

> *"The challenges I see are that there are different levels of closeness to the source, the different levels may result in different costs of management and transmission efficiency"* — P4, Analytic System Architect

The technique to guarantee model performance is another key issue. The models may require more capacity for generalization and automated learning iteration due to varied data features to accommodate rapidly changing customer environments and decrease the risk of poor service.

> *"It's critical for us to not only ensure model performance when it comes to the inference phase in Federated Learning but also to point out what's causing the degradation if any, especially if you've made certain data or network configuration changes in some nodes without informing the supplier."* — P9, Head of Automation and AI

Even though there are many other steps to be taken in order before a company transit to fully Federated Learning, our participants mentioned that it's a good time to consider now what kind of case studies it needs to be used and how these types of capabilities can be moved to a network.

> *"When introducing federated learning, you need to think large, but you probably also need to identify these small steps because these are the most valuable steps."* — P6, New Products Operations Director

There are three problems stated by one of our participants that we have to consider before moving towards and migrating Federated Learning components into embedded systems:

> *"What is the migration story there? How do you go about introducing this new capability? What type of use cases is suitable for Federated Learning?"* — P6, New Products Operations Director

6.4 Discussion

6.4.1 Benefits

From the empirical data, we identify that there are two major reasons which drive our case company to explore Federated Learning. One reason is the data privacy. As mentioned by our participants, Federated Learning may be one of the optimal solution to solve data silos and avoid privacy leakage. Since our case company has large amount of customers, the way of effectively integrating and analysing data that are scattered in various places is one of their biggest challenges. In the Federated Learning, as the data doesn't leave the edge and the analysts do not have direct access to the data, so the various data-related problems mentioned above are resolved. The value contained in the data can be exploited more effectively by the companies while still ensuring the data security of their customers.

Another reason is that companies may be able to respond to customers more quickly. Since Federated Learning can improve data collection and model training efficiency, the learning strategy can assist companies in implementing real-time functions to consume fresh customer data and adapt to environment changes, resulting in better service quality and enhanced model performance for their customers.

6.4.2 Learning Services

When conducting Federated Learning in actual production environment, we must consider not only the coupling and stability of the system, but also the business requirements with multiple data sources. Therefore, in order to cope with complex business requirements for each component of the system, we need to find a balance between flexibility and convenience. As mentioned by one of our participants:

> *"For Federated Learning, a complete training service should support functional features such as pre-checking, mid-term fault tolerance, full-cycle monitoring, and traceability of the results."* — P9, Head of Automation and AI

According to our empirical results, in Figure 6.1 we have summarized the services that a complete Federated Learning system needs to support in industrial scenarios.

Communication services: Since communication is required between the end customers, we must provide a gateway service to handle service routing and expose the API interfaces to the outside world in order to reveal as little information about our services to the other side as possible and to conveniently invoke training services. All queries from external systems will be routed through the gateway service for processing.

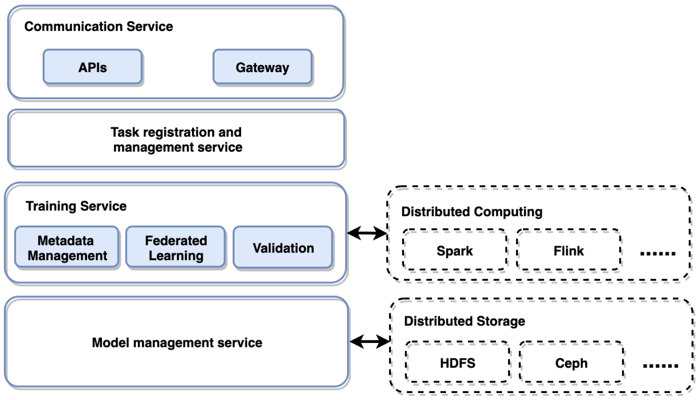


Figure 6.1: Services that a complete Federated Learning system needs to support

Task registration and management service: Task registration and management should be implemented to assure the service's high availability. The service information will be registered to the server when the service is started. When a training request is sent to the gateway, the gateway will retrieve the server's available computing resources and finish the service invocation using the defined load balancing policy.

Training Service: This service includes a metadata management component, Federated Learning component, and a validation component. The metadata management component will be in charge of keeping track of the progress of each training task, as well as the operating status and configuration parameters. In contrast, the Federated Learning component is used to conduct the numerous functions required during the distributed model training process. The validation component will be in charge of validating the configuration settings we submit as well as controlling model quality and performance.

Model management service: When the training task is finished, the training service provides the trained model information to the model management service, which then completes the distributed persistent storing, grouping, and other processes.

6.4.3 Challenges

Based on our empirical results, we derive the challenges for industries stepping towards Federated Learning. Figure 6.2 illustrates five challenges and problems which a typical system may encounter, including components failures, inefficient communication, unstable model performance, large-scaled end customers and incomplete system security.

6.4.3.1 Challenge 1 - Components failures

There are three main components in a typical Federated Learning system, including an aggregation server, communication links and remote edge devices. The architecture applied in current systems may often lead to significant bottlenecks and inevitable single-point failure. The problem can destroy system service stability and largely influence user experience. Furthermore, due to a large amount of communication, the link between servers and edge devices may be fully occupied or disconnected which results in unexpected information drop. Nevertheless, local edge devices may suffer problems of non-reachable server, program-stuck, high latency responses, etc.

> *"From the customer's perspective, nobody wants to hear what's going on in their network in terms of failures and crashes."* — P3, Analytic System Architect

Based on the characteristic of the Federated Learning technique, the problem of how to guarantee continuous federated model training and global model deployment to the edge is highly important to a service-sensitive industrial Federated Learning system. System robustness and fault tolerance issues are significantly more prevalent than in traditional distributed system environments.

6.4.3.2 Challenge 2 - Inefficient Communication

Federated Learning highly relies on a fast network and frequent communication of weight updates, either between peers or servers. However, the network situation for different devices may differ a lot. Since distributed edge devices need to frequently communicate to a central server in order to update model gradients and deploy fresh global models, the bottleneck and high bandwidth occupation at aggregation servers are inevitable issues. Imagining hundreds and thousands of edge devices needs to constantly keep the connection to

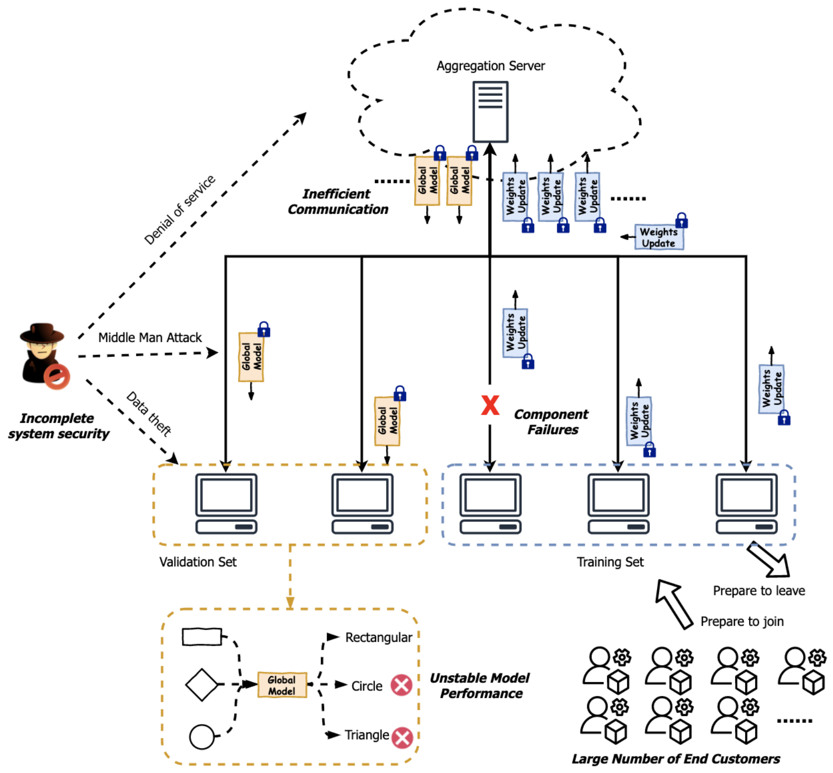


Figure 6.2: Problems and challenges for Federated Learning to be implemented into service-sensitive systems.

the servers, communication resources must be constrained due to frequent model updating and global model deployment. As mentioned by one of our participants:

> *"Efficiency is one of the key features. We want to reduce the cost such as bandwidth utilization but still be able to improve service quality"* — P2, Data and Analytic Manager

Although there are some of the research [101][144] related to communication-efficient Federated Learning systems, the problems of how to reduce the communication round in real industrial scenarios, how to efficiently utilize network resources while maintaining or even improving model prediction performance still need to be searched and verified.

6.4.3.3 Challenge 3 - Unstable Model Performance

For a traditional Machine Learning approach, the main goal of the system is to provide an accurate prediction or classification based on existing user data sets.

> *"Model performance is crucial for Federated Learning. If we cannot guarantee a sustainable model performance, we then have no reason to adopt it."* — P5, Data and Analytic Technical Driver

Similar to Federated Learning techniques, this challenge is the most critical one and also an important metric to evaluate a Federated Learning system. However, in the real world system, data collected from edge devices are non-IID and sometimes are unbalanced [136]. This is due to the different scenarios and environment edge devices exposed. The problem of how to keep or even improve model prediction performance compared to the traditional centralized model training approach, how to ensure the model can perform well on all the edge devices, what is the benchmark tool of evaluating Federated Learning systems are still tricky and need to be verified in different real-world application scenarios as we have described before.

6.4.3.4 Challenge 4 - Large Number of End Customers

As we described before, the Federated Learning system can be considered as Machine Learning clusters with a distributed configuration.

> *"We do have a huge number of customers. With Federated Learning, the way how to properly manage them and handle connections is a question."* — P4, Analytic System Architect

Normally, the system contains numerous edge nodes which may frequently leave and join. In order to achieve system scalability, the mechanism of how to handle device joining in, how to schedule device utilization and how to deal with device leaving without influencing system service still need to be researched and algorithms can be designed.

6.4.3.5 Challenge 5 - Incomplete system security

One of the main advantages of Federated Learning techniques is to prevent the transmission of sensitive user data. This is also the main reason why this technique has broad potential in various application domains. A privacy-preserving system is a big approach to Machine Learning system research.

> *"In the future, even with Federated Learning, We still have to explore a comprehensive approach in complying with applicable privacy regulations and legislation to handle the security and privacy aspects of our products."* — P6, New Product Operations Director

The question of how to avoid data leakage from global shared weights becomes essential. Therefore, a secured Federated System needs to protect not only local user data but also the transmission data from being damaged or leaked and forbidding illegal modification, access or usage of system programs, weight updates and global models. With the increasing attention and focus on AI-powered industrial solutions, security issues are more essential to the service provider.

6.4.4 Criteria for a Reliable Federated Learning System

Based on the issues mentioned by our participants that companies need to consider when implementing Federated Learning, we interpret those challenges to five criteria for a reliable Federated Learning system, including service availability for model training and improvements, efficient model training and sharing, accuracy assurance on the edge, scalable Federated Learning architecture and secured connection between edge and cloud. Those criteria are critical for effective and successful implementation of Federated Learning in embedded systems.

6.4.4.1 Service availability for model training and improvements

Availability means the durability of the system and likely to keep operating for a long period of time. As we described before, customers always need a reliable system service that guarantees continuous model training and deployment (Challenge 1), which is the foundation of model performance improvements. The problem is crucial in a Federated Learning system since most of the learning is real-time and fast-evolving. A reliable Federated Learning system needs to have a fault-dealing mechanism in order to guarantee service availability once a fault occurs. For example, the interaction between local edge devices and model aggregation server may suffer high latency, accidentally connection drop, server stuck due to high concurrence computation, etc.

6.4.4.2 Efficient model training and sharing

Efficiency signifies low resource utilization (CPU, Memory, Disk usage), low communication round and bandwidth occupation. This criterion relates to low-cost model training on the edge and efficient communication between edge and server during weight updating and model deploying procedure (Challenge 2). Furthermore, the way to split training devices is also an important approach to save computing resources. A reliable Federated Learning system should consume fewer resources, less bandwidth and communication utilization while still keeping an acceptable model performance.

6.4.4.3 Accuracy assurance on the edge

Accuracy is another important criterion. The system has to guarantee model performance and has the mechanisms to evaluate models on all edge devices (Challenge 3). Because the main purpose of the machine learning approach is to provide an accurate model for the corresponding application scenario, a reliable Federated Learning system should achieve, guarantee a satisfying model performance on all edge devices and be applied in the real-world industrial environment.

6.4.4.4 Scalable Federated Learning architecture

Scalability refers to the ability to handle an increasing number of tasks by joining more edge devices to the Federated Learning systems (Challenge 4).

Due to the highly distributed devices, a reliable Federated Learning system should be able to locate, find and accept asynchronous join of different types of edge devices and can be extended and cooperate with other learning clusters.

6.4.4.5 Secured connection between edge and cloud

Secured connection implies system data and model safety during local data collection, weights updating and global model aggregation. As described in Challenge 5, industrial systems may still need to face numerous kinds of malicious attacks. A reliable Federated Learning system should protect data collected at the edge devices, secure interaction between local edge devices and aggregation servers and guarantee the integrity of Machine Learning model transactions.

6.5 Conclusions

In this paper, we present a cutting-edge case study that identifies issues that industries are attempting to solve when dealing with Machine Learning cases, as well as the reasons why they anticipate Federated Learning as an applicable technique. Based on our findings, we summarize the services that a complete Federated Learning system needs to support in industrial scenarios. Furthermore, we highlight the issues that industries are attempting to address when adopting and transitioning their machine learning components to Federated Learning, including components failures, inefficient communication, unstable model performance, large-scaled end customers and incomplete system security. In addition, we suggest five critical criteria for designing and operating a dependable industrial Federated Learning system. In the future, we intend to validate our findings in industry cases and investigate solutions to the problems identified in this paper.

In summary, as we observed from the interviews from our case company, although the federated learning idea has considerable benefits, the creation of a trustworthy and relevant federated learning system is often problematic for them and may encounter different kinds of challenges. In this context, in the following section, we concluded the challenges and concerns that the industry is trying to resolve when adopting and migrating to federated learning.

CHAPTER 7

Federated Learning Systems: Architecture Alternatives

This chapter has earlier been published as
Federated learning systems: Architecture alternatives
Zhang H., Bosch J. and Holmström Olsson H.
In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 385-394). IEEE.

Federated learning is a new basic technology of artificial intelligence. It was originally proposed by Google in 2017, with the aim to solve the problems of local model training and updating in mobile edge devices [145][146][147]. The design goal of Federated Learning is to carry out efficient machine learning among multiple participants or computing nodes on the premise of ensuring the information security during massive data exchange, protecting the privacy of terminal data and personal data and ensuring legal compliance. Federated learning has the potential to be the foundation of the next generation of AI collaborative algorithms and networks [136].

Federated learning defines a machine learning framework, in which a global model is designed to solve the problem of collaboration between multiple data owners without exchanging data [145]. The global model is the optimal model which is the aggregated knowledge from all parties. Federated learning requires that the modelling result should be infinitely close to the traditional pattern, that is, the data belonging to multiple owners should be gathered in one place for modelling results [148]. Since the data is not exchanged, it will not take the risk of leaking the user's privacy or affecting the data specification which meets the requirements of legal compliance (such as GDPR [149]). Figure 1 shows the system architecture of Federated learning with two data owners (edge A and edge B) as an example. The system can be extended to scenarios with multiple edge data owners. Suppose that edge A and B want to train a machine learning model jointly, and their business systems have the relevant data of their respective users. If A and B are both allowed to exchange data directly, for example, because of the data privacy and security issues, we may apply the Federated Learning system to build the model.

However, our research shows the challenges of deploying Federated Learning into a real-world industrial context. As defined in *"Engineering AI Systems: A Research Agenda"* [150], AI engineering refers to AI/ML-driven software development and deployment in production contexts. Also, our previous research shows that **the transition from prototype to the production-quality deployment of ML models proves to be challenging for many companies** [99][151]. The situation also applies to Federated Learning systems [45]. Currently, the majority of deployments utilize a single-server centralized architecture which may inevitably face the risk of component failure, system scalability, communication efficiency, etc [136]. Those problems will prevent the AI/ML components from being continuously serviceable in real-world industrial deployments, which can compromise the system and lead to terrible accidents in the end.

To the best of our knowledge, there is limited research that provides an overview of the different architecture alternatives for the Federated Learning systems. In this paper, based on our simulation, we describe and suggest several applicable scenarios and use cases for four different architecture reported in this paper which can be applied to an industrial Federated Learning system. We conduct the study using two well-known image classification data sets, MNIST and CIFAR-10. All the training data are distributed to edge devices that follow a statistical distribution to simulate real-world scenarios. In order to provide comprehensive suggestions, for each alternative, communication latency, model evolution time and model classification performance

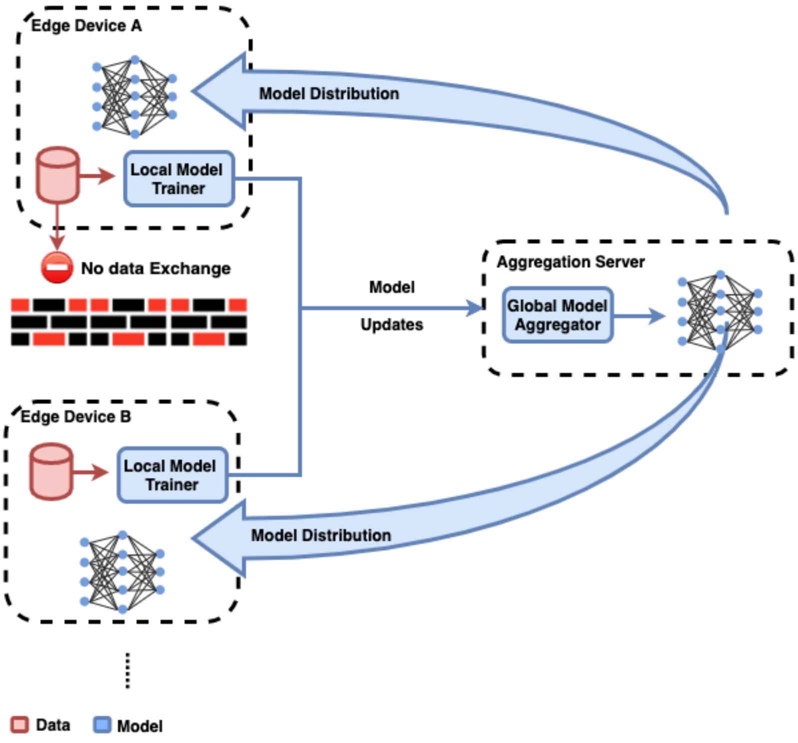


Figure 7.1: System architecture of Federated learning

are measured and compared.

The contribution of this paper is threefold. First, we introduce four architecture alternatives which have been or can be applied to a Federated Learning system and we identify the advantages and disadvantages of each alternative. Second, we evaluate the system performance, including weights update latency, model evolution speed and model classification performance with each of the architecture alternatives. Third, by studying the trade-off between model performance and the overhead of latency and evolution speed, we describe for which industrial scenario each architectural alternative reported in this paper is the optimal choice.

The remainder of this paper is structured as follows. Section II introduces four architecture alternatives. Section III details our research method, including the simulation testbed, the method of distributing the training data set, the utilized machine learning method and the evaluation metrics. Section IV presents the algorithms utilized in each alternative. Sections V evaluates four architecture alternatives applied to the data traces. Section VI outlines the discussion on suitable scenarios and use cases for each alternative. Finally, Section VII presents conclusions and future work.

7.1 Architecture Alternatives

As described in Section I, current Federated Learning systems may face the problem of components failure, system scalability, communication efficiency, etc. Inspired from the empirical results of existing literature [136][145][132][120], we have defined four alternatives which can be utilized in a Federated Learning system from a centralized to a fully decentralized approach, that is, centralized, hierarchical, regional and decentralized architectures. The terms of each architecture are derived based on their characteristic. Figure 7.2 illustrates the concepts.

7.1.1 Centralized Architecture

The centralized architecture is a widely used setting in the majority of current Federated Learning systems[146][126][116]. In this alternative, there is only a single central node which is responsible for communicating all edge devices, aggregating local models, and deploying the global model. The model trans-

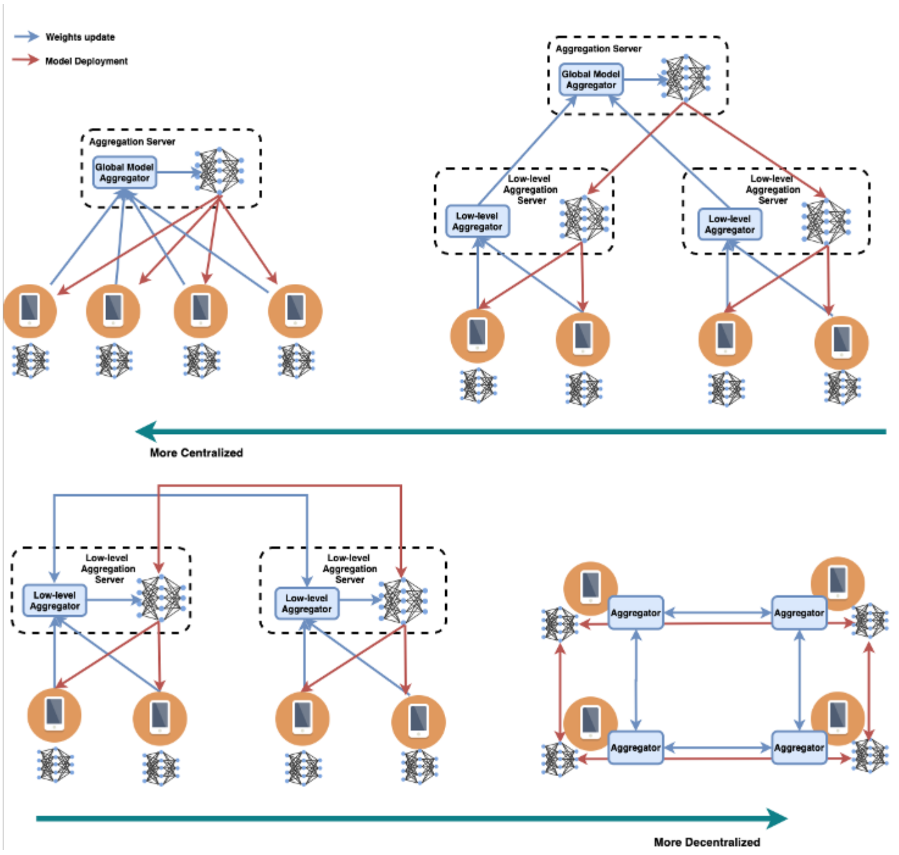


Figure 7.2: Architecture alternatives for Federated Learning systems: centralized, hierarchical, regional and decentralized architecture. For a centralized Federated Learning system ((a)), all the edge nodes are connected to the central aggregation node in order to update local weights and distribute models. An improved way ((b)) is to add several coordinators, the regional aggregation nodes, which aims to reduce data exchange and be in charge of managing local devices. The regional architecture ((c)) will totally remove the central management point in order to remove the risk of the single-point of the failure. A more elegant way ((d)) is to completely move the aggregation function to the edge. Each edge node can perform local training and model aggregation. This is a potential alternative when a global or regional sever faces the problem of heavy traffic and then becoming a bottleneck.

mission within this architecture is smooth and elegant and the single central node has a dedicated system which can be modified to suit customized needs. Quick updates become possible and it is efficient for small systems, as the central systems take limited resources to set up. In addition, any edge node can be easily detached from the system by removing the connection between the client node and the server without influencing other active nodes.

However, because of the single management node, the centralized Federated Learning system will encounter a scalability problem. When thousands of client nodes join, the server node will not have improved performance even if the hardware and software capabilities have been optimized. In addition, communication bottlenecks may appear when the amount of traffic increases exponentially and the system can easily break down when the server suffers a Denial-of-Service attack.

7.1.2 Hierarchical Architecture

As shown in Figure 7.2 (b), different from the centralized architecture, a hierarchical architecture introduces several regional coordination nodes to manage different edge clusters, which can ease the work of the central node, such as model updating and aggregation. This alternative has been introduced in [146], which solves part of the communication bottleneck problem and is scalable for a medium system. However, this approach still has the potential problem of the single-point of failure and being vulnerable to DoS attack since the central node still exists. In addition, the management cost will increase and the industrial deployments may need more budgets for more aggregation servers compared to a centralized architecture alternative.

7.1.3 Regional Architecture

The regional architecture has a similar setting compared to the hierarchical architecture but removes the central aggregation nodes. Each edge cluster will be assigned to a regional aggregation node where models are aggregated and exchanged. One application which utilizes this alternative is reported in [132]. The results demonstrate the computational efficiency compared to a more centralized architecture. The purpose of this design is to avoid the influence of the central node failure and to increase system robustness. In addition, after defining the frequency of local model exchange among regional aggregation nodes, a system may have a chance to focus more on their local sample clusters instead of the whole data set at the edge. However, with the increasing number of servers, real-world deployments may cost more in terms of hardware purchases and server configuration management.

7.1.4 Decentralized Architecture

As shown in Figure 7.2 (d), a decentralized Federated Learning system only contains edges nodes. Compared to the three alternatives above, a decentralized architecture moves the aggregation function to the edge. The idea is firstly tried and reported in [152]. The system is able to minimize the problem of performance bottlenecks since the entire load gets balanced on all the nodes. Furthermore, due to the flexibility of node connections, the system has better autonomy and is able to quickly adapt its local environment changes.

Nevertheless, decentralized architecture can lead to the problem of coordination. Since every node is the owner of its own behaviour, it is difficult to achieve collective tasks and global knowledge. Normally, the models vary a lot which is not optimal for some scenarios. Additionally, it is not suitable for small systems since industries cannot benefit from building and operating small decentralized systems due to inefficient system management and performance.

7.2 Research Method

In this research, the empirical method and learning procedure described in Zhang [92] was applied to make a quantitative measurement and comparison with four architecture alternatives. In the following sections, we present our simulation testbed, the method used for splitting and distributing data sets, evaluation metrics and the machine learning methods used in the experiments.

7.2.1 Simulation Testbed

Figure 7.3 outlines our testbed topology. In order to simulate aggregation and edge functions, we adopted two of the total six machines as our server cluster and the rest work as the edge. (Table 7.1 shows the hardware setup for all the servers) Each edge nodes were implemented as a small process running in one of the edge nodes server cluster (server 3-6).

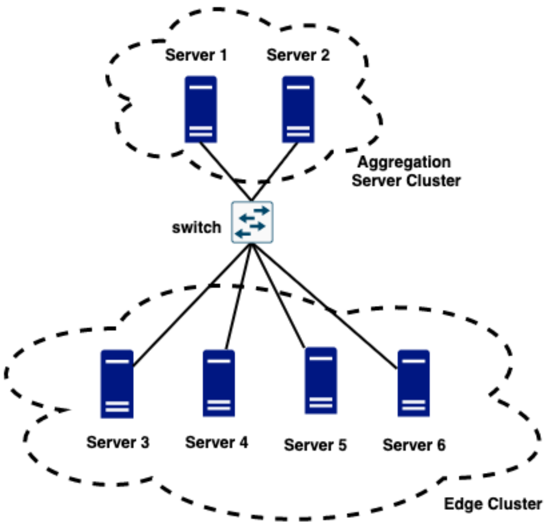


Figure 7.3: Topology of the simulation testbed

Table 7.1: Hardware setup for testbed server

CPU	Intel Xeon Processor (Skylake, IBRS)
Cores	8
Frequency	2.59 GHz
Memory	32 GB
OS	Linux 4.15.0-106-generic

More specifically, in the centralized architecture simulation, aggregation functions were deployed on server 1 while the edge nodes in server 3-6 can push and request the latest model to or from server 1 to continuously learn latent patterns.

In the hierarchical architecture, the central aggregation function was deployed in server 1 while we assigned four regional aggregation processes in server 1 and 2. Edge nodes in each edge server were assigned to one of the regional aggregation processes, which means that those nodes will only contact their corresponding regional aggregation process.

For the regional architecture simulations, the aggregation functions were deployed both on server 1 and 2. Similar to the hierarchical architectures simulation, edge nodes were assigned to one of the aggregation processes once they joined in the system and only communicated with that unique aggregation node. In the decentralized simulation, we removed the aggregation server cluster and moved the aggregation functions to all the edge nodes in order to simulate decentralized features. In each edge nodes, their neighbour nodes were predefined based on their edge ID.

7.2.2 Training Data Distribution

For the purpose of this study, we used two kinds of the edge data distribution to analyze system performance under different architecture alternatives.

7.2.2.1 Uniform Distribution

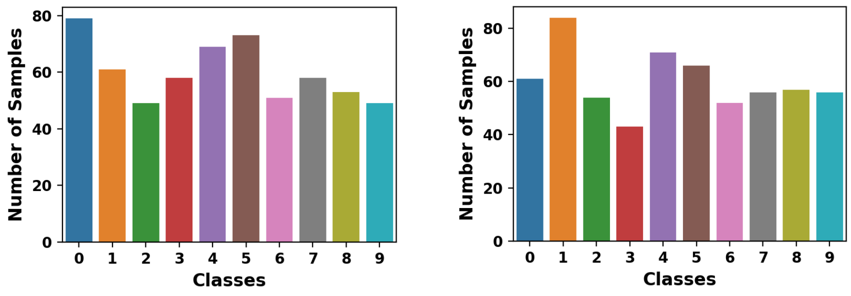
Under this setting, we distributed training data samples to the edge follows the uniform distribution, which means the number of data samples of each target classes is equally likely. Figure 7.4 outlines the data distribution in two example edge nodes.

7.2.2.2 Normal Distribution

With this setting, in each edge nodes, the number of samples in each class follows the normal density function as shown below.

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Here, μ and σ are defined as follows:

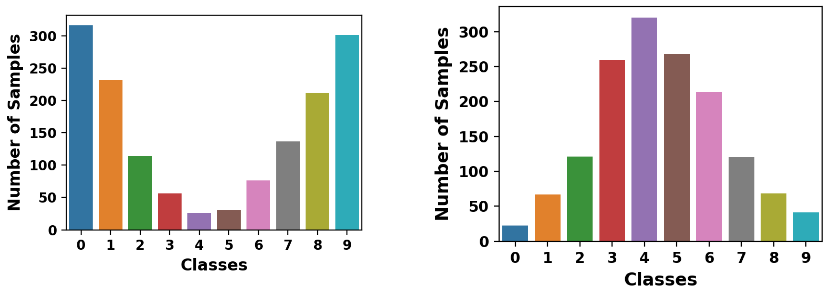


(a) Edge 0 (b) Edge 49

Figure 7.4: Uniform training data distribution

$$\mu = \frac{k \times N}{K},\ \sigma = 0.2 \times N$$

where k is the ID of each edge node, K is the total number of edge nodes and N equals to the total number of target classes in training data. The purpose of this configuration is to provide various distribution in different edge nodes, where each class can have the probability to have the majority number of samples in one node. Figure 7.5 outlines the data distribution in two example edge nodes.



(a) Edge 0 (b) Edge 49

Figure 7.5: Normal training data distribution

7.2.3 Machine Learning Method

The models used in this paper were implemented in Python, using torch 1.4.0 [153], torchvision 0.5.0 [154] and scikit-learn [155] libraries for model building.

In order to achieve a satisfying classification result, two different convolutional neural networks (CNN) [156] were trained for the MNIST and CIFAR-10 data sets. In the MNIST data set experiment, the CNN network contains two 5x5 convolution layers, (The first layer has 10 output channels, while the second has 20, each followed with 2x2 max pooling.) a fully connected layer with 50 units and the ReLu activation, and a linear output layer.

For the CIFAR-10 data set, the CNN network contains four 5x5 convolution layers, (The first layer has 66 output channels; the second has 128 output channels and the stride of convolution equals 2; the third has 192 channels; the fourth has 256 channels and the stride of convolution equals 2.), two fully connected layers (ReLu activation) with 3000 and 1500 units, and a linear output layer.

7.2.4 Evaluation Metrics

In order to demonstrate fruitful results of systems under different architecture alternatives, we selected three metrics including weights update latency, model evolution time and model classification performance (local and global).

7.2.4.1 Weights update latency

The weights updated latency is defined as the time difference of the model transmission from edge nodes to the aggregation nodes (In the centralized, hierarchical, regional architecture, aggregation nodes are central or regional servers which are responsible for collecting models. In the decentralized architecture, since aggregation function is moved to the edge, the aggregation node can be regarded as the peer node which is ready for receiving the updated model). The result is the average of all edge nodes during one training round. This metric indicates the network situation and communication overhead of each architecture alternatives. The metrics were measured in all the model receivers by checking the sending and receiving timestamp.

7.2.4.2 Model Evolution time

Evolution time is defined as the time difference between two different versions of the deployed global model at the edge nodes. The result is the average of all edge nodes during one training round. This metric demonstrates the speed of local edge devices updating their knowledge which is crucial and important for those systems which need to quickly evolve to adapt to the rapidly-changed environment. The metrics were measured in all the edge nodes by checking model deployment timestamp.

7.2.4.3 Model Classification Performance

Classification performance is the most important metric which indicates the quality of the training model. It is defined as the percentage of correctly recognized images among the total number of testing images. Furthermore, in order to have a better understanding of the influence of different architectures on local edge devices. Here, the *local classification performance* was tested on each edge devices by using their updated global model. The test sample distribution should be the same as the training samples (local test set). The result of local classification performance is the average value from all edge nodes. The *global classification performance* is tested by using the global test set, where the number of samples in different classes should be equally likely.

7.3 Algorithms used in each architecture alternative

In order to simulate and compare characteristics of the system with the architecture alternatives reported in this paper, we select Federated Averaging (FedAvg) [145] as the base Federated Learning algorithm during our experiments. This algorithm has been widely used in research and industrial communities for model aggregation. Thus, it is also compelling to see how FedAvg behaves with the architecture alternatives introduced in section 7.1. In a centralized architecture, the original Federated Average algorithm is applied while for the other three alternatives, the base algorithm is modified to fit different architectures.

Algorithm 1: FedAvg - Centralized: In the system, total K edge devices are indexed by k; B is the local mini-batch size; E represents the number of local epochs, and γ is the learning rate.

```
Function Server_Function():
    initialize $w_0$
    for each round $t$ = 1, 2, ... do
        $m \longleftarrow max(C \times K, 1)$;
        $S_t \longleftarrow$(random set of $m$ clients);
        for each client $k \in S_t$ in parallel do
            $w_{t+1}^{k} \longleftarrow Client\_Update(w_t)$;
        end
        $w_{t+1} \longleftarrow \sum_{k=1}^{K} \frac{1}{K} w_{t+1}^{k}$;
    end
End Function
Function Client_Update($w$):
    $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
    for each local epoch $i$ from 1 to $E$ do
        for batch $b \in \beta$ do
            $w \longleftarrow w - \gamma \nabla l(w; b)$;
        end
    end
    return $w$ to server
End Function
```

7.3.1 Centralized Architecture

The algorithm used in the centralized architecture is outlined in Algorithm 1. Since this architecture has been widely used in various fields, we didn't change any components and make the setting remain the same as all existing research. The steps of FedAvg algorithm in the centralized architecture is straight-forward:

Step 1: Edge devices locally compute the model; After reaching the number of local epochs, they send updated model results w to the central aggregation node.

Step 2: The central node performs aggregation by averaging all updated models to form a global knowledge of w_{t+1}.

Step 3: The node sends back the aggregated result to each edge device k.

Step 4: Edge device replaces the local model and performs further local training by using the global deployed model.

7.3.2 Hierarchical Architecture

The algorithm (Algorithm 2) used in this alternative is modified based on the Federated Averaging algorithm. Since the system has several regional coordination nodes, all the edge nodes send their weights updates only to their corresponding regional nodes. After receiving local models, a regional coordination node sums all models and counts the number of received models. Then, these information will then be updated to the central node. Therefore, the central node only needs to process the information sent from coordinator nodes without contacting numerous edge devices, which largely releases and balances the computation work at the central point. The steps can be summarized as follows:

Step 1: Edge devices locally compute the models; After reaching the number of local epochs, they send updated model results w to the regional aggregation nodes.

Step 2: The regional nodes perform aggregation by adding all updated models and calculate the number of updated models. Then, these information will be sent to the central node to form a global knowledge of w_{t+1}.

Algorithm 2: FedAvg - Hierarchical

```
Function Server_Function():
    initialize $w_0$
    for each round $t$ = 1, 2, ... do
        $S_l \longleftarrow (localserverset)$
        for each local server $s \in S_l$ in parallel do
            $w^s_{t+1}, k^s \longleftarrow Local_Server\_Update(w_t)$;
        end
        $K_{t+1} \longleftarrow \sum_{s=1}^{S} k^s$
        $w_{t+1} \longleftarrow \sum_{s=1}^{S} \frac{1}{K_{t+1}} w^k_{t+1}$;
    end
End Function
Function Local_Server_Update($w_t$):
    for each round $t$ = 1, 2, ... do
        $m \longleftarrow max(C \times K, 1)$;
        $S_t \longleftarrow$(random set of $m$ clients);
        for each client $k \in S_t$ in parallel do
            $w^k_{t+1} \longleftarrow Client\_Update(k, w_t)$;
        end
        $w_{t+1} \longleftarrow \sum_{k=1}^{K} w^k_{t+1}$;
    end
    return $w_{t+1}$, $len(S_t)$ to central server
End Function
Function Client_Update($k$, $w$):
    $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
    for each local epoch $i$ from 1 to $E$ do
        for batch $b \in \beta$ do
            $w \longleftarrow w - \gamma \nabla l(w; b)$;
        end
    end
    return $w$ to local server
End Function
```

Step 3: The central node sends back the aggregated result to each regional nodes. Regional nodes will then forward the global model to all registered edge devices k.

Step 4: Edge device replaces the local model and performs further local training by using the global deployed model.

7.3.3 Regional Architecture

In order to remove the central node and move the aggregation functions to the regional nodes, we further modified the algorithm used in hierarchical architecture. In each training epochs, regional nodes are only responsible for aggregating models for their registered edge devices. After a certain number of training iterations, all regional nodes exchange their model information with each other to form a global knowledge. The algorithm is outlined in Algorithm 3 and the steps can be summarized as follows:

Step 1: Edge devices locally compute the models; After reaching the number of local epochs, they send updated model results w to corresponding regional aggregation nodes.

Step 2: The regional nodes perform aggregation by averaging all updated models to form regional knowledge. In addition, every f iterations, there is an exchanging iteration in which the node applies another aggregation function by adding all updated models and calculate the number of updated models. Then, this information will be spread to all the regional nodes to form a global knowledge of w_{t+1}. (If the exchanging iteration is not reached, regional nodes will only aggregate a regional model and send it to all the edge nodes)

Step 3: After calculating the aggregated result in each regional nodes. Regional nodes will then forward the global model to all registered edge devices k.

Step 4: Edge device replaces the local model and performs further local training by using the global deployed model.

Algorithm 3: FedAvg - Regional: The new parameter f defined the frequency of exchanging the models.

```
Function Server_Update($w_t$):
    initialize $w_0$ $S \longleftarrow$ (all neighbour servers)
    for each round $t$ = 1, 2, ... do
        $K \longleftarrow 0$;
        $m \longleftarrow max(C \times K, 1)$;
        $S_t \longleftarrow$(random set of $m$ clients);
        for each client $k \in S_t$ in parallel do
            $w^k_{t+1} \longleftarrow Client\_Update(k, w_t)$;
            K ++;
        end
        $w_{t+1} \longleftarrow \sum_{k=1}^{K} w^k_{t+1}$;
        if $t$ mod $f$ == 0 then
            for each server $s \in S$ in parallel do
                send($W_{t+1}, K$);
                $w^s_{t+1}, k^s \longleftarrow Server^s\_send(w_t)$;
                $K_{t+1} \longleftarrow \sum_{s=1}^{S} k^s$
            end
            $w_{t+1} \longleftarrow \sum_{s=1}^{S} \frac{1}{K_{t+1}} w^s_{t+1}$;
        else
            $w_{t+1} \longleftarrow \frac{1}{K} w_{t+1}$;
        end
    end
End Function
Function Client_Update($k$, $w$):
    $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
    for each local epoch $i$ from 1 to $E$ do
        for batch $b \in \beta$ do
            $w \longleftarrow w - \gamma \nabla l(w; b)$;
        end
    end
    return $w$ to regional server
End Function
```

7.3.4 Decentralized Architecture

In order to realize decentralized characteristics, we remove the aggregation functions from central points but attach them to the edge. Algorithm 4 illustrates the idea. Each edge nodes has an independent process to train, send and receive model weights. There is also a frequency parameter which can control edge nodes exchange their model to their neighbours after several training epochs. The steps can be concluded as follows:

Step 1: Edge devices locally compute training gradients; After reaching the exchanging iteration, they send updated model results w to their registered neighbours.

Step 2: After receiving all the models from the neighbours, each node performs aggregation by averaging all updated models.

Step 3: Edge device replaces the old model and performs further local training by using the updated model.

Algorithm 4: FedAvg - Decentralized

```
Function Client_Update($k$, $w$):
    $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
    $C \longleftarrow$ (all neighbour clients)
    for each round $t$ = 1, 2, ... do
        for batch $b \in \beta$ do
            $w_{t+1} \longleftarrow w_t - \gamma \nabla l(w_t; b)$;
        end
        if $t$ mod $f$ == 0 then
            for each client $c \in C$ in parallel do
                send($W_{t+1}$);
                $w^c_{t+1} \longleftarrow client^c\_send(w_t)$;
            end
            $w_{t+1} \longleftarrow \sum_{c=1}^{C} \frac{1}{len(C)} w^c_{t+1}$;
        end
    end
End Function
```

7.4 Evaluation

In this section, we present the experiment results for four different architecture alternatives and compare them with the system performance in three aspects (The metrics are defined in section 7.2.4) - (1) Weights update latency: time used to transmit model from edge to the aggregation nodes, (2) Model evolution time: time used to train and deploy a new global model, (3) Local and Global model classification accuracy: classification accuracy tested on the local and global test set.

To have a clear comparison, the MNIST data set was used to measure all three metrics while CIFAR-10 data set was used to further validate the result of local and global classification accuracy. During the experiments, we conduct the simulation with the different number of edge nodes which varies from 10 to 1,000 and all the nodes participate training procedure.

7.4.1 Weights update latency

Figure 7.6 present the result of weights updating latency, which illustrates a linear increasing trend of weights latency based on the number of connected nodes and more detailed values are listed in Table 7.2.

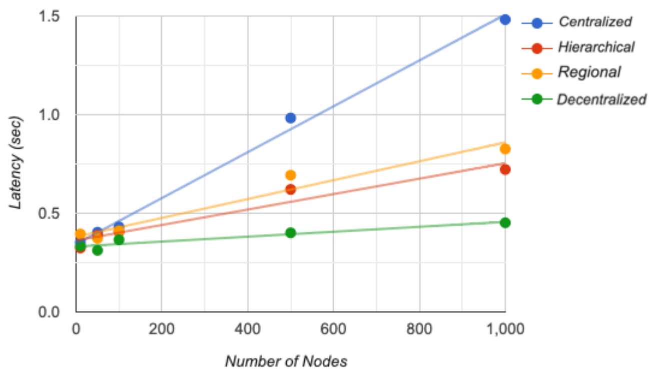


Figure 7.6: Weights latency with different number of nodes in four architecture alternatives

The above figure shows that centralized architecture has the largest weights update latency when the number of nodes is bigger than 500. In a centralized architecture, a single central node needs to handle all receiving, training and sending tasks which may highly influence system performance. It can easily lead to communication bottleneck and single-point failure.

Relatively, in the hierarchical and regional alternatives, after introducing regional nodes, the load on the central node is balanced by multiple regional nodes. The red and orange lines show a linearly increasing trend but slower than the centralized architecture.

Furthermore, in the decentralized architecture, since each node can establish equal connections, the server work is further distributed to the edge. Since nodes can only communicate with their neighbours, every node can balance the weights updating traffic, which leads to the smallest growth rate among four architecture alternatives.

Table 7.2: Weights updating latency

	Latency (sec)			
Number of Nodes	Central	Hierarchical	Regional	Decentral
10	0.353	0.324	0.395	0.334
50	0.404	0.389	0.373	0.312
100	0.431	0.406	0.411	0.366
500	0.983	0.621	0.693	0.401
1000	1.482	0.722	0.826	0.452

In addition to weights updating latency, the number of retransmission is also measured. From Table 7.3, it can be observed that, when dealing with a large number of edge devices, centralized architecture causes more transmission mistake and less communication efficiency than other alternatives. It also proves our findings in weights updating latency.

7.4.2 Model Evolution Time

We then calculated the average model evolution time in all edge nodes, which is outlined in Table 7.4. In our experiments, the model evolution time is influenced by model training time and the weights update latency. With the increasing number of nodes, the training time in each training epoch largely decreases, due to the distribution of training data in each edge nodes and

Table 7.3: Average number of model retransmission during one training iteration

Number of Nodes	Central	Hierarchical	Regional	Decentral
10	-	-	-	-
50	-	-	-	-
100	-	-	-	-
500	6	-	-	-
1000	147	17	21	-

the model training task is separated in numerous workers. However, with the increasing number of nodes, latency may increase as well. In our results, the best number of nodes in the previous three alternatives is 500 while evolution time further increases with the growth of the number of edge nodes.

Table 7.4: Average model evolution time

	Model evolve (sec)			
Number of Nodes	Central	Hierarchical	Regional	Decentral
10	44.218	45.036	46.020	45.017
50	10.052	10.910	12.741	10.311
100	4.839	4.657	4.166	4.327
500	2.584	2.183	2.031	2.049
1000	3.602	2.990	3.016	1.553

7.4.3 Classification Accuracy

In this section, we present model classification accuracy under two different training sample distributions. Here we only present the result measured with 100 edge nodes as the number of edge nodes doesn't have too much obvious influence on classification accuracy.

Table 7.5: Global Prediction performance with MNIST data set follows a uniform data distribution on the edge

	MNIST Global			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	96.63	96.42	96.01	94.91
30	98.10	97.80	97.66	96.87
50	98.55	98.47	98.39	97.08

7.4.3.1 Uniform Distribution

As described in section 13.3.3, the number of classes in each edge device with this distribution are equally likely. Under this setting, the global model classification accuracy (with global test set) can reach 98% in MNIST data set and 88% in the CIFAR-10 data set. The results are outlined in Table 7.5 for MNIST and Table 7.6 for CIFAR-10.

Table 7.6: Global Prediction performance with CIFAR-10 data set follows a uniform data distribution on the edge

	CIFAR-10 Global			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	78.75	78.42	77.33	75.89
30	83.21	81.94	81.24	80.30
50	87.92	88.01	87.37	86.45

However, we see a slight difference in four alternatives where the regional and decentralized architecture has 1% worse accuracy, which we explain that in a more decentral architecture, a model may cost more time to form the global knowledge due to their algorithm. (Especially in the decentralized architecture, the model needs more training rounds to spread and aggregate.) This feature becomes more obvious while the model is trained on the data which is distributed and follows a normal density function.

Table 7.7: Global Prediction performance with MNIST data set follows a normal data distribution on the edge

	MNIST Global			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	89.05	88.95	68.72	33.69
30	95.96	94.16	86.22	45.93
50	97.12	96.31	93.70	81.39

Table 7.8: Local Prediction performance with MNIST data set follows a normal data distribution on the edge

	MNIST Local			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	89.51	89.42	92.70	95.84
30	95.48	95.05	93.51	96.91
50	97.07	96.00	95.29	98.02

7.4.3.2 Normal Distribution

Normal sample distribution is closer to a real-world data set, however, the accuracy of image classification results is worse than the model which is trained on the data set with a uniform distribution. We observe 1% lower accuracy with MNIST global test set under the centralized architecture alternative. Furthermore, in the decentralized architecture, a model needs more time to converge and form a global knowledge on the whole training data set. The results are presented in Table 7.7.

Table 7.9: Global Prediction performance with CIFAR-10 data set follows a normal data distribution on the edge

	CIFAR-10 Global			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	72.14	71.02	63.44	25.51
30	78.74	76.93	71.24	44.34
50	86.82	86.03	80.68	70.65

However, when it comes to model performance on the local test set, we find that the decentralized architecture outperforms the rest of the architectures. Compared to architectures with the central aggregation server, the decentralized architecture focuses more on a local data set which results in a slower process of forming the global model but achieves higher accuracy on local set classification. The results are outlined in Table 7.8.

In order to further validate our findings, CIFAR-10 data set was also used to conduct image classification under predefined architecture alternatives. The results (Table 7.9 and Table 7.10) also shows that a centralized architecture have quicker global model convergence while a decentralized architecture is better to perform classification on local edge data sets.

7.5 Discussion

According to experiment results, each architectural alternative demonstrates its advantages and disadvantages. In order to help industries easily understand the requirements and suitable scenarios for setting up a Federated Learning

Table 7.10: Local Prediction performance with CIFAR-10 data set follows a normal data distribution on the edge

	CIFAR-10 Local			
Number of Epochs	Central	Hierarchical	Regional	Decentral
10	73.01	71.83	75.22	79.31
30	77.68	75.35	79.56	83.67
50	86.07	86.23	87.95	88.24

system, we summarize our findings and suggestions in the following sections.

7.5.1 Centralized

A centralized architecture is suitable for a small scale Federated Learning system. Since there is only a single central point which manages all the participating nodes and provides model aggregation service. Thus, there is a high probability to cause the communication bottleneck if further increase the number of edge nodes.

In other words, companies that would like to speed up training speed and benefit from parallel training but only have small budgets should consider applying this alternative. They can also benefit from the advantages of easy configurations and nodes management with a centralized architecture compared to other options.

As for the model performance, use cases which require centralized knowledge of all distributed data samples should choose a more centralized architecture. For example, in a medical system, human activity recognition, etc [124][131], the number of participated edge nodes is usually small and those cases all need a common knowledge for the target prediction whose input training sample has similar distribution in different edge devices.

7.5.2 Hierarchical

A hierarchical architecture is an improved option compared with the centralized alternative. It is more suitable for a medium or relatively large scale system. The traffic of model updating is balanced because of the introduction of the regional nodes. This architecture is more suitable for companies which

Table 7.11: Comparison between different architecture alternatives

	Centralized	Hierarchical
Number of Edge Nodes	Small scale	Small-Medium scale
Model variation	Identical	Identical
Weights update latency	High	Medium
Model evolution	Slow	Slow
Example Domain	Medical Applications, Human Activity Recognition	Mobile Applications, Wireless Systems

	Regional	Decentralized
Number of Edge Nodes	Medium-Large scale	Large scale
Model variation	Localized	Localized
Weights update latency	Medium	Low
Model evolution	Fast	Fast
Example Domain	Weather Prediction, Geographic Applications, Vehicle and Traffic Application	IoT

need their system to be scalable and able to tolerant node failure. For example, in mobile applications and wireless systems [147], due to numerous connected devices, management and traffic balance point have to be introduced. Hierarchical architecture is the optimal choice to realize serviceable Federated Learning system. However, due to those extra servers, the system requires more budget and needs more resource for system setting and management.

7.5.3 Regional

Different from the previous two options, regional architecture removes the central aggregation node and replace it with several regional nodes. Similar to the hierarchical architecture, it supports a medium or relatively large scale system and needs a medium budget due to more server deployed.

Nevertheless, since the system removes the central point, the aggregated model could gain more knowledge from the local side, especially for those nodes whose data samples may have a similar distribution with their neighbours. This feature is most suitable for use cases such as weather prediction, geographic location detection, vehicle and traffic applications, etc [157][132]. Furthermore, the system can perform a faster model evolution based on local data but still can partially benefit from global knowledge.

7.5.4 Decentralized

For a decentralized architecture, the aggregation functions are moved to the edge devices. This option is suitable for a large and scalable system. Systems such as IoT and network constraint system [120][107] which don't want to waste resources on transmitting large amounts of data ought to consider this alternative. Furthermore, if a system which needs quickly model evolution and more knowledge from local samples (Sensors, etc.), it should choose the decentralized architecture.

However, a decentralized architecture requires a large budget to realize, as all edge devices need to support the local training and model transmission functions.

7.6 Conclusion and Future Work

In this paper, we introduce and compare four architecture alternatives for a Federated Learning system. We analyze the system performance with three important metrics, i.e. weights update latency, model evolution time, classification accuracy. For the model classification accuracy, a centralized system can formalize the global knowledge which covers all participated data samples while a decentralized alternative focuses more on local data sets in edge devices. Additionally, the weights update latency and model evolution time are much shorter in decentralized architectures than in centralized alternatives. Table 7.11 illustrates some of the insights we gained from the study we conducted in this paper.

Future work will include algorithm improvement on the architectures, such as traffic control, peer finding mechanism and neighbour selection methods, etc. Furthermore, additional efforts in studying hardware cost in those four architecture alternatives will take into consideration. Finally, we aim to realize real-world systems based on architecture alternatives reported in this paper.

CHAPTER 8

Real-time End-to-End Federated Learning: An Automotive Case Study

This chapter has earlier been published as
Real-time end-to-end federated learning: An automotive case study
Zhang H., Bosch J. and Holmström Olsson H.
In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 459-468). IEEE.

With the development of distributed edge computer computing and storage capabilities, using computation resources on the edge becomes a viable option [100]. Federated Learning has been adopted as a cost-effective solution due to its model-only sharing and parallel training characteristics. A simple diagram of a Federated Learning system is shown in Figure 8.1. Local model training is carried out in this framework, and data generated by edge devices do not need to be shared. Weight updates are instead sent to a central aggregation server, which generates the global model. The method overcomes the shortcomings of the conventional centralized Machine Learning approach, which only conducts model training on a single central server, such as data privacy, massive bandwidth costs, and long model training time.

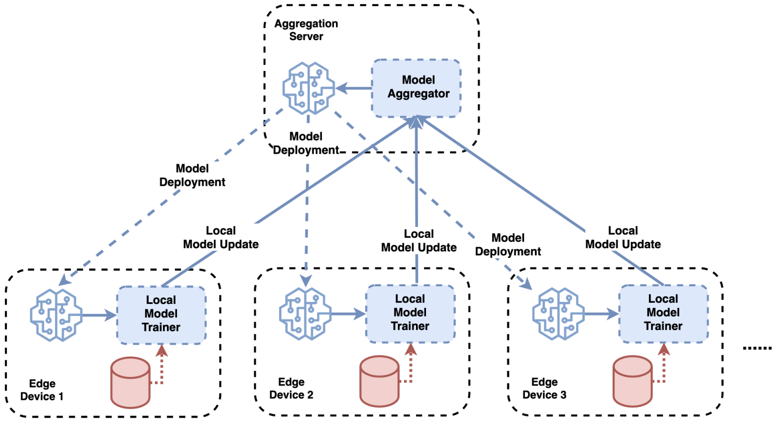


Figure 8.1: A typical Federated Learning System is depicted in the diagram. The light blue components are related to the model, while the red components are related to the data.

This paper builds on our previous research, “End-to-End Federated Learning for Autonomous Driving Vehicles” [158], in which we discovered that Federated Learning can significantly reduce model training time and bandwidth consumption. However, with the synchronous aggregation protocols used in our previous research and current Federated Learning applications and analysis, such as FedAvg [159], we realized that it is difficult for businesses to incorporate Federated Learning components into their software products [45]. Until model aggregation, a synchronous aggregation protocol requires the server to wait for all of the edge devices to complete their training rounds. Since real-world systems may include heterogeneous hardware configurations and network environments [160], the aggregation server cannot expect all participating edge devices to upload their local models at the same time. The situation will become worse and unmanageable with the increasing number of edge devices. Furthermore, our previous research also identified the challenges of deploying AI/ML components into a real-world industrial context. As J. Bosch et al. defined in *"Engineering AI Systems: A Research Agenda"* [150], AI engineering refers to AI/ML-driven software development and deployment in production contexts. We found that the transition from prototype to the production-quality deployment of ML models proves to be challenging for many companies [99] [151].

Therefore, in order to put Federated Learning into effect, in this paper, we present a novel method for consuming real-time streaming data for Federated Learning and combining it with the asynchronous aggregation protocol. This paper makes three contributions. First, we employ Federated Learning, a distributed machine learning technique, and validate it with a key automotive use case, steering wheel angle prediction in the field of autonomous driving, which is also a classic end-to-end learning problem. Second, we present a real-time end-to-end Federated Learning method for training Machine Learning models in a distributed context. Third, we empirically evaluate our approach on the real-world autonomous driving data sets. Based on our findings, we show the effectiveness of our method over other methods of learning, including the common synchronous Federated Learning approach.

The remainder of this paper is structured as follows. In Section 8.1, we introduce the background of this study. Section 11.4 details our research method, including the simulation testbed, the utilized machine learning method and the evaluation metrics. Section 8.4 presents the real-time end-to-end Federated Learning approach utilized in this paper. Sections 8.5 evaluates our proposed learning method to empirical data sets. Section 8.6 outlines the discussion on our observed results. Finally, Section 8.7 presents conclusions and future work.

8.1 Background

The first Federated Learning framework was proposed by Google in 2016 [101]. The main goal of it is to learn a global statistical model from a large number of edge devices. The problem is to minimize the following finite-sum objective function in particular 9.6:

$$\min_{w} f(w), \ \ where \ f(w) := \sum_{i=1}^{n} \lambda_i f_i(w) \tag{8.1}$$

Here, w represents model parameters, n is the total number of edge devices, and $f_i(w)$ is the local objective function which is defined by high dimensional tensor w of the i_{th} device. λ_i ($\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$) gives the impact of i_{th} remote device and is defined by users. This formula is also applied throughout this research.

With the advancement of the concept of cloud computing and decentralized data storage, there has been a surge of interest in how to use this approach to improve Machine Learning. [146] and [147] present two classic applications. The researchers implemented Federated Learning on the Google Keyboard platform to improve the accuracy of virtual keyboard search suggestions and emoji prediction. Their findings demonstrate the feasibility of using Federated Learning to train models while avoiding the transfer of user data. However, since the learning process is synchronous across all edge devices, the aggregation server must wait for all participating edge devices to complete their local training round before conducting model aggregation, which is inflexible and time-consuming while deploying into heterogeneous real-world systems [45]. Furthermore, because of the system environment and difficulties experienced when applying Federated Learning in various cases, we suggest the real-time end-to-end method and validate it in a radically different industrial scenario, steering wheel angle prediction.

8.2 Related Work

8.2.1 Steering Wheel Angle Prediction

One of the first pioneer research of utilizing the neural network for steering wheel angle prediction is described in [161]. The author used pixel information from simulated road images as inputs to predict steering command, which proves that a neural network is able to perform steering angle prediction from image pixel values. Recently, more advanced networks are utilized to predict the steering angles. H. M. Eraqi et al. propose a convolutional long short-term memory (c-LSTM) to learn both visual and dynamic temporal dependencies of driving, which demonstrate more stable steering by 87% [162]. Shuyang et al. [163] designed a 3D-CNN model with LSTM layers to predict steering wheel angles.

The concept of end-to-end learning was first proposed in [164], where authors built and constructed a deep convolutional neural network to directly predict steering wheel angles and monitor the steering wheel. In this research, ground truth was directly captured from real-time human behaviour. Their methods demonstrate that a convolutional neural network can learn steering wheel angle directly from input video images without the need for additional road information such as road marking detection, semantic analysis, and so on. In order to enhance model prediction accuracy, we use a two-stream model in our approach. Due to its robustness and lower training cost as compared to other networks such as DNN [165], 3D-CNN [163], RNN [162], and LSTM [166], the model was first proposed in [167] and applied in [168]. However, previous research for this use case has concentrated primarily on the training model in a single-vehicle. We will use Federated Learning in this paper to accelerate model training speed and boost model quality by forming a global awareness of all edge vehicles.

8.2.2 Federated Learning in Automotive

The automotive industry is a promising platform for implementing Machine Learning in a federated manner. Machine learning models can be used to forecast traffic conditions, identify pedestrian behaviour, and assist drivers in making decisions [169][157]. However, since vehicles must have an up-to-date model for safety purposes, Federated Learning has the potential to accelerate Machine Learning model development and deployment while protecting user privacy [170].

On top of Federated Learning, Lu et al. [116] test the failure battery for an electric vehicle. Their methods demonstrate the efficacy of privacy serving, latency reduction, and security protection. Saputra et al. [117] forecast the energy demand for electric vehicle networks. They dramatically minimize the bandwidth consumption and efficiently protect sensitive user information for electric vehicle users by using Federated Learning. Samarakoon et al. [115] propose a distributed approach to joint transmit power and resource allocation in vehicular networks that enable low-latency communication. When compared to a centralized approach, the proposed method can reduce waiting queue length without increasing power consumption and achieve comparable model prediction efficiency. Doomra et al. [171] present a Federated Learning-trained long short-term memory (LSTM)-based turn signal prediction (on or off) model. All of these approaches, however, are faced with synchronous aggregation protocols that are unsuitable for real-world heterogeneous hardware. As a result, in this paper, we present an asynchronous aggregation protocol combined with Federated Learning and validate it with one of the most essential use cases in the automotive industry.

8.3 Method

The analytical technique and research method mentioned in [92] were used in this study to conduct a quantitative measurement and comparison of real-time Federated Learning and conventional centralized learning methods. The article presents some recommendations for applying machine learning methods to software engineering activities, as well as methods for demonstrating how they can be conceived as learning problems and addressed in terms of learning algorithms. The mathematical notations, testbed and hardware configuration, convolutional neural network, and evaluation metrics used to solve the problem of steering wheel angle prediction are presented in the following sections.

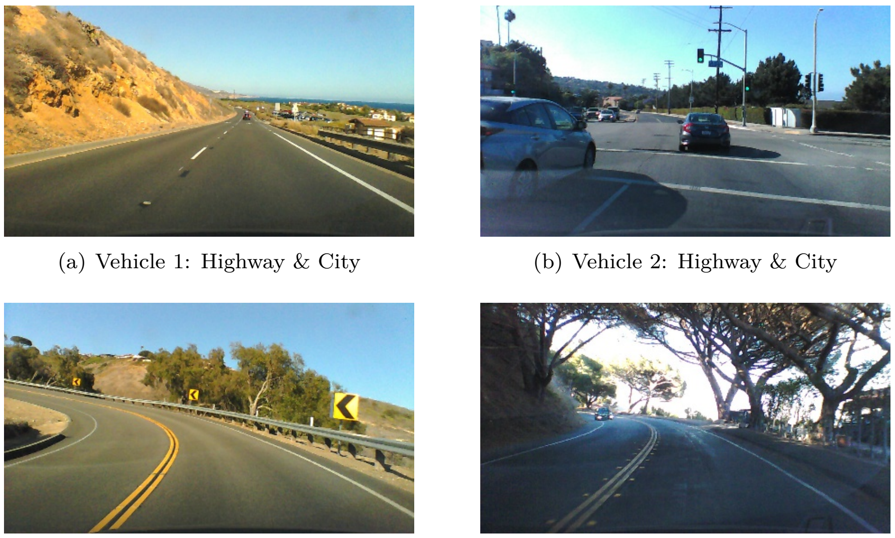
8.3.1 Mathematical Notations

The mathematical notations that will be used in the paper are introduced here first:

A_t	An image frame matrix at time t
$O_t = f(A_t, A_{t-1})$	An optical-flow matrix at time t
θ_t	Steering wheel angle at time t
$\hat{\theta}_t$	Predicted steering wheel angle at time t

8.3.2 Data Traces and Testbed

The datasets used in this paper are from the SullyChen collection of labelled car driving data sets, which can be found on Github under the tag [172]. To conduct our experiments, we chose Dataset 2018 from this collection. The dataset contains various driving data such as road video clips, steering angle on roads, and so on. Dataset 2018 is 3.1 GB in size and contains approximately 63,000 files. This dataset tracks a 6-kilometer path along the Palos Verdes Peninsula in Los Angeles. Our experiment datasets were chosen from the first 40,000 image frames.



(a) Vehicle 1: Highway & City

(b) Vehicle 2: Highway & City

(c) Vehicle 3: Hill

(d) Vehicle 4: Hill & City

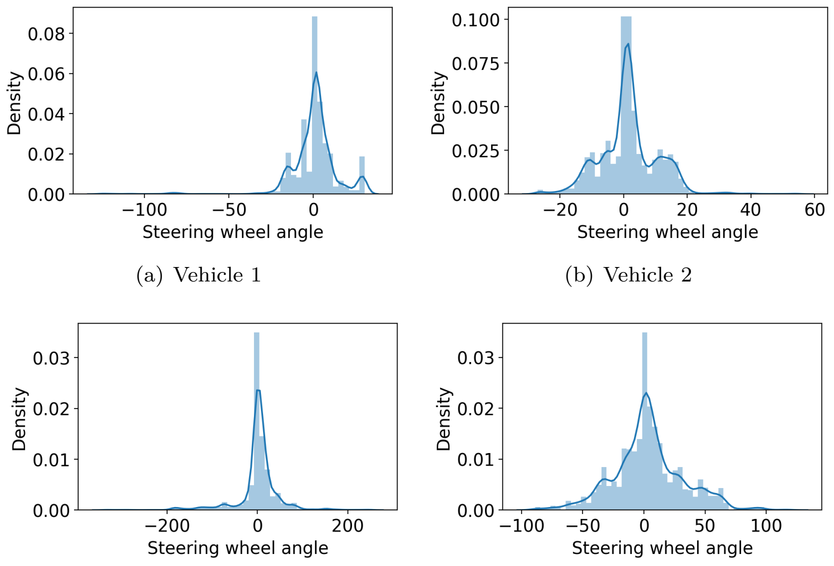
Figure 8.2: Driving scenarios in each edge vehicle.

The data streams were simulated on four edge vehicles to provide a comprehensive evaluation. The data was divided into four sections and distributed to edge vehicles prior to our simulation. In each edge vehicle, the first 70% of data are considered input streaming driving information that was used for model training, while the remaining 30% are potential stream information. As shown in Figure 8.2, training datasets for each edge vehicle in our experiment include a variety of driving scenarios.

Table 8.1: Hardware setup for testbed servers

CPU	Intel(R) Xeon(R) Gold 6226R
Cores	8
Frequency	2.90 GHz
Memory	32 GB
OS	Linux 4.15.0-106-generic
GPU	Nvidia Tesla V100 GPU (Edge vehicle 1) Nvidia Tesla T4 GPU (Edge vehicle 3, 4)

The data distribution in each edge vehicle is depicted in Figure 8.3. When driving on a hill, the steering wheel angles have a greater range than when driving on a highway or in a neighbourhood. The majority of driving angles in edge vehicles 1 and 2 falls within the range $[-50°, 50°]$, while in edge vehicles 3 and 4, the range is $[-100°, 100°]$. The graph shows that when driving on a hill, vehicles may encounter more turns than when driving on a highway or in a city.



(a) Vehicle 1 (b) Vehicle 2

(c) Vehicle 3 (d) Vehicle 4

Figure 8.3: Data distribution in each edge vehicle.

The models were continuously trained based on the recorded data and used future streaming driving data to perform prediction and validation on the steering wheel angle information.

The hardware information for all of the servers is given in table 8.1. To simulate aggregation and edge functions, one of the five servers was designated as the aggregation server, while the others operated as edge vehicles. In order to simulate a heterogeneous edge area, GPU settings were only available in Vehicles 1, 3, and 4 (Vehicle 1: Tesla V100, Vehicle 3, 4: Tesla T4).

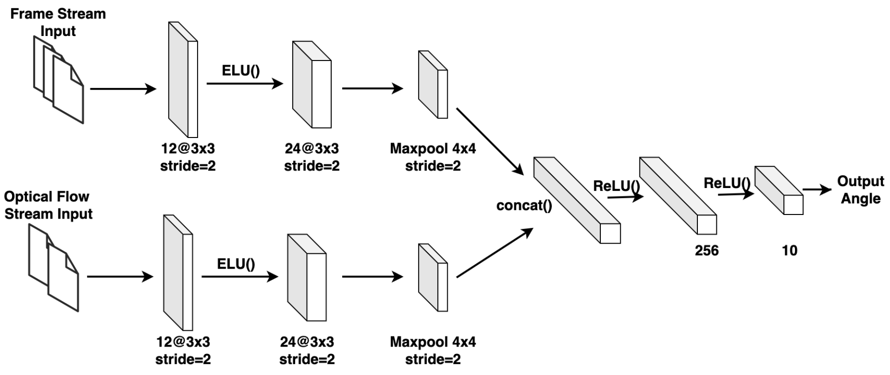


Figure 8.4: The two input branches each have two 3x3 convolution layers in a convolutional neural network. The first layer has 12 output channels that are enabled with the ELU function, while the second layer has 24, which is then followed by 4x4 max pooling. All with stride values of 2 or higher. With the ReLu activation, there are two completely linked layers with 250 and 10 units after concatenating two branches.

8.3.3 Machine Learning Method

In this paper, steering wheel angle prediction is performed using a two-stream deep Convolutional Neural Network (CNN) [167] [168]. The architecture is described in detail in Figure 8.4. Each stream in our implementation has two convolutional layers and a max-pooling layer. After concatenating, there are two fully-connected layers activated by the ReLU function.

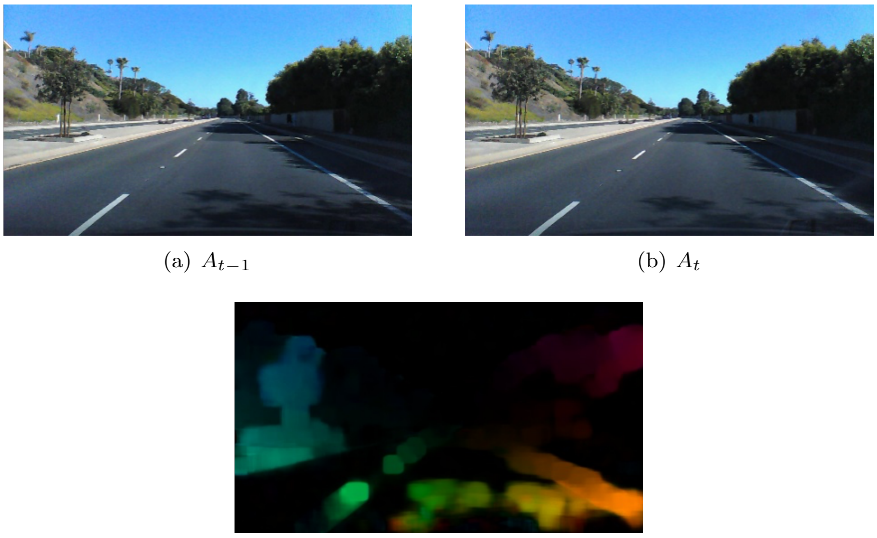
The model has two distinct neural branches that take spatial and temporal information as inputs to two streams and then output the expected steering angle. The model consumes three frames of RGB images for the first stream, which can be denoted as $\{A_{t-2}, A_{t-1}, A_t\}$. The second stream is a two-frame optical flow measured from two consecutive frames $O_{t-1} = f(\{A_{t-2}, A_{t-1}\})$ and $O_t = f(\{A_{t-1}, A_t\})$.

Optical flow is a typical temporal representation in video streams that captures the motion differences between two frames [173]. The optical flow calculation method used in this paper is based on Gunnar Farneback's algorithm, which is implemented in OpenCV [174]. Figure 8.5 shows an example optical flow matrix created by two consecutive image frames.

The aim of training a local convolutional neural network is to find the model parameters that result in the smallest difference between the prediction and ground truth steering angles. As a result, we choose mean square error as the local model training loss function in this case:

$$Loss = \frac{1}{N}\sum_{t=1}^{N}(\theta_t - \hat{\theta}_t)^2 \tag{8.2}$$

Here, N represents the batch size while θ_t and $\hat{\theta}_t$ represent the ground truth and the predicted steering wheel angle value at time t. During the process of model training in each edge vehicle, all the image frames will be firstly normalized to $[-1, 1]$. The batch size is 16 while the learning rate is set to 10^{-5}. The optimizer utilized is Adam [175], with parameters $\beta_1 = 0.6$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$.



(a) A_{t-1}

(b) A_t

(c) $O_t = f(\{A_{t-1}, A_t\})$

Figure 8.5: Example of the optical flow (a) Previous Frame (b) Current Frame (c) Optical flow of current vision frame.

8.3.4 Evaluation Metrics and Baseline Model

We chose three metrics and three baseline models in order to provide fruitful outcomes and assessment. The three metrics include angle prediction performance, model training time and bandwidth cost:

- **Angle prediction performance**: Root mean square error (RMSE), a common metric for measuring the difference between prediction results and ground truth. The metrics will provide a reasonable estimate of the trained model's quality in each edge vehicle.
- **Model training time**: The total time cost for training a model at the edge vehicles is known as this metric. As a consequence, the average of four edge vehicles is obtained. This metric shows the pace at which local edge devices update their model, which is critical for systems that need to evolve quickly in order to adapt to a rapidly changing environment. By testing the model deployment timestamp, the metrics were calculated in all of the vehicles.
- **Bandwidth cost**: The total number of bytes transmitted during the entire training procedure is known as this metric. This metric shows the overall cost of communication resources needed to achieve an applicable convolutional neural model.

The three baseline models include models trained by applying the traditional centralized learning approach, the locally trained model without model sharing and the Federated Learning with the synchronous aggregation protocol:

- **Traditional Centralized Learning model (ML)**: This baseline model was trained using a centralized learning method, which is still widely used in current machine learning research and software applications. All data from edge vehicles is collected to a single server prior to model training. The hyper-parameters of this model training are identical to those of Federated Learning, as described in section 8.3.3. The results can then be compared to models trained using Federated Learning techniques.
- **Locally trained model without model sharing (Local ML)**:

Each edge vehicle is used to train this baseline model. In contrast to Federated Learning, no models will be exchanged during the training process. The prediction accuracy can be applied to the Federated Learning model to see if Federated Learning outperforms those independently trained local models.

- **Synchronous Federated Learning (FL)**: FedAvg is the algorithm applied here. It is a synchronous method that is widely used in Federated Learning research. Before aggregating global models, the server has to wait for all participants to finish updating their local models.

8.4 Real-time End-to-End Federated Learning

This section describes the algorithm and method used in this article. The diagram of the learning process in a single edge vehicle is shown in Figure 8.6. Images are firstly stored in a fixed-sized storage window in order to conduct real-time end-to-end learning based on the continuous image stream. When the storage window reaches its size limit, the most recent picture frames are moved into the training window, while an equivalent number of old frames are dropped. (In our case, the storage window is 100 images wide and the training window is 2,000 wide. These values provide us with the highest model prediction accuracy.) The optical flow information is measured at the same time. Inside the training window, image frames and optical flow frames are fed into a convolutional neural network. The network's performance is compared to the ground truth for that picture frame, which is the human driver's recorded steering wheel angle. Back-propagation is used to adjust the weights of the convolutional neural network in order to enforce the model output as close to the target output as possible.

Following the completion of each training epoch, local models in edge vehicles will be updated to the aggregation server, forming a continuous global awareness among all participating edge vehicles. The following are the steps of the algorithm used in this paper (Algorithm 5):

Step 1: Edge vehicles compute the model locally; after completing each local training epoch, they retrieve the global model version and compare it to their local version. The decision is based on the frequency bound limits (a_l and a_u) and the model version difference ver (global model

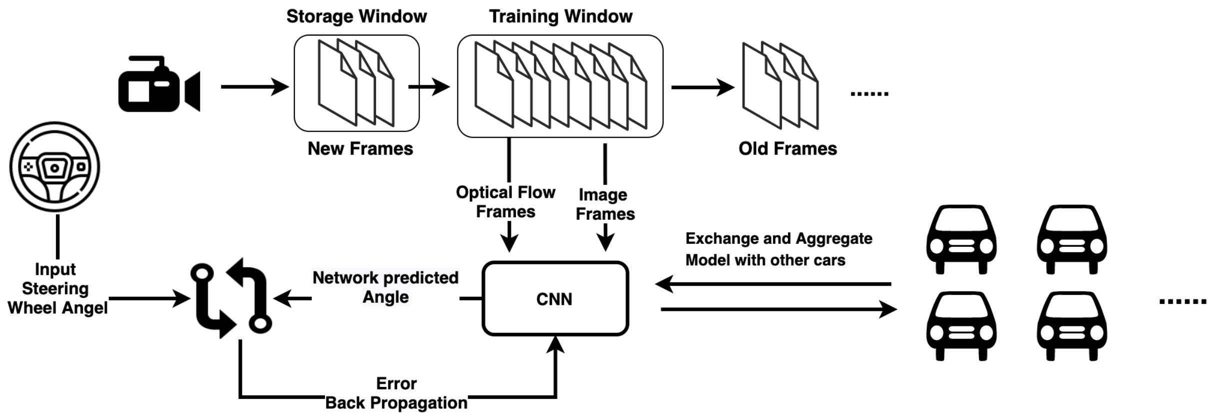


Figure 8.6: Diagram of real-time end-to-end Federated Learning in a single vehicle.

Algorithm 5: Asynchronous Federated Learning: In the system, total K edge vehicles are indexed by k; B is the local mini-batch size; E represents the number of local epochs, and γ is the learning rate.

```
Function Server_Function():
    initialize $w_0$
    initialize $ver \longleftarrow a_l$
    while True do
        $w^k_{t+1}, ver_k \longleftarrow Client\_Update(w_t, ver)$;
        $w_{t+1} \longleftarrow (1 - \alpha) \times w_t + \alpha \times w^k_{t+1}$
        where $\alpha = \frac{1}{ver - ver_k + 1}$;
        $ver \longleftarrow ver + 1$;
    end
End Function
Function Client_Update($w$, $ver$):
    $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
    while True do
        for each local epoch $i$ from 1 to $E$ do
            for batch $b \in \beta$ do
                $w \longleftarrow w - \gamma \nabla l(w; b)$;
            end
        end
        When ready for an update, pull global model version $ver$ from
         the server
        if $ver - ver_k > a_u$ then
            // Client version is too old
            Fetch w, ver from the server
            continue
        else if $ver - ver_k < a_l$ then
            // Client version is too close to the global
            continue
        else
            return $w$, $ver$ to server
    end
End Function
```

version) and $verk$ (local model version of edge vehicle k). The upper limit of the model version difference is represented by a_u, while the lower limit is represented by a_l. There are three conditions:

- If the local version is out of date (the client version is too old), the edge vehicle can retrieve the most recent model and conduct local training again.
- If the local version is too similar to the latest version (Client is too active), it should stop upgrading and re-train locally.
- Clients should then submit modified model results to the aggregation server if the local version is between the upper and lower limits.

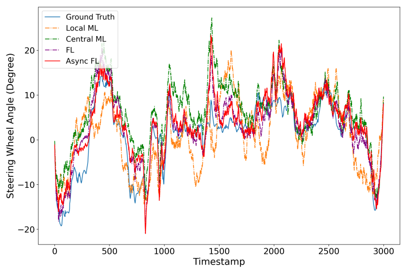
Step 2: In order to form a global awareness of all local models, the central server performs aggregation based on the ratio determined by the global and local model versions.

Step 3: The aggregation server returns the aggregated result to the edge vehicles that request the most recent model.

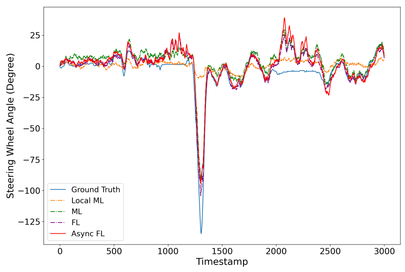
Since the algorithm is push-based, the aggregation server only deploys the global model if the edge vehicles request it. When the edge vehicles update their local models, the server aggregates them based on the local model version. The older the model version, the lower the ratio when shaping the global model. Furthermore, although the model update frequency is entirely dependent on local hardware settings, there are two bound limits in place to ensure that the update frequency of local clients is within a reasonable range $[a_l, a_u]$. (In our case, based on the number of the participated vehicles, the lower frequency bound a_l we set equals to 2 while the upper bound a_u is 6.)

8.5 Results

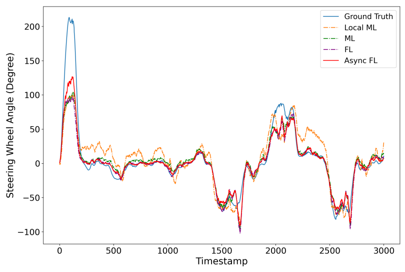
We present the experiment results of the real-time end-to-end Federated Learning approach to steering wheel angle prediction in this section. The device output is evaluated based on three factors, as defined in Section 11.4. (The metrics are described in 13.3.5.) - (1) Angle prediction performance (2) Model Training Time (3) Bandwidth cost. The results are compared with other three



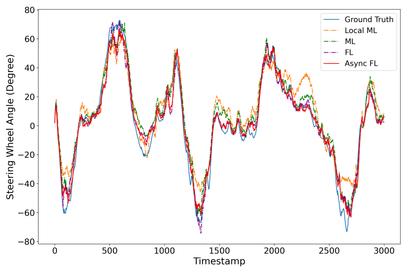
(a) Vehicle 1



(b) Vehicle 2



(c) Vehicle 3

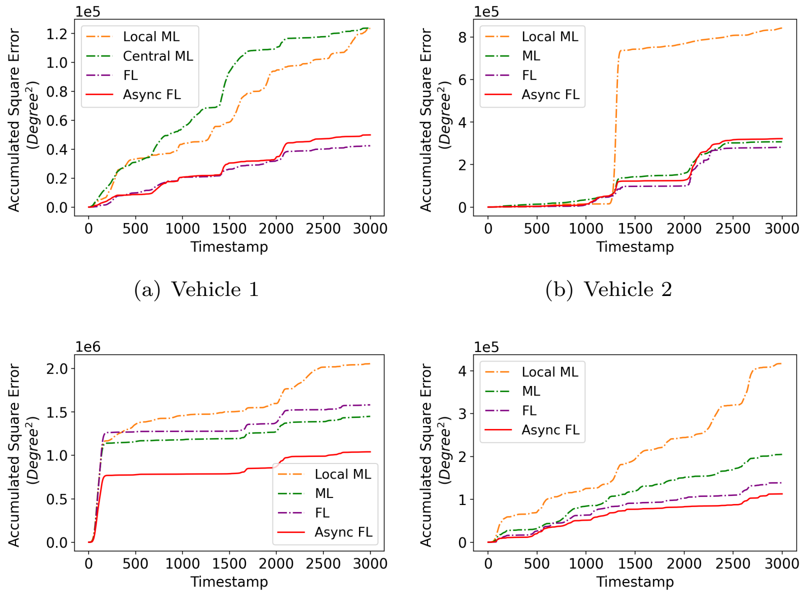


(d) Vehicle 4

Figure 8.7: The comparison of angle prediction performance on four local vehicle test set with Federated Learning and three baseline models.

baseline models which are trained by - 1) Traditional Centralized Learning (ML) 2) Local training without model sharing (Local ML) 3) Synchronous Federated Learning (FL)

Figure 13.4 compares the angle prediction output of the model trained by asynchronous Federated Learning (Async FL) to the other baseline models. The results show that the Federated Learning models (synchronous and asynchronous) may achieve the same or even better prediction accuracy than the traditional centralized trained model. The Federated Learning model reacts faster than other learning approaches, particularly at the timestamps that require rapid changes in steering wheel angle. Furthermore, when compared to independently trained models, Federated Learning approaches can provide a much better prediction that is much closer to the ground truth.



(a) Vehicle 1 (b) Vehicle 2

(c) Vehicle 3 (d) Vehicle 4

Figure 8.8: Accumulated error on test dataset in 4 edge vehicles with asynchronous Federated Learning and other baseline models.

To provide a clear view of model output with different approaches, we accumulated the square error between expected angle and ground truth (calculated by $(\theta_t - \hat{\theta}_t)^2$) and demonstrate it in Figure 8.8. The results provide the same information as Figure 13.4. We find that asynchronous Federated Learning outperforms centralized learning and local machine learning. In addition, as compared to synchronous Federated Learning, our method achieves higher prediction accuracy in edge vehicles 3 and 4. Table 8.2 displays detailed numerical results, including the regression error (RMSE) on each test dataset in each vehicle and the overall average accuracy among the test datasets of all participating edge vehicles.

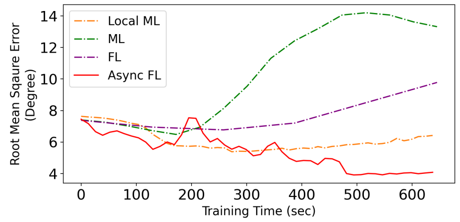
Table 8.2: Steering wheel angle regression error (RMSE) on test set of each edge vehicle (4 vehicles in total)

	Vehicle 1	Vehicle 2	Vehicle 3	Vehicle 4	Overall
Async FL	4.077	10.358	**18.629**	**6.129**	**11.275**
FL	3.758	9.933	22.967	6.795	12.754
ML	6.422	10.118	21.985	8.264	13.183
Local ML	6.416	16.749	26.196	11.788	16.954

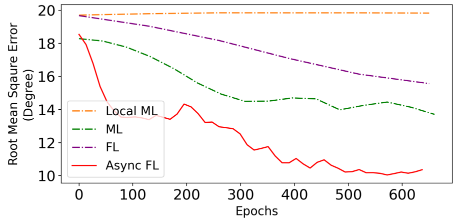
The findings show that asynchronous Federated Learning outperforms other baseline models in vehicles 3 and 4. In vehicle 1 and 2, models trained by asynchronous Federated Learning only perform about 0.2 and 0.4 worse than the synchronous Federated learning method. Based on our findings, we may conclude that the asynchronous Federated Learning model can provide better prediction performance than the local independently trained model, and its behaviour can achieve the same or even higher accuracy level when compared to centralized learning and the synchronous Federated Learning model.

Furthermore, Figure 8.9 illustrates the shift in regression error with model training time in order to evaluate model training efficiency. The results show that the asynchronous Federated Learning method outperforms all of the baseline approaches in terms of model training efficiency. With the same training period, our approach can achieve better prediction efficiency (with approximately 50% less regression error) and converge approximately 70% faster than other baseline models.

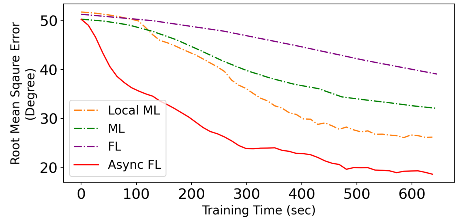
The comparison of total training time and bytes transferred between Federated Learning and three baseline models is shown in table 8.3. For all the models, the total number of training epochs is 50. With async FL, FL, and Local ML learning approaches, model training is accelerated by Nvidia Tesla



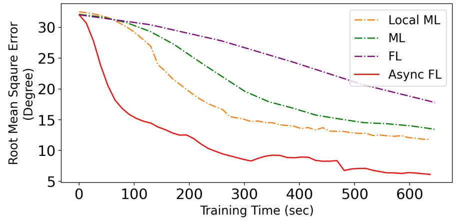
(a) Vehicle 1



(b) Vehicle 2



(c) Vehicle 3



(d) Vehicle 4

Figure 8.9: The comparison between angle prediction performance and the model training time on four local vehicle test set with asynchronous Federated Learning and three baseline models.

Table 8.3: Total Training Time and Bandwidth cost with different model training methods (4 Vehicles in total)

	Async FL	FL	ML	Local ML
Training Time (sec)	**669.2**	5,982.8	2143.7	5,903.4
Bytes Transferred (GB)	**0.78**	0.78	2.02	-

V100 GPU in edge vehicle 1, while model training is accelerated by Nvidia Tesla T4 GPU in edge vehicle 3, 4. The ML method completes training on a single server with Nvidia Tesla T4 GPU acceleration. As compared to the traditional centralized learning approach, the bandwidth cost of both Federated Learning methods is reduced by approximately 60%. The results for model training time indicate that asynchronous Federated Learning needs significantly less training time than other baseline methods. However, since there is no GPU available for synchronous Federated Learning and local learning, edge vehicle 2 becomes the burden of the entire system. Other vehicles must wait for vehicle 2 to complete its local training round before performing model aggregation and further training tasks, which is inflexible and time-consuming. The performance of these two methods is even lower than that of the centralized learning system with GPU acceleration. In summary, as compared to the traditional centralized learning process, asynchronous Federated Learning reduces training time by approximately 70% and saves approximately 60% bandwidth. Since our method consumes real-time streaming data, there is no need to store and train on a large dataset in a single edge unit, making it cost-effective and relevant to real-world systems.

8.6 Discussion

Based on the findings of our experiments, our method has major advantages over widely used centralized learning and synchronous Federated Learning approaches. Our asynchronous approach achieves the same or better model prediction accuracy while substantially reducing model training time and bandwidth costs. Our method not only outperforms synchronous Federated Learning in terms of forming the global model of entire datasets without requiring any user data transmission, but it also tolerates heterogeneous hardware settings of different edge devices and dramatically improves model training performance. Furthermore, the model quality is greatly improved and can produce much better results with the model sharing and aggregation process.

Because of these benefits, real-time end-to-end Federated Learning can assist in a number of other meaningful use cases. The technique described in this paper can be applied not only to self-driving vehicles, but also to other applications that involve continuous machine learning model training on resource-constrained edges, such as camera sensors, cell phones, household electrical appliances, and so on. Furthermore, due to user data privacy and network bandwidth limitations, our approach can be implemented in systems that need a constantly evolving model to adapt to rapidly changing environments.

8.7 Conclusion and Future Work

In this paper, we present a novel approach to real-time end-to-end Federated Learning using a version-based asynchronous aggregation protocol. We validate our approach using a critical use case, steering wheel angle prediction in self-driving cars. Our findings show the model's strength and advantages when trained using our proposed method. In our case, the model achieves the same or even better prediction accuracy than widely used centralized learning methods and other Federated Learning algorithms while reducing training time by 70% and bandwidth cost by 60%. Note that the decrease would be more visible if the number of participating devices is expanded more, which proves to be cost-effective and relevant to real-world systems.

In the future, we plan to further analyze our algorithm with different combinations of hyper-parameters, such as the aggregation frequency bound a_l and a_u. As the parameter settings become more important with the number of participating learning vehicles increases, we would like to add more federated edge users in order to test device output that may differ with these bounds. In addition, we will test our approach in additional use cases and investigate more sophisticated neural networks combined with our approach. In addition, we plan to develop more appropriate aggregation algorithms and protocols in order to increase model training performance on resource-constrained edge devices in real-world embedded systems.

CHAPTER 9

AF-DNDF: Asynchronous Federated Learning of Deep Neural Decision Forests

This chapter has earlier been published as
AF-DNDF: Asynchronous Federated Learning of Deep Neural Decision Forests
Zhang H., Bosch J., Holmström Olsson H. and Koppisetty, A.C.
In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 308-315). IEEE.

Federated learning is an emerging machine learning methodology that was first proposed by Google [101] in 2016. The concept was originally used to solve the problem of local model training and updating in Android mobile devices [103]. The design goal of Federated Learning is to carry out efficient machine learning among multiple parties or multiple computing end nodes with the purpose of protecting the privacy of the end-user personal data during big data exchange. Since the edge devices in Federated Learning train the machine learning models continuously on new data, bottlenecks with centralized training and deployment of ML models on edge are minimized. Due to these characteristics, the advantage of Federated learning is significant. It is capable to utilize local computation resources and ease the computation pressure of the central server. Furthermore, the system can provide rapid model deployment and evolution because of the local training fashion [176].

In addition, the machine learning algorithm that can be used in Federated Learning is not limited to neural networks, but can also include other important algorithms such as the random forests, etc. With the inspiration of [177], we further investigate the concept of Deep Neural Decision Forests (DNDF) and the way to combine it with Federated Learning. As the network unites deep neural networks and decision forests, the methodology leverages the robustness of decision trees where the final fully connected layer in convolutional neural networks are sensitive. Thus, it is desirable to be utilized for classification tasks with the help of Federated Learning.

Although the concept of Deep Neural Decision Forests and training the model with the Federated Learning method has significant benefits, it is often a complicated process for industries and companies to build a reliable and applicable Federated Learning system [99]. Our previous research shows the challenges of deploying Artificial Intelligent (AI)/Machine Learning (ML) components into a real-world industrial context. As we defined in *"Engineering AI Systems: A Research Agenda"* [150], AI engineering refers to AI/ML-driven software development and deployment in industrial production contexts. We found that the transition from prototype to the production-quality deployment of ML models proves to be challenging for many companies [151].

In this paper, in order to make the concept to be applicable to real-world industrial requirements, such as heterogeneous hardware settings and limited communication bandwidth, we propose a novel algorithm "AF-DNDF". The contribution of this paper is threefold. First, we combine the asynchronous Federated aggregation protocol with the concept of Deep Neural Decision Forests. The asynchronous approach can enhance the model training efficiency among all participated edge devices. Second, we introduce an optimal method for selecting decision trees based on their classification performance, which significantly reduces the communication bandwidth when updating local models to the aggregation server. Third, we evaluate our approach with an important automotive use case, road object recognition in the field of autonomous driving. Based on our results, we show that our AF-DNDF algorithm significantly reduces the communication overhead and, at the same time, accelerates model training speed without sacrificing model classification performance, which turns out to be more suitable when deploying the method in an industrial context.

The remainder of this paper is structured as follows. In Section 9.1, we introduce the background of this study. Section 9.2 details our research method, including the simulation testbed, the utilized machine learning method and the evaluation metrics. Section 9.3 presents the Asynchronous Federated Deep Neural Decision Forests approach proposed in this paper. Sections 9.4 includes evaluation of the proposed method to data sets that are relevant to industrial applications. Section 9.5 outlines the discussion on our observed results. Finally, Section 9.6 presents conclusions and future work.

9.1 Background

9.1.1 Deep Neural Decision Forests

Deep neural decision forests were firstly were first introduced by Kontschieder et al. [178] in the year 2015. Their results demonstrated that the method outperforms the baseline convolutional neural network and random forest with the same individual architectural settings. A DNDF consists of two parts, namely convolutional neural network and decision forests. Convolutional neural networks are often used in image classification and object detection because of their excellent performance without explicit feature extraction. By using different convolution kernels, features can be extracted from the data source. The decision forest is also a common machine learning algorithm based on the tree structure. Due to the low complexity and strong learning ability, decision forests have been successfully applied to many machine learning problems [179].

DNDF algorithm is a modification of convolutional neural networks where the final softmax layer of CNNs is replaced by decision forests. Predictions are made by applying a certain routing algorithm in the decision tree in order to reach the last leaf node. Figure 9.1 demonstrate a specific network structure.

In this network, the full connection layer and the previous layer are the same as the general convolutional neural network, and the mapping from the full connection layer node to the decision node is the same.

$$d_n(x;\theta) = \sigma(f_n(x;\theta)) \tag{9.1}$$

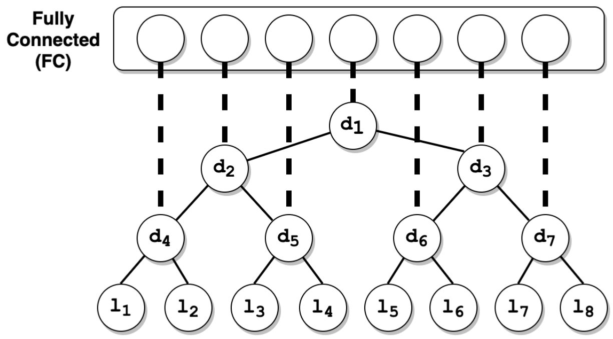


Figure 9.1: Network Structure of the Deep Neural Decision Forests

where X is the input. θ represents the parameter. σ is the sigmoid function. The function realizes the mapping from full connection layer to the decision nodes. In order to reach the leaf node through the tree, we need to plan the routing algorithm:

$$\mu_l(x|\theta) = \prod_{n \in N} d_n(x;\theta)^{\swarrow} \bar{d_n}(x;\theta)^{\searrow} \tag{9.2}$$

where $d_n(x;\theta) = 1 - d_n(x;\theta)^{\swarrow}$. N is the decision node set. $d_n(x;\theta)^{\swarrow}$ indicates the route from the current node to the left while l is the leaf node. According to the formula, if we want to route to leaf node 4:

$$\mu_{l_4} = d_1(x)\bar{d_2}(x)\bar{d_5}(x) \tag{9.3}$$

The probability of classifying input x as y is:

$$P[y|x,\theta,\pi] = \sum_{l \in L} \pi_{l_y} \mu_l(x|\theta) \tag{9.4}$$

For the decision forests $F = T_1, T_2, ..., T_k$:

$$P_F[y|x] = \frac{1}{k} \sum_{h=1}^{k} P_{T_h}[y|x] \tag{9.5}$$

The accuracy of classification can be significantly improved by discriminating different decision trees in the decision forests.

9.1.2 Federated Learning and Asynchronous Aggregation

Machine Learning has attracted tremendous attention from both research and society. The main challenge of it is: although the computation capability continues to increase with time, the computational needs of many Machine Learning systems grow even faster [99]. For example, when applying deep neural networks, in order to achieve good model performance, the network needs to contain millions or even billions of neurons, which may result in the problem of longer training time and less model flexibility [98].

With the concept of cloud computing and decentralized data storage, AI engineering [150] has the opportunity to expand to a distributed setting. Federated Learning is proposed to improve traditional Machine Learning approaches, as it enables edge devices to collaboratively and continuously learn a shared Machine Learning model. The theory of Federated Learning has been explored previously in [101][102] where the main goal was to build a global statistical model using a variety of edge devices. The challenge is to minimize the following finite-sum objective function 9.6 in particular:

$$\min_{w} f(w), \ \ where \ f(w) := \sum_{i=1}^{n} \lambda_i f_i(w) \tag{9.6}$$

Here, w denotes model parameters, n the total number of edge devices, and $f_i(w)$ the local objective function defined by the ith device's high-dimensional tensor w. λ_i ($\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$) is the impact of the ith remote device which is defined by users.

With the concept first applied by Google in 2016 [103], there have been several Federated Learning architectures, frameworks and solutions proposed to solve real-world issues [136]. A Federated Learning system requires no transfer of the edge data but only local model updates are sent to a central aggregation server to form a consensus global knowledge. The model updates could be performed either by updating the complete model architecture or just by updating the model parameters and hyper-parameters. Furthermore, with the local training and validation, a Machine Learning model can be quickly and continuously verified and deployed, which is more suitable for a quick-evolving system.

However, the commonly applied Federated Learning aggregation algorithms, such as FedAvg [159], assumes that all the participated edge devices have the same computation power and be able to update the models at the same time, which is incompatible with the industrial cases. In our previous research [180], we proposed a version-based asynchronous aggregation protocol to tackle these challenges. With the asynchronous aggregation protocol, the edge devices no longer need to wait for other equipment to complete their model training round but directly send the local model to the aggregation server. In this paper, we will not only combine asynchronous aggregation protocol with DNDF but also optimize the local model updating procedure by further reducing the communication overhead without sacrificing the model classification performance.

9.2 Method

To produce a quantitative assessment and comparison between Federated Learning and centralized learning techniques, the empirical method and learning procedure provided in [92] were applied in this study. We also compared our AF-DNDF approach with the commonly used synchronous Federated Learning algorithm. In the following sections, we present our testbed, data traces, the convolutional neural network architecture and hyper-parameters of decision forests that were utilized in this research.

9.2.1 Data Traces and Testbed

In this research, in order to provide a comprehensive evaluation, we selected two well-known automotive driving data sets, namely FLIR and BDD 100K data set.

FLIR dataset is a thermal image data set [181]. With the development of thermal imaging cameras, the automotive industry has begun to explore the use of thermal imaging for machine learning to develop advanced driving assistance and autonomous driving systems. The data set contains annotated thermal images of day and night scenes, from which we extracted three categories of road objects. Figure 9.2 demonstrate the example samples in the data set.

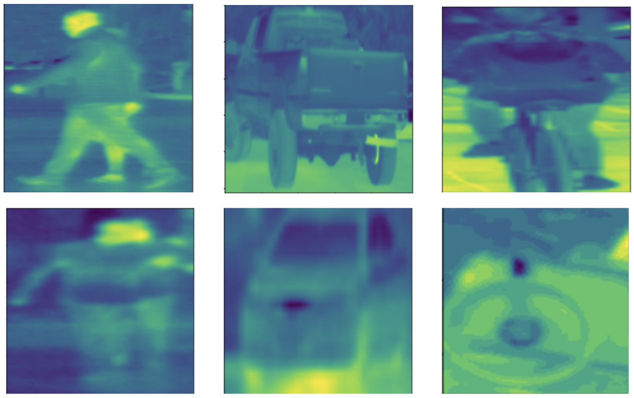


Figure 9.2: FLIR Thermal Sensing Advanced driver-assistance system Dataset

The second data set applied is BDD 100K [182], which was released by the AI Laboratory of Berkeley University. The data set contains the largest and most diverse public driving records. The data sets contain 10,000 pieces of high-definition video. Each video is about 40 seconds while the keyframe is sampled to get 10,000 pictures. The image size is a 1280x720 RGB image. Each image file is pointed to a specific number that can be found in every label image. In order to perform objective recognition, in this paper, we extracted 60,000 samples from the RGB frame images based on the labelled objective bounding area. The training data contains six different road object classes with 10,000 samples in each category. Figure 9.3 gives an example of the selected data set.

Before our simulation, as we included three vehicles in our experiment, the data from both data sets were divided into three parts and uniformly distributed to those edge vehicles. In each edge vehicle, the first 70% data were regarded as the input driving information which was used for model training while the rest 30% were considered as the test set. All the image samples were resized to 512×512 and normalized to $[-1, 1]$.

Hardware information for all of the servers is provided in Table 9.1. In order to simulate aggregation and edge operations, one server was designated as the aggregation server, while the others were designated as edge vehicles.

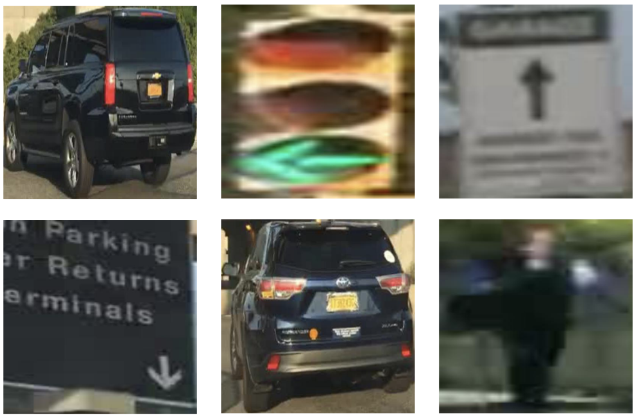


Figure 9.3: BDD100K: A Large-scale Diverse Driving Video Dataset

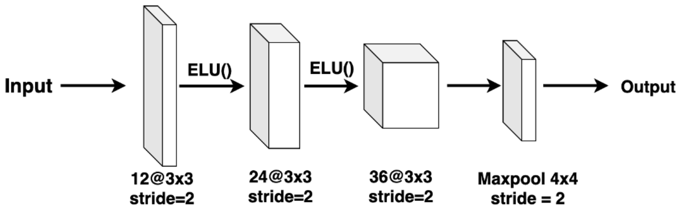


Figure 9.4: Convolutional neural network layer description

9.2.2 Machine Learning Method

In order to find the optimal network, the random search [183] strategy was applied. The following is the detailed description of the convolutional neural network architecture and for the forest hyper-parameters, we selected parameters that eventually gave the best classification accuracy. Figure 9.4 illustrates the convolutional neural network part of our deep neural decision forest architecture. Three 3x3 convolution layers were set in the input branch, which has 12 output channels. The second layer contains 24 output channels while the third layer has 36 output channels, followed by 4x4 max pooling. All layers

Table 9.1: Hardware setup for testbed servers

CPU	Intel(R) Xeon(R) Gold 6226R
Cores	8
Frequency	2.90 GHz
Memory	32 GB
OS	Linux 4.15.0-106-generic
GPU	Nvidia Tesla T4 GPU

are activated with the ELU function [184]. The output is connected to the decision forests. The convolutional neural network is acting as a feature layer that can abstract useful features from source images and pass them to the decision forests.

The optimal model parameters for training a local DNDF network are those that minimize the following model training loss function:

$$L(\theta, \pi; x, y) = -log(P_T[y|x, \theta, \pi]) \tag{9.7}$$

For the hyper-parameter of decision forests, we list all the settings in the following table 9.2:

Table 9.2: Hyper-parameter settings for Decision Forests layer

NUMBER_EPOCHS	20
TREE_DEPTH	8
NUMBER_TREE	12
FEATURE_RATE	0.75
DROPOUT_RATE	0.05

9.2.3 Evaluation Metrics and Baseline Model

We chose three evaluation metrics and three baseline models to give a complete review. The three metrics reflect three important aspects when deploying ma-

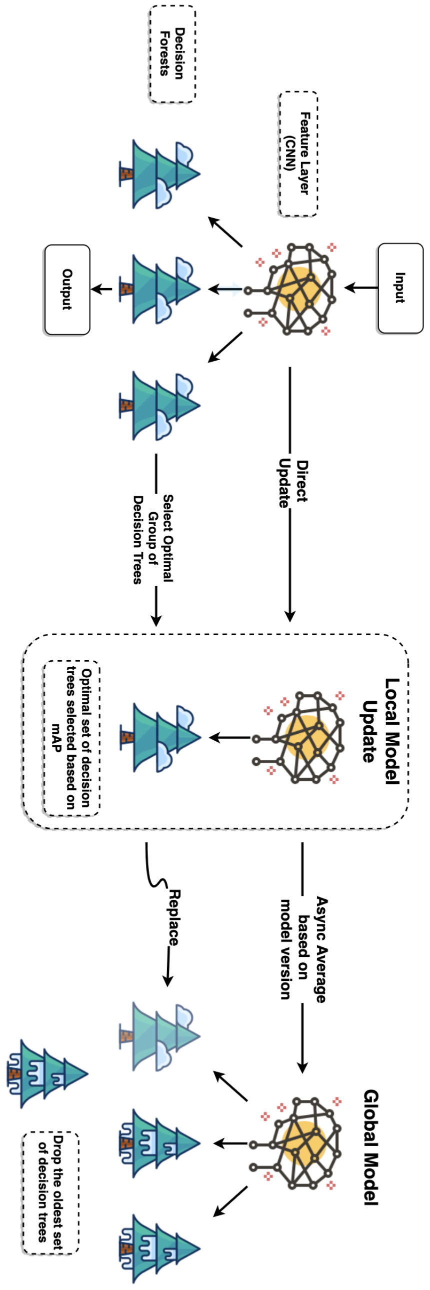

Input
Feature Layer (CNN)
Decision Forests
Output
Direct Update
Select Optimal Group of Decision Trees
Local Model Update
Optimal set of decision trees selected based on mAP
Async Average based on model version
Replace
Global Model
Drop the oldest set of decision trees

Figure 9.5: Local updating and aggregation of the Asynchronous Federated Deep Neural Decision Forests

chine learning methods in an industrial context, namely model performance, model training efficiency and communication cost for data transfer between the edge nodes and the central server. [185].
- Classification Accuracy & mean Average Precision: Classification accuracy and average precision are important metrics that indicate the quality of the classification model. Classification accuracy is defined as the percentage of correctly recognized images among the total number of testing images.

$$AUCC = \frac{T}{T + F} \tag{9.8}$$

where T represents the number of correct classifications while F is the number of false classifications. Average precision summarizes the precision-recall curve as a weighted average of the precision obtained at each threshold. Here the increase of recall from the previous threshold is used as a weight [186][187].

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{9.9}$$

where P_n and R_n are the precision and recall at the nth threshold. We calculate the mean of the Average Precision among all target classes to evaluate the quality of the classifier.
- Model training time: This metric represents the cost of training a model at the edge in terms of time. This metric demonstrates the speed at which the edge vehicles locally update their knowledge, in this case, gained from the machine learning models. The metric is crucial for those systems which need to evolve continuously and adapt rapidly to the changes in data that is caused by changes in the local environment. The metrics were measured in all the vehicles by checking the model deployment timestamp.
- Bandwidth utilization: The total number of bytes transferred during the whole training operation is defined as this metric. This statistic depicts the overall cost of communication resources required to achieve an applicable AF-DNDF model.

The three baseline models include the model trained by applying the centralized learning approach (CL), the independently local learning (IL) and the synchronous Federated Average algorithm (FedAvg). These baseline approaches are commonly used architectures used in federated learning systems and are used to benchmark our proposed AF-DNDF architecture.
Centralized Learning (CL):

The centralized learning approach is used to train this baseline model. All data from edge vehicles is gathered to a single server prior to model training. The hyper-parameters applied are the same as Federated Learning which is mentioned in section 9.2.2.

Independently Local Learning (IL):

Each edge vehicle is directly used to train these baseline models. Unlike Federated Learning, however, throughout the training process, there will be no model exchange between the edge and central nodes. To show how Federated Learning can outperform those individually trained local models, the prediction performance may be compared to the Federated Learning model.

Synchronous Federated Average algorithm (FedAvg):

FedAvg [159], a synchronous Federated Learning aggregation protocol that is frequently used in Federated Learning research, is the method used here. Before performing global model aggregation, the server must wait for all of the participating edge vehicles to finish their local training cycles.

9.3 AF-DNDF: Asynchronous Federated Deep Neural Decision Forests

With the asynchronous aggregation protocol, the vehicle no longer needs to wait for other equipment to complete its local model training iterations. Instead, they can directly send the optimal model to the aggregation server and fetch the global knowledge, which significantly improves the efficiency of global model training. Figure 9.6 illustrates the diagram of the learning procedure in the whole system. In each training cycle, the edge vehicle has to perform local model training and performance validation locally. In order to further save the communication bandwidth, in an AF-DNDF network, each edge nodes will only submit partial decision trees instead of the whole forest. Figure 9.5 demonstrates how an individual edge vehicle updates its optimal local model to the aggregation server.

The decision trees are selected based on mean average precision among all objective categories.

$$mAP = \frac{\sum_{k=1}^{K} AP(k)}{K} \tag{9.10}$$

Here, K represent the total number of classes while k represents a specific class. The optimal group of decision trees will be selected based on the metric and updated to the aggregation server together with the feature layer and form a global knowledge among all participating edge vehicles.

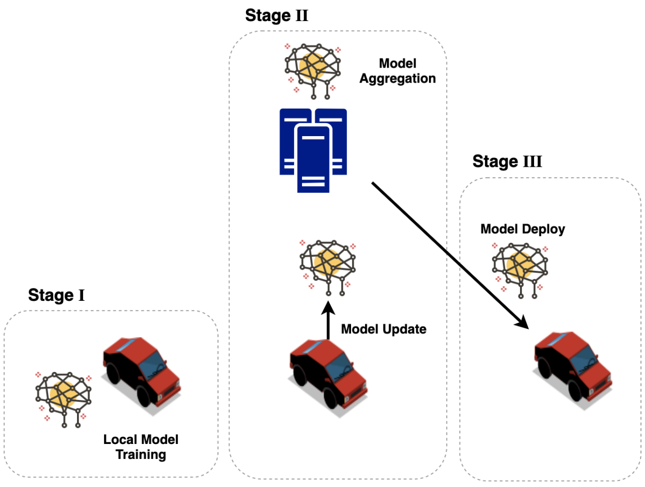


Figure 9.6: Three stages of the Asynchronous Federated Deep Neural Decision Forests

In this paper, as we described in Section 9.2.2, the local model contains 12 decision trees in each vehicle. However, only 4 trees and the local feature CNN layer were updated to the aggregation server. At the end of the iteration, the model will be sent back to the vehicle and continuously enhance the local model.

Figure 9.6 shows the diagram of AF-DNDF updates and aggregation. A detailed description of the three stages within an AF-DNDF system are listed below:

Stage I: Edge vehicles calculate the model locally, then pull the global model version and compare it to their local version value after each local training session. The model version difference computed by ver (global model version) and ver_k (local model version of edge vehicle k) and the frequency bound limitations (a_l and a_u) are used to make the decision. The maximum limit of the model version difference is a_u, whereas the lower limit is a_l. There are three requirements:

- If the local version is out of date (the client version is aged), the edge vehicle should download the most recent model and restart local training.
- If the local version is too near to the most recent version (the client is too active), it should be avoided upgrading and local training should be performed again.
- Clients can then assess the outputs of all decision trees in the forest if the local version is between the higher and lower bounds. After that, the best set of trees is chosen and submitted to the aggregation server, along with the feature CNN layers.

Stage II: To build a global knowledge of all local models, the central server executes aggregation based on the ratio computed by the global and local model versions. The decision forests layer will be replaced by the updated set of local decision trees in the network, while the feature CNN layer will be aggregated using the formula:

$$w_{t+1} \longleftarrow (1-\alpha) \times w_t + \alpha \times w^l_{t+1} \tag{9.11}$$

where w_t represents the global model while w^l_{t+1} is the updated local model. In addition, the ratio α is defined based on model versions:

$$\alpha = \frac{1}{ver_g - ver_l} \tag{9.12}$$

where ver_g is the version of the global model while ver_l represents the updated local model version.
Stage III: The aggregation server updates the global model and sends back the aggregated result (including the incremented model version) to the edge vehicles who request the latest model.

The aggregation server only deploys the global model if the edge vehicles request it as the method is a pull-based algorithm. In terms of aggregation, once the edge vehicles update their own models, the server will aggregate them based on their local version. When it comes to merging global knowledge, the older the model version is, the lower the ratio is. Furthermore, while the model update frequency is entirely dependent on local hardware settings, there are two bound limitations to guarantee that local client update frequencies are within an acceptable range $[a_l, a_u]$. In our paper, based on the number of participating cars, we set the lower frequency bound a_l to 1 and the upper frequency bound a_u to 4. Throughout our experiments, the settings we chose resulted in the best classification accuracy and model training efficiency.

9.4 Results

9.4.1 Classification Performance

We first compared the classification accuracy of our approach with a pure convolutional neural network (CNN) and random forest (RF) to demonstrate the effectiveness of DNDF when encountered the task of object recognition. The hyper-parameter settings are the same as AF-DNDF. (Random forests settings are the same as the values listed in Table 9.2 while the CNN network is demonstrated in Figure 9.4.) The results of the FLIR data set are listed in Table 9.3.

Table 9.3: Classification Accuracy and mean Average Precision of FLIR Dataset

	AF-DNDF	RF	CNN
AUCC	79.9%	67.3%	74.7%
mAP	0.894	0.751	0.828

From the results, we can observe that after combining the decision forests and convolutional neural network, the classification accuracy of AF-DNDF increased 10% compared with RF and about 5% compared with CNN. The situation also applies to the mean average precision, where the value improved by about 10% if the DNDF network is applied.

Table 9.4: Classification Accuracy and mean Average Precision of BDD 100K Dataset

	AF-DNDF	RF	CNN
AUCC	68.6%	49.4%	63.3%
mAP	0.753	0.505	0.705

Table 9.5: Comparison of Classification Accuracy and mean Average Precision with three baseline learning approach in two data sets

		AF-DNDF	FedAvg
FLIR	AUCC	79.9%	80.7%
	mAP	0.894	0.904
BDD 100K	AUCC	68.6%	67%
	mAP	0.753	0.748
		CL	IL
FLIR	AUCC	75.1%	58.4%
	mAP	0.875	0.690
BDD 100K	AUCC	69.8%	60.8%
	mAP	0.766	0.662

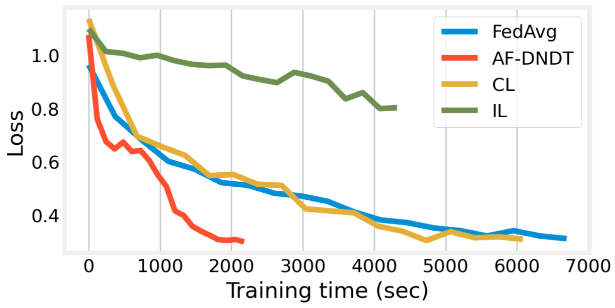
The same conclusion can be obtained with BDD 100K data set. Table 9.4 gives the classifier performance among three different machine learning methods. The classification accuracy can increase around 18% compared with RF and about 5% compared with CNN. For the mean Average Precision, the value can be improved at least 5% with AF-DNDF. The results above demonstrate the effectiveness of DNDF after combining CNN and decision forests when performing object recognition with our data sets.

Moreover, in order to analyze the model classification performance, we compared our model training approach with three baseline learning architectures, namely centralized learning (CL), independently local learning (IL) and synchronous Federated Average algorithm (FedAvg). First of all, Table 11.3 shows the classifier accuracy and mean Average Precision.

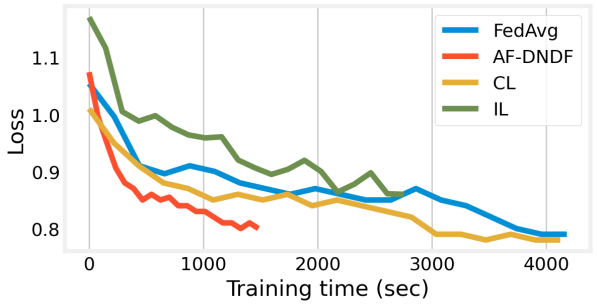
The results show that when compared to centralized learning and synchronous Federated Learning, the AF-DNDF model can achieve the same or even higher levels of accuracy. If we compare it with an independently trained model, the AF-DNDF model can provide a more accurate prediction which is about 20% better with the FLIR data set and 10% better with the BDD 100k data set.

9.4.2 Model Training Efficiency

During the evaluation, in order to simulate heterogeneous hardware settings, we accelerated local model training by Nvidia Tesla T4 GPU in two edge vehicles while another one is trained without hardware acceleration. Figure 9.7 shows the change of the loss value with model training time. The results reveal that the AF-DNDF approach surpasses all baseline approaches in terms of model training efficiency. With the same amount of training time, our method can converge 60% quicker than other existing baseline models.



(a) *BDD100K*



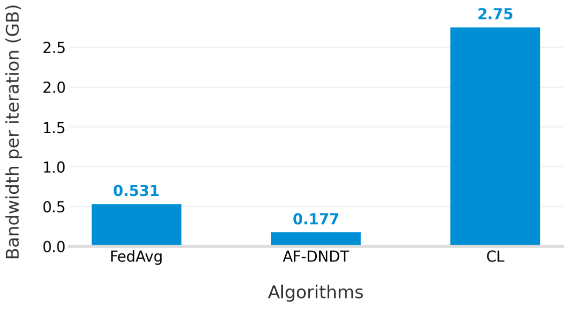
(b) *FLIR*

Figure 9.7: The comparison between model training loss and the model training time with AF-DNDF and three baseline models

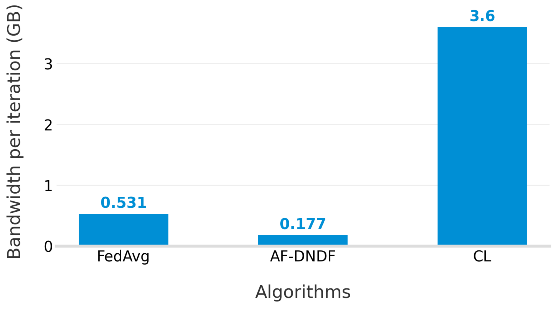
9.4.3 Bandwidth Utilization

During the experiment, we also recorded bandwidth utilization by applying each learning approach. Figure 9.8 shows the results for bandwidth utilization.When compared to the centralized learning technique, the bandwidth cost of both Federated Learning methods is lowered by roughly 80%. Moreover, in our AF-DNDF approach, as our method only selects the optimal part of the model to update, the bandwidth usage is further reduced by about 60% compared with the synchronous Federated Learning algorithm.

In summary, AF-DNDF reduces training time by around 60% and saves bandwidth by about 80% when compared to the centralized learning approach, which is cost-effective and suitable to real-world systems.



(a) $BDD100K$



(b) $FLIR$

Figure 9.8: Bandwidth consumption of different learning algorithm with two data sets

9.5 Discussion

From our experiment results, our architecture of asynchronous Federated Learning of deep neural decision forests proves to have significant advantages compared with commonly used centralized learning and synchronous Federated Learning methods.

Furthermore, the results demonstrate that AF-DNDF requires much less model training time than alternative baseline learning architectures. However, as for synchronous Federated Learning and independently local learning, the vehicle without access to GPU settings becomes a heavy burden among all participated learning vehicles. Other vehicles must wait until all edge vehicles have completed their local training round before performing model aggregation and additional training, which is rigid and time-consuming. These two strategies are even less efficient than the centralized learning method.

Without losing model classification accuracy, our asynchronous method may considerably reduce model training time and bandwidth costs. Our approach not only outperforms synchronous Federated Learning by forming a global knowledge of the entire data sets without requiring any user data sharing, but it also tolerates heterogeneous hardware settings across different edge vehicles, which improves training efficiency significantly. Furthermore, the model quality is considerably improved as a result of the model sharing method and produce much better outcomes.

Because of these benefits, AF-DNDF can be applied in various use cases. The algorithm introduced in this paper can be used in different applications involving object detection on resource-constrained edges, such as camera sensors, mobile phones, and home electrical appliances, in addition to self-driving vehicles. Furthermore, the new aggregation methodology for decision forests and the neural network combination might stimulate further research and increase possible commercial applications.

9.6 Conclusion

In this work, we introduce “AF-DNDF", a new technique for DNDF model training in an asynchronous federated manner. We validate our technique with a real-world case: recognizing road objects in self-driving vehicles. Our findings illustrate the model’s strength and benefits by using the method we suggest. In comparison to the commonly applied centralized learning approach and other Federated Learning architectures, the model can achieve at least equal or even greater prediction accuracy but decreases training time by 60% and bandwidth cost by 80% in our scenario. We highlight that if the number of participating vehicles is raised further, the decrease will be more noticeable, which is cost-effective and suitable to industrial scenarios.

In the future, we would like to add additional edge nodes so that we can assess the system's performance on a broader scale. We also intend to identify more appropriate aggregation protocols to further improve model training efficiency, reduce bandwidth utilization and enhance edge model quality on resource-constrained edge devices in real-world embedded systems.

CHAPTER 10

Autonomous navigation and configuration of integrated access backhauling for UAV base station using reinforcement learning

This chapter has earlier been published as
Autonomous navigation and configuration of integrated access backhauling for UAV base station using reinforcement learning
Zhang H., Li J., Qi Z., Lin X., Aronsson A., Bosch J., and Olsson H. H.
In *2022 IEEE Future Networks World Forum (FNWF)* (pp. 184-189). IEEE.

Like food, water and medicine, the ability to communicate has proven to be an essential tool for first responders, governments, and survivors in disaster response and relief. To provide connectivity in areas that cannot be fully covered by the existing mobile network, for example, when the network infrastructure is damaged or not available, unmanned aerial vehicles (UAVs) carrying base stations (BSs) can be used to provide temporary coverage for users located in the disaster area.

UAV-BS assist wireless communication networks have recently gained increased interest in both academic and public safety communities[188]–[192].

Thanks to the great mobility and flexibility of UAVs, it is expected that UAV-BSs can bring fast connectivity for mission-critical (MC) communications. However, there are a number of challenges that must be addressed when deploying UAV-BSs in practice. The deployment and configuration of the UAV-BSs play a critical role in the performance of the target services. When integrating a UAV-BS into an existing mobile network, a fast and reliable backhaul connection between the UAV-BS and on-ground BSs is required to ensure the end-to-end quality of service (QoS) for the interested users. In addition, reliable and scalable backhaul links between different UAV-BSs are needed when multiple UAV-BSs are used to cover a wider area. Therefore, it is crucial to ensure the good quality of both the access and backhaul links when optimizing the deployment of UAV-BSs. The deployment optimization also depends on many other factors such as the limitations on UAV's flying altitude, operation time, antenna capabilities and transmit power, the network traffic load distribution, and user movements.

While many works on UAV-BS deployment focused on the problems of UAV placement, trajectory design, and number of UAV-BSs, etc., only a few previous results have considered the wireless backhaul aspects[193]–[196]. In [193], the authors investigated how to rapidly deploy the minimum number of UAV-BSs to assist the existing mobile network to evenly serve as many users as possible while guaranteeing a robust wireless connection among the UAV-BSs and fixed on-ground BSs. It is assumed that all UAV-BSs are flying at the same and fixed height, and the robustness of the backbone network among the deployed UAV-BSs is guaranteed by ensuring a bi-connection network topology so that if one UAV-BS fails, there still exists at least one route between any UAV-BS and a fixed on-ground BS. In [194], a UAV-BS 3-D placement algorithm is proposed to maximize the total number of served users or the sum of user data rates subject to capacity constraints of both access and backhaul links. This work was further extended by the authors in [195], where a mixed-integer non-linear programming approach is proposed to jointly optimize a UAV-BS location and the system bandwidth allocation without exceeding the backhaul and access capacities.

With 5G new radio (NR), there is an opportunity to use the integrated access and backhaul (IAB) feature to wirelessly connect multiple UAV-BSs and integrate them to an existing mobile network seamlessly [188], [196]. The NR IAB feature supports multi-hop wireless backhaul with a flexible and adap-

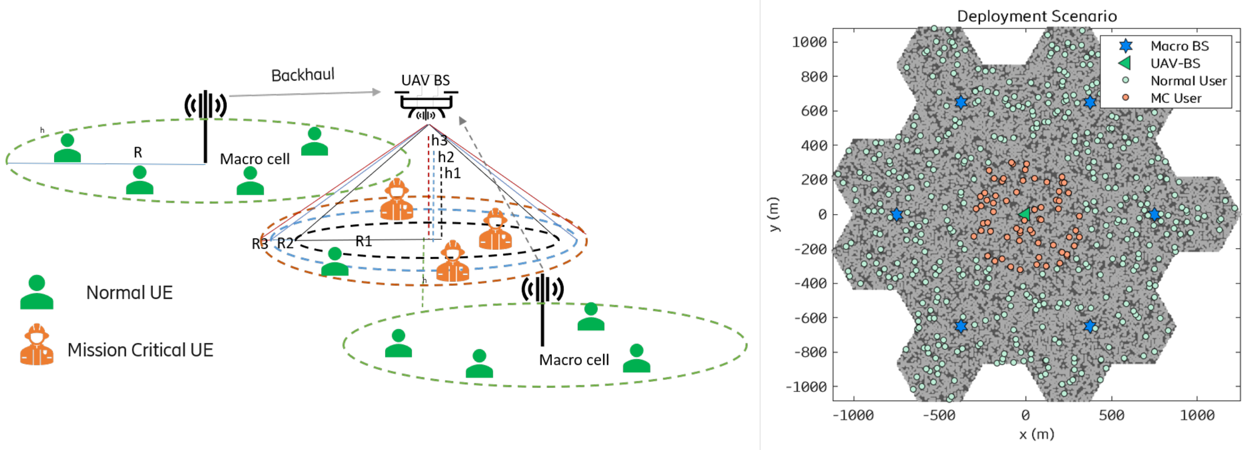


Figure 10.1: A UAV-BS assisted wireless network deployment using IAB

tive network architecture [197]. Figure 1 illustrates an example of UAV-BS-assisted network deployment using IAB. A macro-BS with a wired connection to the core network is configured as an IAB donor node, and a UAV-BS is configured as an IAB node. The UAV-BS connects to a parent or donor node using wireless backhaul, and it services on-ground users using access links. The IAB network topology can adapt to the varying backhaul link conditions and traffic load situations.

In [196], the authors evaluated the mean user throughput and user fairness performance of a UAV-based IAB system in millimeter-wave (mmWave) urban deployments, where the UAV-BS 2-D location is optimized to follow the user movement using a particle swarm optimization method. They assumed separate channels for access and backhaul links as well as dedicated antenna arrays for each interface. In this work, we consider a UAV-BS-assisted IAB network for providing temporary coverage to MC users in an emergency area. We assume that the system operates in a mid-band, which provides better coverage than mmWave bands. The same frequency band and the same antennas are shared between access and backhaul links to reduce the cost and weight of the BS carried on the UAV. We investigate how reinforcement learning (RL) can support autonomous navigation and configuration of IAB for UAV-BS-assisted networks. A framework and signalling procedure are proposed to support applying RL in an IAB network architecture. In addition, an RL algorithm is designed to jointly optimize the antenna configuration and the 3-D location of the UAV-BS to best serve on-ground MC users while maintaining a good backhaul connection. System-level simulations are performed to gain insights into the impact of different optimizing parameters on the considered system performance, i.e., the throughput and drop rate of MC users. The simulation data has also been utilized for the RL algorithm design and validation.

10.1 Use Case and System Model

We consider a multi-cell mobile cellular network as illustrated in the right plot of Figure 10.1. The network initially consists of seven macro-BSs. However, due to, for example, a natural disaster, the macro-BS located in the middle of the network map is damaged. Hence, a UAV-BS is temporally set up to provide wireless connectivity to the MC users in the disaster area (a circle area with a 350 m radius in the middle of the deployment map). The UAV-

BS is modelled as an IAB node. To reduce the complexity and weight of antennas put on the UAV-BS, we assume that the same antennas are used for wireless access and backhaul links. The UAV-BS measures the wireless links to the six functioning macro-BSs, and it dynamically selects one of these macro-BSs that gives the best link quality as its donor node. Then, a wireless backhaul link is established between the UAV-BS and the selected donor-BS. Both normal users and MC users are allowed to access the UAV-BS. A user selects its serving BS (a macro-BS or a UAV-BS) based on the end-to-end wireless path quality.

It is assumed that all macro-BSs and the UAV-BS have three sectors each, and they operate at the same carrier frequency of 3.5 GHz with a time division duplex (TDD) pattern that consists of four time slots, i.e., downlink (DL), DL, uplink (UL) and DL. The pattern is repeated with a periodicity of 2 ms. The 100 MHz total system bandwidth is shared between the access and backhaul links. To reduce the complexity and mitigate interference, we further assume that the UAV-BS operates in a half-duplex mode, i.e., it cannot transmit and receive signals simultaneously. The UAV-BS's flying height is assumed to be below 35 meters so that the rural macro propagation model can be reused for UAV-BS in this case.

Users are randomly dropped in the map shown in Figure 10.1. In each time slot, a number of users are activated following a dynamic traffic model with a predefined traffic arriving rate and a predefined average traffic size. The DL and UL traffic of activated users are scheduled based on the access and backhaul link quality, the network scheduling strategy, and the allowed transmission directions at a given time slot at each BS. The throughput of each served user is calculated based on its served traffic size and the time used for delivering the traffic. Note that for a user connected to the UAV-BS, its throughput depends not only on the access link between itself and the UAV-BS but also on the wireless backhaul link between the UAV-BS and the donor-BS. A user will not be served with more traffic than required, and a user can also be dropped/blocked in case of poor link quality or insufficient radio resources. User throughput and drop rate are the key performance indicators considered in our RL algorithm design.

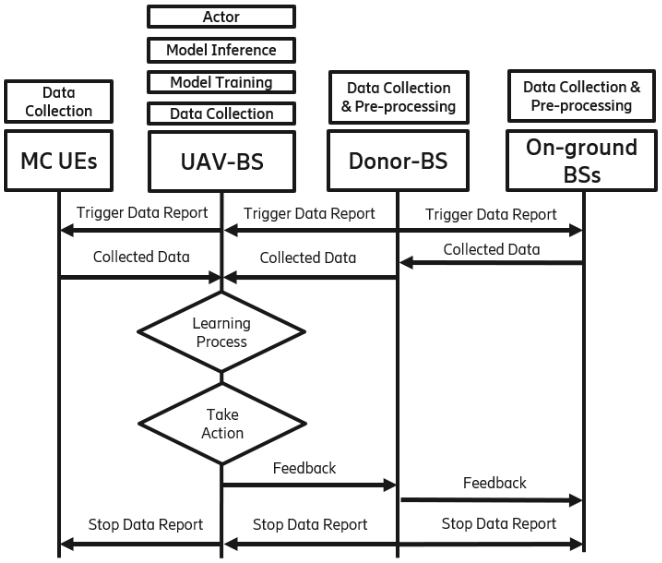


Figure 10.2: Framework and signaling procedure

10.2 Framework and Signaling Procedure

In this section, we propose a signalling procedure to support applying RL to the considered use case. The functional framework studied in 5G NR for radio access network intelligence is used as our reference [198].

The blocks above each entity (UE or BS) shown in Figure 10.2 denote different machine learning functionalities, including data collection, model training, model inference and actor. Data collection is a function that is responsible for collecting input data for model training and model inference functions. The model training function performs the training of the learning model while the model inference function provides the learning output. Finally, the actor function receives the output from the model inference module and triggers or performs corresponding actions. Figure 10.2 shows the proposed signalling procedure, which consists of the following key steps:

1): Data requests triggered by a donor BS: After a UAV-BS completes its network integration procedure, the donor-BS triggers data collection to assist

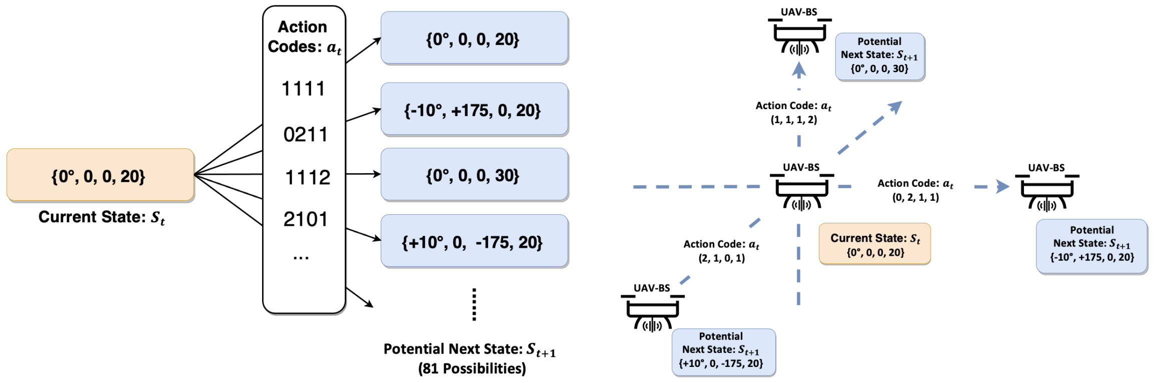

{0°, 0, 0, 20}
Current State: S_t
Action Codes: a_t
1111
0211
1112
2101
...
{0°, 0, 0, 20}
{-10°, +175, 0, 20}
{0°, 0, 0, 30}
{+10°, 0, -175, 20}
Potential Next State: S_{t+1}
(81 Possibilities)
UAV-BS
Potential Next State: S_{t+1} {0°, 0, 0, 30}
Action Code: a_t (1, 1, 1, 2)
UAV-BS
Action Code: a_t (0, 2, 1, 1)
UAV-BS
Potential Next State: S_{t+1} {-10°, +175, 0, 20}
Current State: S_t {0°, 0, 0, 20}
Action Code: a_t (2, 1, 0, 1)
UAV-BS
Potential Next State: S_{t+1} {+10°, 0, -175, 20}

Figure 10.3: Example of the state transition from current state $\{0^\circ, 0, 0, 20\}$

the UAV-BS in optimizing its configuration and deployment by sending a data request message to the relevant users and BSs.

2): Data collection at UAV-BS: After receiving the data request, MC users, the donor-BS, and related on-ground BSs will send the requested data to the UAV-BS. The data collection procedure can include: a) data collected by the UAV-BS itself, i.e., from its connected MC users, radio measurements, and onboard sensors. b) data is firstly reported from a set of users and a set of on-ground BSs to the donor-BS and then forwarded to the UAV-BS.

3): Learning process in the UAV-BS: The UAV-BS processes the collected data. And the model training and inference functions are initiated using the processed data. After training, a set of ML-related parameters are generated corresponding to the trained model. Then, the deployment strategy and configuration are developed based on the output of model inference.

4): The UAV-BS takes action: The actor function in the UAV-BS performs antenna tilt and location adjustment based on the output of the model inference.

5): The UAV-BS sends feedback to its donor-BS, who can then forward the feedback or action recommendations to related on-ground BSs.

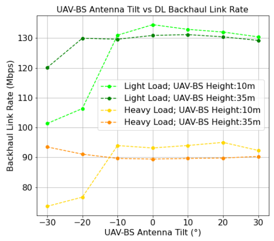
6): The donor-BS and/or the related BSs adjust their configuration (e.g., antenna tilt, transmit power, etc.), using the feedback from the UAV-BS as input data. Finally, the donor-BS stops requesting data.

Steps 2)-6) can repeat till certain criteria are fulfilled. The donor-BS then can stop the learning process by sending a stop data reporting message to its connected users and BSs.

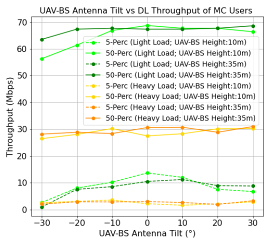
10.3 Reinforcement Learning Algorithm Design

In this section, we design an RL algorithm to jointly optimize the access and backhaul antenna tilt value and the 3-D location of the UAV-BS in the considered scenario. We use a deep Q-Network as our base algorithm [199].

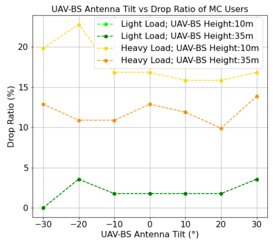
The algorithm is modified and implemented to solve our system optimization problem.



(a) Antenna Tilt vs Backhaul Link Rate



(b) Antenna Tilt vs Throughput for MC Users



(c) Antenna Tilt vs Drop Rate for MC Users

Figure 10.4: UAV-BS antenna tilt's impact on backhaul link rate, MC user throughput and MC user drop rate

10.3.1 Algorithm Environment

10.3.1.1 State Space

A UAV-BS's state at a given time instance t has four dimensions and it is denoted as $s_t = \{\sigma_t, x_t, y_t, z_t\}$, where σ_t represents the electrical tilt value of the access and backhaul antenna, and $\{x_t, y_t, z_t\}$ denotes the 3-D location of the UAV-BS at time t. The candidate values for the x and y axis are $[-350, -175, 0, +175, +350]$ meters, which covers the disaster area shown in Figure 10.1. The candidate values of z axis are $[10, 20, 30, 35]$ meters. The candidate antenna tilt values are $[-30, -20, -10, 0, +10, +20, +30]°$, where a positive tilt value means applying an electrical down-tilt to the access and backhaul antenna, and a negative tilt value maps to applying an electrical up-title to the antenna.

10.3.1.2 Action Space

For each state dimension, the UAV-BS can select an action out of three candidate options. These three alternative action options are coded by three digits $\{0, 1, 2\}$, where "0" denotes that the UAV-BS reduces the status value by one step from its current value; "1" represents that the UAV-BS does not need to take any action at this state dimension and it keeps the current value; and "2" means that the UAV-BS increases the status value by one step from its current value. For instance, if the UAV-BS is at the space point where the value of the x dimension is equal to 0 meter, then, an action coded by "0" for this dimension means that the UAV-BS will select an action to reduce the value of x axis to -175 meters, an action coded by "1" implies that the UAV-BS will hold the current value of x axis (0 meter), and an action coded by "2" implies that the UAV-BS will increase the value of x axis to 175 meters. The same policy is used for all the dimensions of the state space. Since one state has four dimensions and each state dimension has three action options, the action pool contains in total 81 action candidates that can be programmed to a list of $[0000, 0001, 0002, 0010..., 2222]$. Hence, at a given time t, the UAV-BS can select an action a_t from these 81 candidates. Figure 11.2 demonstrates an example of state transition from a given state $s_t = \{0°, 0, 0, 20\}$.

10.3.1.3 Monitoring Feature Metrics and Reward Function

For MC communications, it is more important to serve as many MC users as possible with adequate service quality rather than maximizing the peak rate of a subset of MC users. Hence, for the reward function design of the RL algorithm, we have chosen six key feature metrics to reflect the overall quality of service for MC users, including:

- The drop rates of MC users for UL and DL (β_{ul}, β_{dl}), which reflect the percentage of unserved MC users.
- The 50-percentile throughput values of MC users for both UL and DL $(\alpha_{ul-50\%}\alpha_{dl-50\%})$, which represent the average performance of the MC users, and
- The 5-percentile throughput values of MC users for both UL and DL $(\alpha_{ul-5\%}, \alpha_{dl-5\%})$, which represent the "worst" performance of the MC users.

To balance these key performance indicators, the reward function is designed as a weighted sum of these six feature values as follows. All features are normalised within the range $[0, 1]$ before model training.

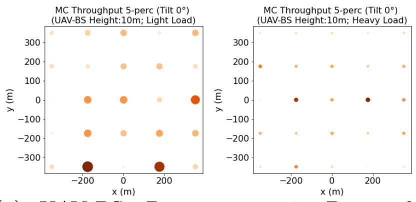
$$\begin{aligned} R_s = &\omega_1 \times \frac{(1-\beta_{dl}) + (1-\beta_{ul})}{2} + \omega_2 \times \frac{(\alpha_{ul-5\%} + \alpha_{dl-5\%})}{2} \\ &+ \omega_3 \times \frac{(\alpha_{ul-50\%} + \alpha_{dl-50\%})}{2} \end{aligned} \tag{10.1}$$

In addition, we set $\omega_1 + \omega_2 + \omega_3 = 1$ to normalise the reward value such that R_s is between $[0, 1]$. The weights assigned to each metric signify the system's relevance. In our scenario, we placed higher weights on user drop rates and 5-percentile MC-user throughput features in order to serve and guarantee acceptable service to all MC users. Thus, the weight values used in our algorithm are $\omega_1 = 0.5$, $\omega_2 = 0.3$ and $\omega_3 = 0.2$.

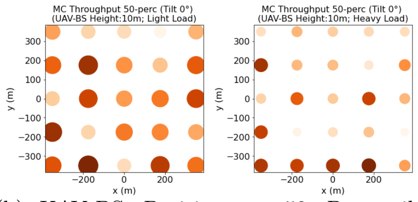
A deep Q-network is used as our base algorithm to achieve better self-control decisions for the autonomous UAV-BS. At each training episode, the UAV-BS explores the state space and performs Q-value iterations. An ϵ-greedy exploration is applied when determining the action to take at the next time instance. The probability of exploration is given by parameter ϵ. The exploration probability specifies the likelihood that the agent will execute state exploration and choose actions at random. Otherwise, the agent will perform the action that is believed to yield the highest expected reward. The data for each training step is stored in a replay batch $\mathcal{D}$. Specifically, each row of $\mathcal{D}$ contains the tuple (s_t, a_t, r_t, s_{t+1}), namely, current state, action, reward and next state. Samples will then be randomly selected and used for Q value model updating.

10.4 Performance Evaluation

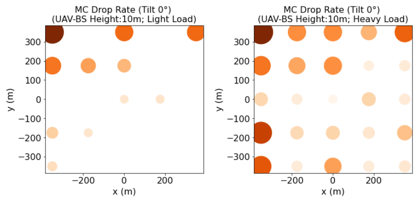
In this section, we firstly investigate the impact of the antenna configuration and 3-D location of the UAV-BS on the performance of MC users in terms of throughput and drop rate by using system-level simulations. Then, we evaluate the performance of the proposed RL algorithm using the data generated from the simulator.



(a) UAV-BS Position vs 5 Percentile Throughput for MC Users



(b) UAV-BS Position vs 50 Percentile Throughput for MC Users



(c) UAV-BS Position vs Drop Rate for MC Users

Figure 10.5: Impact of UAV-BS position on MC user throughput and drop rate

10.4.1 System Performance Analysis

We consider the system model discussed in section II. It is assumed that all BSs and users have two transmitting and two receiving antennas. The maximum allowed transmit powers for a macro-BS, a UAV-BS and a user are configured as 46, 40, and 23 dBm, respectively. Due to space limitations, in this subsection, we only discuss the DL performance results, considering two different levels of traffic load in the system (light and heavy load cases with different user arriving rates). However, both DL and UL metrics discussed in section III have been used when evaluating the proposed RL algorithm.

To gain insights into the impact of UAV-BS antenna tilt and flying height on the considered performance metrics, we fix the UAV-BS's 2-D position in the center of the deployment map and investigate the system performance in terms of the backhaul link rate, DL throughput and drop rate of MC users for different traffic load levels, as shown in Figure 10.4. It can be seen from Figure 10.4(a) that both UAV-BS antenna tilt and height have a significant impact on the backhaul link rate, while UAV-BS height has a bigger impact in the heavy load case. Another observation is that the network has a lower backhaul link rate in case of heavy load. This is because of the increased interference levels, and also the UAV-BS needs to share the resource with the on-ground users connected to its donor-BS (in-band IAB operation). Hence, increasing traffic load means that more users will share the radio resource with the UAV-BS for its backhaul traffic delivery.

From Figure 10.4(b), we see that the impact of tilt values on the throughput performance in the light load case is more visible, while for the high load case, the impact of tilt values is negligible. Meanwhile, UAV-BS flying height has no significant impact on the throughput in the considered UAV-BS 2-D location. Figure 10.4(c) shows that both antenna tilt and UAV-BS height significantly impact drop rate performance. If the system load is light, the curve for the UAV-BS height at 10 m overlaps with the curve for the case of 35 m. This implies that in the current context, UAV-BS height has no discernible effect on the drop rate of MC users, while UAV-BS antenna tilt plays a more prominent role. For the heavy load situation, the results imply that flying the UAV-BS at a higher altitude can reduce the drop rate.

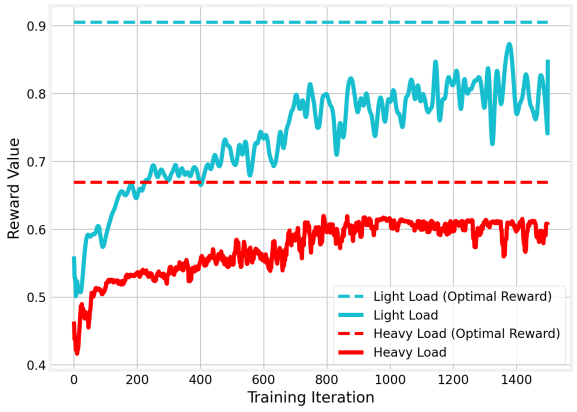


Figure 10.6: Reward value with number of training iterations in two different network traffic load scenarios

Figure 10.5 shows the impact of UAV-BS's 2-D position (x-axis and y-axis) on the considered performance metrics when the UAV-BS antenna tilt is fixed to 0° and its flying height is fixed to 10 m. The size of the circles shown in Figures 10.5(a), 10.5(b) and 10.5(c) represents the exact value of

Table 10.1: Comparison of the average value of the six system performance metrics during UAV-BS deployment in two load scenarios

	DL 50‰ Through-put	DL 5‰ Through-put	DL Drop Percent-age	UL 50‰ Through-put	UL 5‰ Through-put	UL Drop Percent-age	Reward Value
Optimal State (Light Load)	106.11	26.03	0%	7.73	2.18	2.4%	0.905
Reinforcement Learning (Light Load)	105.47	25.56	0.1%	7.7	2.15	2.6%	0.865
Optimal State (Heavy Load)	35.5	3.51	8.9%	4.68	0.15	2%	0.669
Reinforcement Learning (Heavy Load)	58.4	1.55	12%	5.08	0.1	3%	0.617

the 5-percentile throughput, 50-percentile throughput and drop rate of MC users, respectively. The larger the size is, the higher value is for a considered performance metric. To make it easier to identify the 2-D location that gives the highest value of a considered performance metric, in each subplot of Figure 10.5, we use different colours of a circle to represent a relative value of the considered performance metric. The darker the colour is, the higher value of the performance metric is.

It can be seen from Figure 10.5 that for a given performance metric, e.g., 5-percentile MC user throughput, the optimal UAV-BS 2-D location changes when the network traffic load level changes. We can also observe that the optimal UAV-BS position is close to the edge instead of the center of the MC area. This is because the UAV-BS can keep a good backhaul connection with its donor-BS at the edge of the MC area. In addition, we see that the optimal UAV-BS position is different when considering different performance metrics, e.g., maximizing the 5-percentile MC user throughput, maximizing the 50-percentile MC user throughput, or minimizing the MC user drop rate. Therefore, the weights selected for different performance metrics in the reward function will impact the optimal location of the UAV-BS.

10.4.2 Reinforcement Learning Performance Evaluation

In this section, we present the results obtained through our RL algorithm proposed in Section IV. We compare the result of our algorithm with the global optimal state, which is obtained by using grid search through the whole data set. We show the benefits of using the proposed algorithm and its ability to adapt to the changing wireless environment after model training. During the training, the learning rate is set to 5×10^{-5}, the initial starting exploration probability ϵ is set to 1 while the exploration decay is 0.995 and the number of training iterations equals 1500.

We first analyze the convergence of the algorithm during the model training. Figure 11.5 illustrates the reward value as a function of the number of training iterations. We can observe that, for both the light load and heavy load cases, the proposed algorithm can learn from the history and eventually approach the optimal state that gives the largest reward value in both load scenarios. The same conclusion can also be made by comparing the reward value column of Table 11.3.

Table 11.3 also shows the performance of the considered six feature metrics during the UAV deployment. Each deployment contains 100 steps for a UAV-BS to make decisions and take action. We can observe that based on past experience and a well-trained learning model, the proposed algorithm can quickly configure and navigate the UAV-BS to optimize the considered performance metrics. Only a limited number of steps are needed for the UAV-BS to reach a stable state. As shown in Table 11.3, the reward value (a weighted sum of the six considered performance metrics) achieved by the proposed RL algorithm is only about 4% to 5% less than that provided by the global optimal solution for the light load and high load scenarios, respectively. Since the reward value summarizes the overall system performance metrics during the UAV-BS deployment, our results demonstrate the strength of the algorithm and the ability to provide fast connectivity to MC users in different traffic load scenarios.

10.5 Conclusions and Future Work

In this paper, we developed an RL algorithm to autonomously configure and navigate a UAV-BS to provide temporary coverage for MC users. The UAV-BS is connected to an on-ground donor BS and integrated into an existing mobile network using the 5G IAB technology. A functional framework and signalling procedure are proposed to support data collection, model training and decision making for the considered use case. An action encoding strategy is introduced to represent UAV-BS decisions with multiple state dimensions, including the 3-D space location and the access and backhaul antenna electrical tilt. Our results demonstrate the benefits and efficiency of our proposed algorithm in different traffic load scenarios. The algorithm can help a UAV-BS quickly find the optimal 3-D location and its antenna configuration to provide a stable connection to MC users.

In the future work, we will further investigate various hyper-parameter and reward function combinations based on different service requirements. We will also investigate various frameworks and signaling procedures to support the application of centralized or distributed machine learning for the considered use case.

CHAPTER 11

Deep Reinforcement Learning in a Dynamic Environment: A Case Study in the Telecommunication Industry

This chapter has earlier been published as
Deep Reinforcement Learning in a Dynamic Environment: A Case Study in the Telecommunication Industry
Zhang H., Li J., Qi Z., Lin X., Aronsson A., Bosch J., and Olsson H. H.
In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 68-75). IEEE.

11.1 Introduction

Reinforcement Learning (RL) is a machine learning method that learns a model for mapping the present circumstance to an action that maximizes the numerical payoff signal reward. The agent will not be told what actions should be taken but discover by itself through trial and error. The agent will then understand which action may offer the highest reward [60].

This type of learning approach differs from the commonly used supervised and unsupervised learning approaches in the field of machine learning, where supervised learning is performed using a training set with annotations provided by an external supervisor and unsupervised learning is typically a process of determining the implicit structure in data without annotations [200]. Reinforcement learning poses a unique problem in that it requires the intelligence to leverage previous experience while also undertaking explorations that enables better action choices in the future.

With the rise of this wave of artificial intelligence, the introduction of AlexNet [156] in 2012, and AlphaGo's victory over Lee Sedol in the game of Go [201], reinforcement learning has regained the attention of academia and industry. However, when applied in a real-world scenario, reinforcement learning is usually demonstrated to be fragile and unable to generalize to diverse contexts. To the best of our knowledge, the majority of reinforcement learning research places the algorithm in a game environment or assumes a fixed running space [64][202]. However, since the real world is a complicated dynamic system rather than a static environment, the reinforcement learning algorithm's practicality must also be addressed in real-world applications. As most reinforcement learning models are built on a static environment, our proposed method allows industry to apply the methods into real-world embedded systems.

Controlling a UAV is a critical topic that must be addressed if telecommunications companies want to ensure continuous operation in an emergency scenario. The goal of this research is to assist the industry in using reinforcement learning algorithms in real-world scenarios. In this paper, we offer a novel dynamic reinforcement learning approach for adapting to complicated industrial environments. We apply and validate our approach to a telecommunications use case. A realistic situation is chosen in which we simulate the wireless base station and drone characteristics in great detail. And because user movements cause constant change, the throughput and drop rate values will also change over time. In this context, the reinforcement learning algorithm will autonomously configure and control an unmanned aerial vehicle base station (UAV-BS) for robust connectivity to the mission-critical user equipment. The contributions are as follows:
1). We propose a novel dynamic reinforcement learning algorithm to monitor and adapt reinforcement learning exploration rate when deploying into the production environment.

2). We verify the algorithm with a continuous environment in order to explore if the algorithm can adapt to the dynamic environment and maintain high-quality service performance.
3). We validate the algorithm in an empirical environment by including mission-critical user movements.

The remainder of this paper is structured as follows. In Section II, we introduce the background of this study. Section III details our research method, including validation case and environment, the simulation testbed and the utilized learning method. Section IV presents the novel algorithm utilized in this paper. Section V evaluates our proposed learning method to empirical data sets. Section VI outlines the discussion on our observed results. Finally, Section VII presents conclusions and future work.

11.2 Background

The concept of a reinforcement learning algorithm is straightforward: the agent is reinforced to make better decisions based on the past learning experience. This method is similar to the different performance rewards that we encounter in everyday life [203].

Reinforcement learning is a machine learning method for comprehending and automating goal-directed learning and decision issues. It stresses that an intelligence learns through direct interaction with its environment, without the use of imitable supervised signals or comprehensive modelling of its surroundings, and hence has a different paradigm than other computational techniques [200]. The algorithm defines the process by which a learning intelligence interacts with its environment using a formal framework of Markov decision processes. In the algorithm, the agent and the environment are the two main objects that can be interacted with reinforcement learning. The agent detects the state of the environment and learns to choose a suitable action based on Reward feedback in order to maximize overall long-term gain. The environment receives a series of actions and returns a quantifiable signal to the agent.

Most reinforcement learning research places the algorithm into static running spaces. The learning method has been successfully researched and applied in gaming environments, such as Atari [203], Go game [201], etc. There are also some empirical applications that applied deep reinforcement learning.

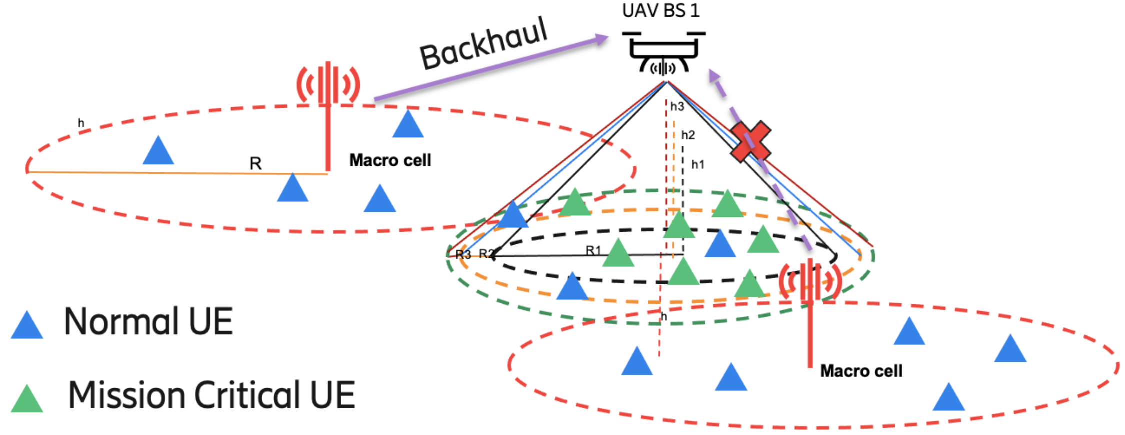


Figure 11.1: Case diagram: With the movement of the traffic on the ground, reinforcement learning algorithm should control UAV-BS for better serving the traffic as time goes. The controls are made by adjusting the three-dimensional space location and electrical antenna tilt.

However, due to the complicated industrial environment, the assumption is not able to aid in the deployment of a reinforcement learning algorithm into the real-world embedded systems [204]. The problem then becomes a critical challenge when practically implementing reinforcement learning. To help RL adapt to environmental changes and improve its robustness, a continuous exploration and training strategy can thus be developed. In this study, we propose a novel dynamic reinforcement learning method to address this problem. The algorithm is capable of rapidly evolving and adapting to complex industrial situations.

11.3 Dynamic Environment

The goal of this research is to assist the industry in using reinforcement learning algorithms in real-world scenarios. The classic RL's fundamental drawback is its low performance in a changing environment. Typically, if the environment has changed (the environmental values observed by the agent have changed), the agent must retrain the entire algorithm to catch up with the environmental changes [204][205]. However, to the best of our knowledge, limited work has been done to help the algorithm in adapting to changing environments, and they exclusively focus on relatively static space where the observation data are rarely modified. The dynamic environment indicates that the major elements that the agent observed will vary substantially over time.

In the real-world context, it differs when we compare a gaming environment to our actual scenario. The observation at the same location in a gaming scenario will not change or has restricted possibilities throughout time. In a go game, for example, the same spot only has three choices (black, white, or none), or in a moon landing game, the situation is fixed since the observation is a fixed value at each image pixel. In our scenario, however, even at the same position, the observation value (e.g. throughput, drop rate) will be constantly changing.

In our situation, user movement would significantly affect throughput and drop rate values. Thus, a triggering strategy for adaptation and investigation is a vital component if companies want to ensure high service quality. In our work, we demonstrated the effectiveness of our adaptability technique and the efficiency of combining previously trained RL models.

11.4 Method

The analytical technique and research method mentioned in [92] were used in this study to conduct quantitative measurement and validation on deep reinforcement learning. The article presents recommendations for applying machine learning methods to software engineering activities, as well as methods for demonstrating how these can be conceived as learning problems and addressed in terms of learning algorithms. The mathematical notations, testbed configuration and the base algorithm used in our research are presented in the following sections.

11.4.1 Mathematical Notations

The mathematical notations that will be used in the paper are introduced here first:

$\alpha_{dl-i\%}$	$i\%$ percentile Downlink throughput for Mission-Critical User Equipments
$\alpha_{ul-i\%}$	$i\%$ percentile Uplink throughput for Mission-Critical User Equipments
β_{dl}	Downlink percentage of dropped and blocked Mission-Critical users
β_{ul}	Uplink percentage of dropped and blocked Mission-Critical users
S	State Space
A	Action Space
R	Reward Function for UAV control
x	UAV Drone X-axis value
y	UAV Drone Y-axis value
z	UAV Drone Z-axis (Height) value
σ	UAV Drone Electric antenna tilt degree

11.4.2 Validation Case

Stable connectivity is crucial for improving situational awareness and operational efficiency in various mission-critical situations. In catastrophe or emergency scenarios, the existing cellular network coverage and capacity in the emergency area may not be available or sufficient to support mission-critical communication needs [188]. In these scenarios, deployable-network technologies like portable base stations (BSs) on unmanned aerial vehicles (UAVs) or trucks can be used to quickly provide mission-critical users with dependable connectivity.

In this paper, we consider a mission-critical scenario shown in Figure 11.1, where a macro BS is damaged due to natural disasters and a UAV-BS is set up to provide temporary wireless access connection to mission-critical users that are performing search and rescue missions in the disaster area. The UAV-BS is integrated into the cellular network by connecting itself to an on-ground donor BS (e.g., a macro-BS) using wireless backhaul. The same antenna hardware is used for both the access and the backhaul links.

In a multi-network situation that includes both established BSs on the ground and temporarily deployed UAV-BS, the deployment and configuration of the UAV-BS are crucial to ensuring the performance of the target users/services, such as the mission-critical users. It may also have an effect on the overall performance of the system. Because the UAV-BS is linked to the core network through wireless backhaul, it is critical to guarantee that both the backhaul and access lines are of high quality while executing this system optimization. The limits imposed by the UAV's flight height, antenna capabilities, and other crucial elements contribute to the optimization difficulty. The best solution is determined by a variety of criteria such as network traffic load distribution, quality of service (QoS) needs, user mobility, transmit power, and antenna settings at the time of deployment. As a result, jointly optimizing these UAV-BS parameters is a difficult system-level optimization issue that must be handled in a dynamic environment.

In order to best serve the on-ground mission-critical users, and, at the same time, maintain a good backhaul connection, we apply reinforcement learning to configure the three-dimensional space location of the UAV-BS and the electrical tilt for the access and backhaul antenna of the UAV-BS. More importantly, the decision of the reinforcement learning algorithm should also be able to adapt to the real-time changing environment (e.g., when mission-critical traffic moves on the ground), where traditional reinforcement learning algorithms are not applicable and would result in inappropriate UAV-BS configuration decisions.

11.4.3 Case Environment

11.4.3.1 State Space

As we described in the validation case section, in order to better serve the traffic as time goes by, the controls of a UAV-BS were made by adjusting the three-dimensional location in space and the electrical antenna tilt. A state of a drone is represented by $\{\sigma, x, y, z\}$ where σ represents the tilt degree of the drone electric antenna while $\{x, y, z\}$ represents the three-dimensional space location. In our simulation, the range of the x, y axis was restricted in list $[-350, -175, 0, +175, +350]$ meters while z axis values belonged to list $[10, 20, 30, 35]$ meters. In addition, the range of the tilt values belonged to list $[-30, -20, -10, 0, +10, +20, +30]^{\circ}$. Since we observe obvious feature value changes in those location and tilt points, the states mentioned above were included and used for algorithm training and validation.

11.4.3.2 Action Space

There are three different options for each state dimension. The drone can either decrease, increase state value by one step or hold in the same position. For example, if the drone is located at the space point where the x axis value equals -175, the drone can then select the action which can either increase the x-axis value to 0, decrease the value to -350 or hold the value and stay at value -175. The policy is the same and applies to all four state dimensions.

In addition, in order to help the drone understand the actions and process the data, three action options were coded by using digits $0, 1, 2$ where 0 represents decreasing the value by one step, 1 represents holding the current value and 2 represents increasing value by one step. As each dimension can choose from three options, there are 81 action combinations in the action pool and can be coded to list $[0000, 0001, 0002, 0010..., 2222]$. Figure 11.2 illustrates an example for the UAV-BS transit from one state to another by selecting different action combinations.

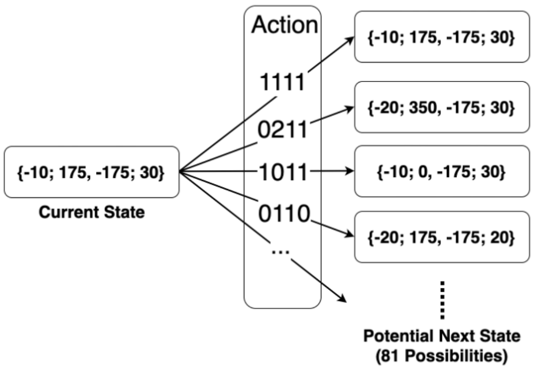


Figure 11.2: Example of the state transition from current state {-10, 175, -175, 30}

11.4.3.3 Reward Function

We now describe the reward function used in our model. Six crucial features were selected as the metrics to monitor the quality of the connection to the mission-critical user equipment, including uplink and downlink percentage of dropped and blocked Mission-Critical users (β_{ul}, β_{dl}), 50 percentile uplink and downlink throughput ($\alpha_{ul-50\%}, \alpha_{dl-50\%}$) and 5 percentile uplink and downlink throughput ($\alpha_{ul-5\%}, \alpha_{dl-5\%}$). Those features can demonstrate if the UAV-BS can provide robust and high-quality connections to the mission-critical user equipment. Hence, we constructed a weighted combination of those features in the reward function as follows:

$$\begin{aligned} R_s = &\kappa \times (2 - \beta_{dl} - \beta_{ul}) + \mu \times (\alpha_{ul-5\%} + \alpha_{dl-5\%}) \\ &+ \eta \times (\alpha_{ul-50\%} + \alpha_{dl-50\%}) \end{aligned} \tag{11.1}$$

Since connection services are provided to the mission-critical user equipment, more attention should be paid to ensure that all the equipment is connected and at least has acceptable link quality. Hence, we prioritized the percentage of dropped and blocked Mission-Critical users and 5 percentile uplink and downlink throughput. Besides, in order to normalise the reward value between $[0, 1]$, the sum of the weight ratios κ, μ, η should follow:

$$\kappa + \mu + \eta = 0.5 \qquad (\kappa > \mu > \eta) \tag{11.2}$$

After wide exploration, we eventually set κ equal 0.25, μ equal 0.15 and η equal to 0.1 since this combination can eventually lead to fast model convergence during algorithm training and provide stable performance for UAV-BS connection.

11.4.4 Data Traces and Testbed

In order to simulate the dynamic environment, we created four phases. The continuous data traces were generated in those phases by introducing mission-critical user movement. When entering a new phase, users will move a random distance on both x and y axes. The distance was selected uniformly from 0 to 10 meters. Therefore, each state will map to a different set of feature values in those four phases. Due to the environment changes, the optimal state may also be different in each phase and the algorithm has to adjust its decision-making model to adapt to the dynamic environment.

During the experiment, phase I was used to train the deep reinforcement learning model from scratch while the rest three phases are used for dynamic algorithm validation. Table 11.1 summarizes all the optimal states which return the best reward value based on our defined reward function in those four different phases.

Table 11.1: Optimal states in four phases

	Optimal State $\{\sigma, x, y, z\}$	Reward (r)
Phase I	$\{30, 175, -350, 35\}$	0.905
Phase II	$\{30, -175, -350, 35\}$	0.901
Phase III	$\{30, -175, -350, 35\}$	0.86
Phase IV	$\{30, -175, -350, 10\}$	0.837

11.4.5 Deep Reinforcement Learning Method

In order to achieve better self-control decisions for our validation scenario, we applied deep q-network as our base reinforcement learning algorithm. The

algorithm was firstly proposed by Mnih et al. in [203][199] by combining convolutional neural networks with Q-learning algorithms [206] in traditional Reinforcement Learning. In order to alleviate the problems of the nonlinear network representing the value function, two major improvements were made to the traditional Q-learning [207] algorithm.
1) Experience Replay: The experience samples $e_t = \{s_t, a_t, r_t, s_{t+1}\}$ obtained from the interaction between the agent and the environment are stored in the playback memory $M = \{e_1, e_2, e_3, ...e_t\}$ unit during training at each time step t. The agent will randomly select training samples from the memory and use the Stochastic Gradient Descent algorithm to update the network parameters. When training deep networks, samples are usually required to be independent of each other. This random sampling greatly reduces the correlation between samples, thus improving the stability of the algorithm.

2) Function Approximation: The deep-Q network uses a deep convolutional network to achieve an approximate representation of the current value function and another network is also used separately to generate the target Q-value. Specifically, $Q(s, a|\theta)$ indicates the output of the current value network and the value function is used to evaluate the current state-action pair. $Q(s, a|\theta')$ represents the target network value output, namely, the target Q value.

$$Q' = r + \gamma max_{a'} Q(s', a'|\theta') \tag{11.3}$$

The parameters of the current value network are updated in real-time and copied to the target value network after every N iterations. The network parameters are updated by minimizing the mean square error between the current Q-value and the target Q-value. The loss function is listed below:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}[(Q' - Q(s, a|\theta))^2] \tag{11.4}$$

11.4.6 Deep Q-Network Architecture and Hyper-parameters

In order to find the optimal network, the random search [183] strategy was applied. Figure 11.3 illustrates the neural network part of the deep reinforcement learning model. The four-dimensional state input is connected to two fully connected layers in the learning model. Each layer contains 16 units. All layers are activated with the ReLU function. The output contains 81 values which are corresponding to 81 possible action selections.

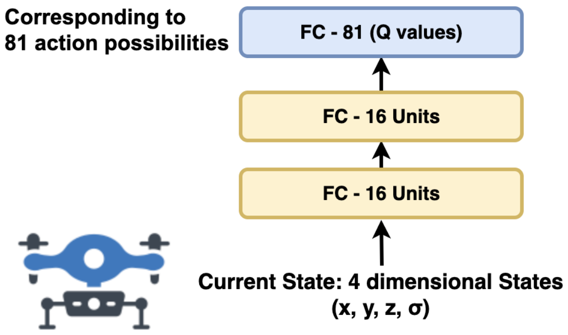

Corresponding to 81 action possibilities
FC - 81 (Q values)
FC - 16 Units
FC - 16 Units
Current State: 4 dimensional States
(x, y, z, σ)

Figure 11.3: Neural network architecture applied in the dynamic reinforcement learning algorithm

For the hyper-parameters of the deep Q-network, we explored various sets of combinations in order to achieve acceptable results. During the training, the exploration probability decay is set to 0.995, the learning rate is set to 5×10^{-5} and the number of training iterations equals 1500.

11.5 Deep Reinforcement Learning in the Dynamic Environment

This section describes the improved dynamic algorithm based on a deep Q-network for the continuous environment applied in this article. Due to the environment change, the original deep Q-network should also update its model in order to fit feature value modification. Hence, we implemented a dynamic exploration probability which is triggered by a dramatical reward value drop. Following the completion of each learning iteration, the last reward value will be checked and compared to the pre-defined drop threshold and the upper reward threshold. The adjustment will be made to exploration probability ϵ based on the result. We chose DQN as our foundation because of the algorithm's generalization. The adaptation can be employed not only in DQN, but also in other DQN-developed algorithms such as DDQN[208], etc due to the similar exploration processes. The following are the steps of the algorithm used in this paper (Algorithm 7 in appendix).

Step 1: UAV-BS explores the space and performs Q-value iterations at each training episode. An ϵ-greedy exploration is applied when determining to select the best action or randomly explore the new state. The probability of exploration is given by parameter ϵ.

Step 2: The data for each training step is stored in a replay batch $\mathcal{D}$. Specifically, each row of $\mathcal{D}$ contains the tuple (s_t, a_t, r_t, s_{t+1}), namely, current state, action, reward and next state for a training step. Samples will be randomly selected and used for Q value model updating.

Step 3: At the end of each learning iteration, the latest reward value is evaluated and compared to a pre-defined drop threshold and an upper reward threshold that includes three different conditions:

- If the latest reward value is smaller than the previous reward and the difference is larger than the drop threshold, the exploration probability will be increased to 0.1.
- If the most recent reward value is larger than the upper reward threshold, we can conclude that the algorithm has already discovered the ideal zone that is capable of providing a reliable connection to mission-critical users. The exploration probability will be set to the ending probability.
- Otherwise, after each learning cycle, the exploration probability will multiply an exploration decay and be linearly declined.

11.5.1 Baseline Algorithms

In this section, we present two baseline algorithms which were used to compare our proposed dynamic RL.

- **Retrained RL**: The retrained RL method removes previous knowledge and randomizes the ML model parameters at the start of each new phase. The exploration rate will be reset to one, and the algorithm will investigate the newly modified environment and rebuild its model from the scratch. We picked retrained RL since it is the most commonly utilized approach in a wide range of applications. Reinforcement learning is a highly sensitive technique. If we want the algorithm to be continuously trained, it must closely follow the environment. Otherwise, earlier trained knowledge may be background noise in the current context (Drone may then converge into a wrong low-performance position). The method is compared to our dynamic RL algorithm in order to demonstrate that our technique may avoid these issues by closely monitoring reward decreasing and responding fast to environmental changes, as well as the benefits and efficiency of reusing prior information.

- **Reused RL**: The initial model learned during training will be employed by the Reused RL algorithm. The exploration rate will always be set to the ending rate value 1e-3 after the training phase, regardless of how the environment has altered. Using re-used RL means that the algorithm will not explore new states even if monitored metrics show poor performance. This algorithm is commonly utilized in most empirical applications. However, it is incapable of dealing with dynamic state spaces and adapting to environmental changes. The method is utilized to compare and demonstrate the benefits of our proposed dynamic RL algorithm for handling environmental changes.

11.5.2 Validation Phases

The validation includes three phases (validation phases II, III, IV). Each phase is defined by time which includes a ten-minute walk by mission-critical users. Since the observed metrics will obviously drop after ten minutes, the reward-dropping threshold will be triggered and caused the restart of the algorithm exploration. We compared the performance of several algorithms when they started and finished their training cycle. The optimal states in each phase are calculated afterwards, which cannot be known before. After we finished our simulation, we retrieved all the history records and grid search the best value of that phase. At the time when we deploy the drone, it has no understanding of what the values and conditions are in each position, so the algorithm will help the drone investigate the environment on its own.

11.6 Results

We present the experiment results of the dynamic Reinforcement Learning (Dynamic RL) algorithm for autonomously controlling UAV-BS in this sec-

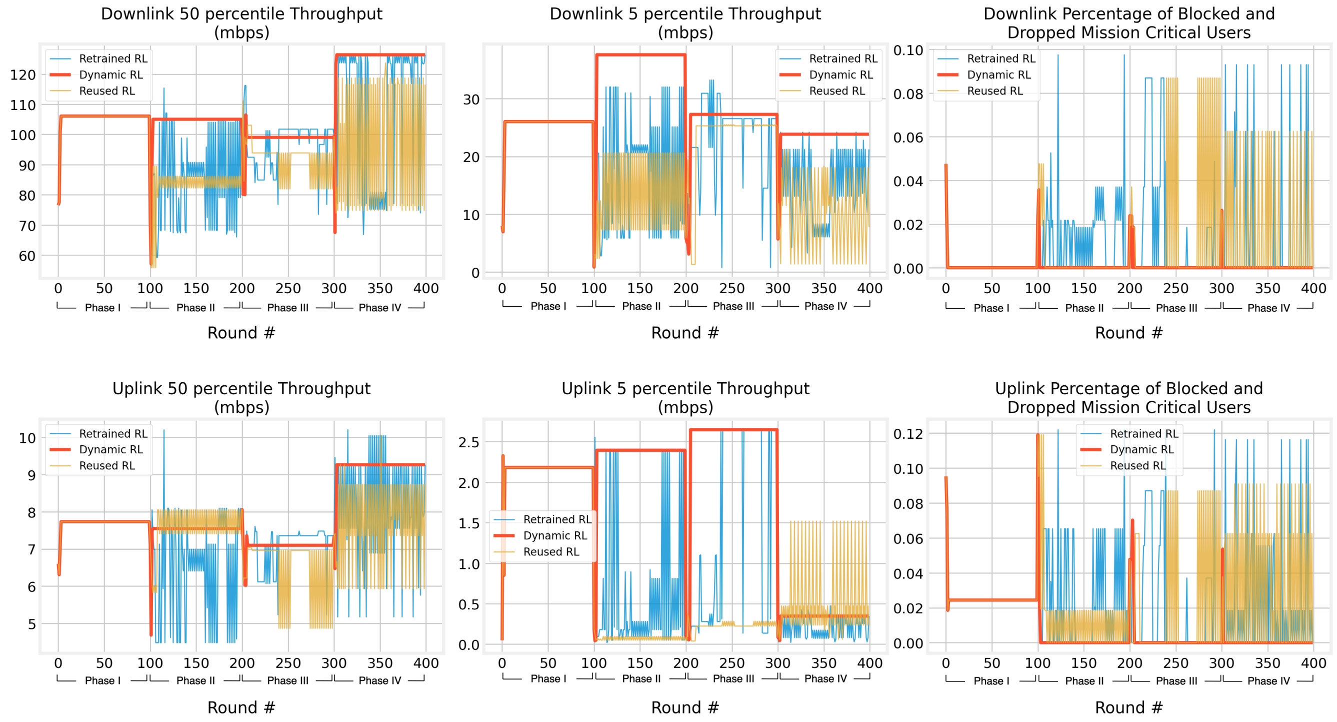


Figure 11.4: Link metrics comparison between Dynamic RL and two baseline algorithms

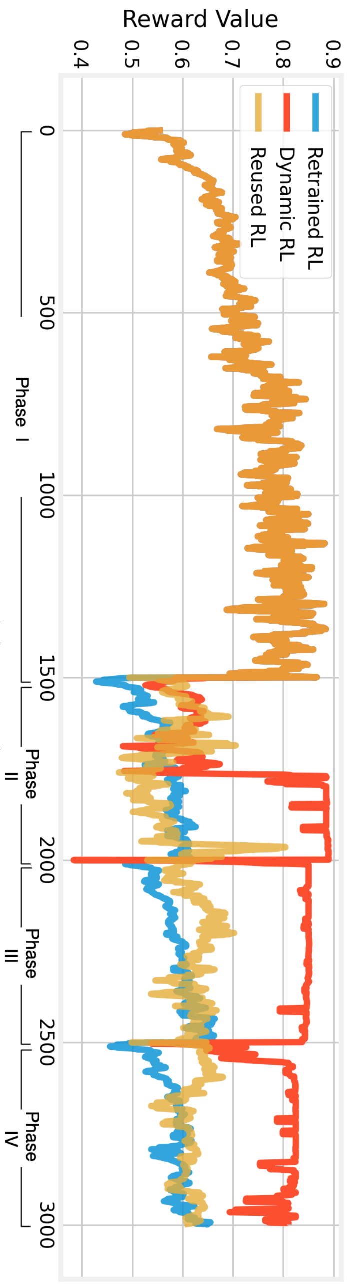


Figure 11.5: Reward Value with number of training iterations

Table 11.2: Reward value at the end of each phase

	Phase I (Training Phase)	Phase II	Phase III	Phase IV
Optimal	0.905	0.901	0.86	0.837
Dynamic RL	0.905	0.897	0.851	0.828
Re-trained RL	0.905	0.616	0.734	0.689
Reused RL	0.905	0.635	0.645	0.608

tion. The output is evaluated based on two factors - (1) Six features (determined in Section IV - Validation Environment) which demonstrate link service quality (2) Model Training in each phase. The results are compared with two baseline models - (1) Retrained Reinforcement Learning (Retrained RL) in each phase (2) Reused Reinforcement Learning (Reused RL) method. As we described before, the experiment contains four different phases. With the dynamic RL method, the algorithm will train the model from scratch in the training phase (Phase I) and then constantly improve itself in the subsequent phases. The past learned experience will be reused and the exploration rate will be dynamically set based on the comparison of the system reward value. With the Retrained Reinforcement Learning method, the UAV-BS will clear the previous knowledge and randomize the model parameters. The algorithm will retrain the model from scratch in each phase. With Reused Reinforcement Learning method, no matter how the environment changes, the algorithm will not perform any modification but continuously use the original model that was learnt from phase I.

We first compared those six features which demonstrate the link service quality of our approach with the other two baseline models to demonstrate the effectiveness of the dynamic reinforcement learning algorithm when encountering the changeable environment. Figure 13.4 illustrate the link metrics comparison between Dynamic RL, Retrained RL and the Reused RL.

From the results, we can observe that with a dynamic reinforcement learn-

Table 11.3: Comparison of the six link quality metrics with two baseline learning approaches in validation phases

Phase II	DL 50% Throughput	DL 5% Throughput	DL Drop Percentage	UL 50% Throughput	UL 5% Throughput	UL Drop Percentage
Optimal State	108.52	37.62	0%	7.55	2.39	0%
Dynamic RL	105.05	37.62	0%	7.55	2.39	0%
Retrained RL	85.5	17.27	1.5%	6.501	0.62	2.7%
Reused RL	82.2	7.25	0%	8.05	0.05	1.9%
Phase III	DL 50% Throughput	DL 5% Throughput	DL Drop Percentage	UL 50% Throughput	UL 5% Throughput	UL Drop Percentage
Optimal State	101.78	26.56	0%	7.35	2.64	0%
Dynamic RL	99.04	27.28	0%	7.09	2.64	0%
Retrained RL	96.83	23.41	1.7%	7.08	1.52	2%
Reused RL	93.87	25.3	0%	6.96	0.224	0%
Phase IV	DL 50% Throughput	DL 5% Throughput	DL Drop Percentage	UL 50% Throughput	UL 5% Throughput	UL Drop Percentage
Optimal State	126.43	25.25	0%	9.26	0.35	0%
Dynamic RL	126.43	23.85	0%	9.26	0.34	0%
Retrained RL	107.98	14.47	1.2%	8.12	0.20	2.5%
Reused RL	116.57	7.77	0%	8.73	0.23	0%

ing algorithm, UAV-BS can quickly find and reach the optimal state in each learning phase. However, due to the lack of training time and flexibility, the drone controlled by baseline models cannot reach the optimal state and may eventually oscillate between different states. The same conclusion can also be obtained from the detailed numeric Table 11.2, which compares the metrics of the six link quality monitoring features between random search, dynamic reinforcement learning method and the retrained model in validation phases II, III, IV. With the dynamic reinforcement learning method, the algorithm can reuse the previous experience obtained in the past environment and adapt its exploration probability to explore the new state and update the environment changes.

The advantage can be observed in Figure 11.5. The result demonstrates that after the environmental change in each phase, dynamic reinforcement learning can quickly update its model to the new environment while retrained RL model may need more time to converge and may not update itself and reach the acceptable service quality and the Reused RL model may be trapped into the previous knowledge and not suitable for the current environment. Table 11.2 shows the algorithm reward value at the end of each phase.

11.7 Discussion

From our experiment results, our dynamic reinforcement learning method proves to have significant advantages and is able to help commonly applied reinforcement learning methods to adapt to dynamic industrial environments. The results demonstrate that dynamic reinforcement learning requires much fewer model training iterations than retraining learning models when the environment has changed. The overall service performance of a UAV-BS can be improved by about 20 % with the dynamic reinforcement learning method if we compare the results with retrained RL and reused RL approaches. In addition, our suggested algorithm can eventually reach the ideal state in each environmental phase within a short amount of time and provide the best connectivity to the mission-critical user equipment. Our results prove that in some cases, our dynamic reinforcement learning method can lead to autonomously improvement and continuously update itself without human interaction.

Because of these advantages, the method can be used in a variety of applications. In addition to wireless autonomous drone control, the technique presented in this research can be applied in various applications requiring self-improving systems. The method can be employed not only in telecommunication scenarios but also in other circumstances where the environment may change dramatically over time. Furthermore, the novel idea of incorporating reinforcement learning into a continuous real-world environment may encourage further study and expand potential commercial applications.

11.8 Conclusion and Future Work

In this paper, we present a novel approach to apply reinforcement learning algorithms in a dynamic real-world environment. We validate our approach using a critical use case, autonomously UAV-BS control for mission-critical users. Our findings show the strength and advantages of the algorithm when trained using our proposed method. In our case, the model can help the drone quickly reach the optimal state even with environmental changes. The method can monitor the drone performance and react to changes by self-adapt exploration probability and requires much fewer model training iterations by reusing past experience.

In the future, we plan to further analyze our algorithm with different combinations of hyper-parameters, such as the upper reward threshold and lower drop threshold. As the parameter settings are sensitive and crucial for reinforcement learning and may behave completely different when use cases are changed, a more generalized learning method may help the method fit on more settings. In addition, we will test our approach in additional use cases and investigate more sophisticated reinforcement learning methods combined with our approach.

11.9 Appendix: Dynamic Reinforcement Learning Algorithm

Algorithm 6: Dynamic Reinforcement Learning with reward value monitor and adaptive exploration probability

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with two random sets of weights $\theta, \theta'$
Initialize exploration probability $\varepsilon$ to 1
Initialize restarting exploration probability $\varepsilon_{Restart}$ to 1e-1
Initialize ending exploration probability $\varepsilon_{End}$ to 1e-3
Initialize $r_{previous}$ equal to 0
for $Iteration = 1, M$ do
  for $t = 1, T$ do
    Select a random action $a_t$ with probability $\varepsilon$
    Otherwise, select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action $a_t$, collect reward $r_t$ and observe next state $s_{t+1}$
    Store the transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
    Sample mini-batch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
    if $s_{j+1}$ is terminal then
      Set $y_j = r_j$
    else
      Set $y_j = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta')$
    end if
    Perform a gradient descent step using targets $y_j$ with respect to the online parameters $\theta$
    Set $\theta' \leftarrow \theta$
  end for
  if $r_{previous}$ - $r_T$ > Drop threshold then
    Set $\varepsilon = \varepsilon_{Restart}$
  else if $r_T$ > Upper reward threshold then
    Set $\varepsilon = \varepsilon_{End}$
  else
    Set $\varepsilon = \varepsilon \times 0.995$
  end if
  Set $r_{previous} = r_T$
end for
```

CHAPTER 12

5G network on wings: A deep reinforcement learning approach to the UAV-based integrated access and backhaul

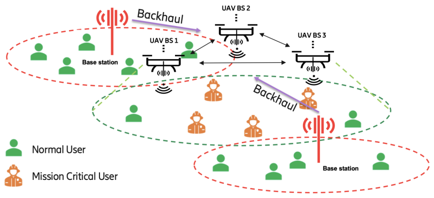
This chapter has earlier been published as
5G network on wings: A deep reinforcement learning approach to the UAV-based integrated access and backhaul
Zhang H., Qi Z., Li J., Aronsson A., Bosch J., and Olsson H. H.
In *IEEE Transactions on Machine Learning in Communications and Networking)*, vol. 2, pp. 1109-1126. IEEE.

12.1 Introduction

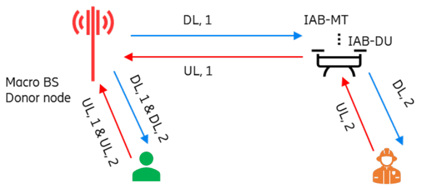
Traditional cellular infrastructure provides fast and reliable connectivity in most use cases. However, when a natural disaster happens, such traditional wireless base stations (BSs) can be damaged and therefore they cannot provide mission-critical (MC) services to the users in the disaster area. In this context, further enhancements of the cellular networks are needed to enable temporary connectivity and on-demand coverage for MC users in various challenging scenarios.

Vehicular networking can be enabled by various vehicle types including not only cars but also buses, trucks and UAVs. By equipping with a cellular tower and transceiver on a truck or trailer, cell-on-wheels have fewer cruising duration constraints and can transmit with a higher power to provide a relatively large coverage area [209]. However, cell-on-wheel placement may be less flexible for MC operations in rural areas with complex environments, such as forest firefighting, mountain search and rescue. UAV-BS (cell-on-wings) on the other hand, can be deployed in a more flexible and mobile manner. Specifically, UAVs can be used to carry deployable BSs to provide additional or on-demand coverage to users, thanks to their good mobility and higher chances of light-of-sight (LOS) propagation. However, there are a number of challenges when implementing UAV-BS assisted wireless communication networks in practice [210][211]. The system performance and user experience are significantly impacted by the deployment and configuration of UAV-BSs, including the UAV's 3-D position, operation time, antenna capabilities, transmit power, etc [212]. Using wireless backhaul, UAV-BSs can connect to the on-ground BSs (e.g., cell-on-wheels or macro BSs) and be integrated into the cellular system. Hence, it is necessary to jointly optimize the configuration parameters for the access links (between UAV-BS and on-ground users) and the backhaul links (between UAV-BSs and on-ground BSs), when optimizing UAV-BS based wireless communication systems. The optimization problem becomes even more complicated when considering different system loads and user movement on the ground. In some cases where multiple UAV-BSs are needed to cover a wide area, the complexity of providing reliable and scalable backhaul links will further increase.

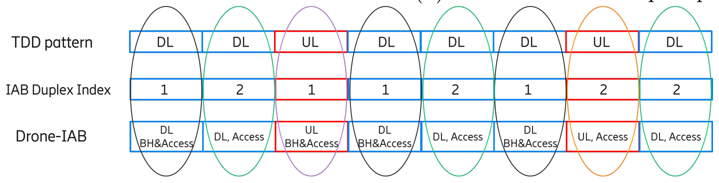
Despite the fact that there are numerous applications for UAV-based reinforcement learning algorithms, the fundamental drawback of classic RL is its low performance in a changing environment. If the environment changes (the environmental values observed by the agent change), the agent usually has to retrain the entire algorithm to keep up with the environmental changes [204][205]. In our case, user mobility would have a major impact on the system performance in terms of MC user throughput and drop rate. As a result, to ensure good service quality, a triggering mechanism needs to be implemented for algorithm adaptation and analysis. The dynamic environment, in this case, indicates that the states (user throughout and drop rate) that the agent observed will vary substantially due to wireless communication environment changes and user movement.



(a) UAV-BS assisted MC scenario



(b) IAB based TDD duplex pattern



(c) Four interference cases from the combination of IAB and TDD duplex mode

Figure 12.1: A UAV-BS assisted wireless network design enabled by a half-duplex IAB operation

12.1.1 Related Work

In recent years, UAV-BS assisted wireless communication networks have attracted significant attention from both industry and academia [188]–[192]. To guarantee a robust wireless connection between the UAV-BSs and the core network, more and more research work has started working on improving the wireless backhaul link [193]–[196], [213]. Authors in [193] assume that all the UAV-BSs are flying at a fixed height, and a robust backbone network among UAV-BSs is guaranteed by ensuring that there is always at least one path between any UAV-BS and a BS on the ground. Then they investigate the rapid UAV deployment problem by minimizing the number of UAVs to provide on-demand coverage for as many users as possible. In [194], optimal 3-D deployment of a UAV-BS is investigated to maximize the number of connected users with different service requirements by considering the limitation of wireless backhaul links. In [195], the limitation of backhaul and access capacities is also considered, and a heuristic algorithm is proposed to optimize the UAV navigation and bandwidth allocation. Similar to [194], the authors in [213] also investigate a coverage improvement problem enabled by UAV-BS with backhaul limitation but with a machine learning (ML) based solution.

Enabled by 5G new radio (NR), the integrated access and backhaul (IAB) feature can be applied to wirelessly integrate multiple UAV-BSs to an existing cellular network seamlessly [197]. Figure 12.1 shows an example of UAV-BS assisted network deployment using IAB technology. The macro-BSs who have connections with the core network are serving the normal users, and some of them can also be acting as donor-BSs, who can provide wireless backhaul connections to the flying UAV-BS. Based on the wireless backhaul link, the UAV-BS is acting as an IAB node, which can be deployed at different locations to provide on-demand services to MC users and/or normal users who are out of the coverage of the existing mobile network. To evaluate the performance of the UAV-assisted wireless system enabled by IAB, authors in [196] propose a dedicated dynamic algorithm based on the particle swarm optimization (PSO) method to optimize the throughput and user fairness. By intertwining different spatial configurations of the UAVs with the spatial distribution of ground users, [214] proposes an interference management algorithm to jointly optimize the access and backhaul transmissions. Their results prove that both coverage and capacity can be improved.

Due to the characteristics of revealing implicit features in large amounts of data, the ML methodology draws growing attention and has been extensively applied in various fields. As a sub-field of ML, agent-based reinforcement learning (RL) features in interacting with the external environment and providing an optimized action strategy. Hence, it has been used to solve complicated optimization problems that are difficult to be addressed by traditional methods. As two of the promising technologies for the next-generation wireless communication networks, it is natural to combine ML with deployable UAV-BS to solve high complexity optimization problems [215], [216].

Specifically, ML is frequently used to solve problems on deployment [213], [217], [218], scheduling [219]–[222], trajectory [223]–[231] and navigation [232]–[235] in UAV assisted network. In [217], a deep RL-based method is proposed for UAV control to improve coverage, fairness, and energy efficiency in a multi-UAV scenario. To solve the scheduling problem in a high mobility environment, the authors in [219] develop a dynamic time-division duplex

(TDD) configuration method to perform intelligent scheduling. Based on the experience replay mechanism of deep Q-learning, the proposed algorithm can adaptively adjust the TDD configuration and improve the throughput and packet loss rate. From the perspective of distributed learning, [220] proposes a framework based on asynchronous federated learning in a multi-UAV network, which enables local training without transmitting a significant amount of data to a central server. In this framework, an asynchronous algorithm is introduced to jointly optimize UAV deployment and scheduling with enhanced learning efficiency.

For ML-based trajectory and navigation, the authors in [223] investigate a trajectory strategy for a UAV-BS by formulating the uplink rate optimization problem as a Markov decision process without user-side information. The authors in [227] introduces a UAV-based downlink communication model that addresses UAVs' limited energy resources by using simultaneous wireless information and power transfer technology. By optimizing UAV trajectory, power splitting ratio, and communication scheduling through a deep reinforcement learning framework, the approach significantly enhances energy efficiency and communication quality, outperforming conventional methods. Paper [228] proposes a novel adaptable integrated sensing and communication mechanism in UAV-enabled systems, optimizing communication and sensing beamforming along with UAV trajectory to maximize system throughput while ensuring quality-of-service. Authors in [229] proposes a UAV trajectory optimization scheme based on reinforcement learning to maximize energy efficiency and network resource utilization through load balancing. Based on deep reinforcement learning, authors in [230] and [231] both propose solutions to jointly optimize the UAV trajectory and resource scheduling in UAV-assisted network. To enable UAV autonomous navigation in large-scale complex environments, an online deep RL-based method is proposed in [232] by mapping UAV's measurement into control signals. Furthermore, to guarantee that the UAV always navigates towards the optimal direction, authors in [233] enhance the deep RL algorithm by introducing a sparse reward scheme and the proposed method outperforms some existing algorithms.

Additionally, the limited battery life of a UAV restricts its flying time, which in turn affects the service availability that can be provided by the UAV. Therefore, many works have been focusing on designing energy-efficient UAV deployment or configuration schemes either with non-ML [188], [236], [237] or

ML methodologies [226], [238], [239].

12.1.2 Contributions

In this paper, we consider a scenario with multiple macro-BSs covering a large area, but due to disaster, one of the macro-BSs is damaged, which creates a coverage hole where the first responders execute their MC operations. The deployable UAV-BSs are set up to fill the coverage hole and provide temporary connectivity for these MC users. Compared with the related works and our previous paper[240] navigating only one single UAV-BS, we propose in this paper a novel RL algorithm combined with adaptive exploration and value-based action selection algorithms to autonomously and efficiently deploy multiple UAV-BSs based on the requirements. Furthermore, to extend the algorithm in a scalable manner, a decentralized architecture is proposed for the collaboration of multiple UAV-BSs. More specifically, the contributions of this paper include the following aspects:

1) We propose the framework to support applying RL algorithm for the considered use case in an IAB network architecture.
2) We applied two strategies, i.e., adaptive exploration control and value-based action selection for the RL algorithm so that the algorithm itself can adapt to a dynamic environment (e.g., MC user movement and change of channel characteristics) in a fast and efficient way.
3) We demonstrated deployment in a decentralized method for supporting multiple UAV-BSs deployment to respond to varied industrial scenarios.
4) We validate the proposed RL algorithm in a continuously changing environment with consecutive MC user movement phases. Our results show that the proposed algorithm can create a generalized model and assist in updating the decision-making on UAV-BSs and navigation in a dynamic environment.

The remainder of this paper is structured as follows. Section II introduces the system model considered in this paper. In section III, we propose a framework to enable ML in an IAB network architecture. Section IV discusses our proposed ML algorithm. Section V presents the system-level simulation results and evaluates the proposed RL algorithm. In Section VI, we summarize our findings and discuss future works.

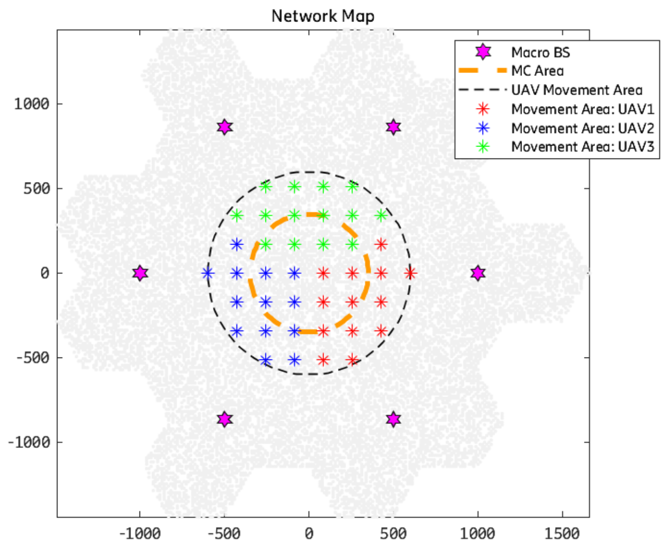


Figure 12.2: System model: UAV-BSs assisted network deployment

12.2 System Model and Problem Formulation

12.2.1 System Description

For the system model, we consider a multi-cell mobile cellular network, consisting of a public network and a deployable network, as shown in Figure 12.2. Initially, seven macro-BSs are serving users uniformly distributed in the whole area. However, one of the macro-BSs in the center of the scenario is damaged due to, e.g., a natural disaster that creates a coverage hole. For users in the central emergency area with a predefined radius (marked as an orange dashed circle), they might have very limited or no connectivity with the public network. Hence, multiple UAV-BSs, which are integrated into the public network using IAB technology, can be set up to provide temporary or additional coverage to the users in this emergency area, which is also the research target of this paper. In this paper, the UAV-BSs are limited only to stay at the discrete points indicated by the colored stars in Figure 12.2. The total number of discrete points is selected based on the criteria that the simulation data is large enough to train the proposed model but not too much to spend an excessive amount of simulation time. Hence, the navigation in the scope of this paper refers to trajectory optimization among these discrete location points. To avoid the UAV-BSs staying too close to cause strong interference to each other, we split the whole UAV-moving area (inside the black dashed circle) into three non-overlapping areas denoted by colored discrete points. For example, UAV-BS 1 is only allowed to move between location points marked as red. In the considered scenario, there are two types of users: The users located in the MC area are marked as MC users, while the others are normal users. User equipment (UE), either an MC user or a normal user, can select either a macro-BS or a UAV-BS as its serving-BS, based on the wireless link qualities between the UE and these BSs.

For the traffic pattern design, we apply a FTP-based dynamic traffic model, which is commonly used in the Third Generation Partnership Project (3GPP) [241]. All the users are randomly dropped in the scenario. For each time slot, the users are activated with a predefined arrival rate. Only these activated users can be scheduled and initiate fixed-size data transmission based on the link quality (both access and backhaul links) and system load for downlink and uplink, respectively. When the data transmission is completed, the user will leave the system and wait to be activated again. Then the user throughput can be calculated with actually served traffic and consumed time to deliver the traffic.

As mentioned before, the UAV-BSs work as the IAB nodes in the current scenario. They will measure the wireless link to all macro-BSs and select one with the best link quality as their donor-BSs. Once the wireless backhaul link between the UAV-BSs and their donor-BSs is established, the three sectors of the UAV-BSs will share this wireless backhaul link and provide access service to both normal users and MC users. For the users served by the UAV-BSs, the corresponding throughput depends not only on the access link but also on the wireless backhaul link. While selecting the access links, the users with too bad link quality, for instance, below a certain threshold, will be dropped. To reduce the complexity and the load-bearing of the UAV-BSs, it is assumed that the same antenna configuration is applied for both access and backhaul antennas of the UAV-BSs. The reason to make such an assumption also includes that the positions of UAV-BSs have more impact on the key performance metrics (e.g., user throughput and drop rate) than varying the tilt of access and backhaul antennas. The proposed model is also applicable in the real world. The number of discrete points for UAV-BSs can be selected according to the computing capability of the target system in certain scenarios. If the UAV-BS supports two separate antenna panels for access and backhaul links, the antenna configuration (e.g., antenna tilt) for access and backhaul links can be adjusted respectively to further improve the performance when the same configuration is applied for both access and backhaul antennas. The proposed model can handle such cases by adding antenna tilt as a new input feature.

The system operates under a TDD model, and the time slot pattern consists of downlink (DL), DL, uplink (UL), and DL, which is repeated with a periodicity of 2 ms [242]. The system bandwidth is 100 MHz, and it is shared between backhaul and access links. The time slots assigned for UL/DL Access/Backhaul links are shown in Figure 12.1(b) and (c). Two full TDD periods are required to cover all eight UL/DL Access/Backhaul combinations, which lead to four interference cases, denoted as: DL1, DL2, UL1, and UL2. As shown in Figure 12.1(b), for the UAV-IAB node, DL1 and UL1 are reserved for backhaul link transmission, while DL2 and UL2 are reserved for providing access services for users. For the donor-BS and all other macro-BSs, all the time slots can be used for access link transmission. In each interference case, the interfering nodes for users are different from those in other interference cases. For example, in the DL1 case where the time slots are used for both backhaul and access links, the UAVs are acting as users and the interfering nodes in this case only include marco-BSs. While in the DL2 case where the time slots are only used for access links, the UAVs are acting as BSs to serve users. In this case, the interfering nodes in DL include both macro-BSs and UAV-BSs. These features are all captured in the proposed ML model by importing the performance metrics(e.g., user throughput and drop rate) in the reward function, which will be introduced in detail in Section III.

To validate the performance of the proposed algorithm in adapting to a dynamic environment, we established five distinct phases in the time domain. Data traces required for training were captured at the beginning of each phase while the user movement (i.e. changes in user locations) was considered between phases. Upon entering a new phase, each user moves a random distance along both the horizontal and vertical directions in two-dimensional space, with the moving distances uniformly selected from 0 to 10 meters. In addition to the changes in user locations, the configuration related to channel conditions including fading and multi-path components are updated when switching phases. Consequently, the state of each phase maps to a different set of feature values across these five phases. Due to these environmental changes, the optimal state may vary in each phase, requiring the UAV-BS to adjust its decision-making model to adapt to the dynamic environments.

12.2.2 Transmission Model

For the public network in this paper, we use an urban-macro propagation model [243], while a refined aerial model from 3GPP standardization is used for UAV-BS [241]. It is assumed that the network consists of D macro-BS, M UAV-BSs and N users, denoted by $\mathcal{D} = \{1, 2, ..., D\}$, $\mathcal{M} = \{1, 2, ..., M\}$ and $\mathcal{N} = \{1, 2, ..., N\}$. The whole available bandwidth W is divided into K sub-channels and each one has a bandwidth denoted by $B_k = \frac{W}{K}$. Assuming that the three dimensional coordinates of the m^{th} UAV-BS and the n^{th} user are (x_m, y_m, h_m) and (x_n, y_n, h_n), respectively. Based on 3GPP channel model[243], the following formula is applied to represent the probability of LOS propagation between UAV-BS m and user n:

$$Pr_{LOS}^{(m,n)} = \begin{cases} 1, & d_{2D}^{(m,n)} \leq d_{2D}^{Th} \\ [\frac{18}{d_{2D}^{(m,n)}} + \exp(\frac{-d_{2D}^{(m,n)}}{63}) & \\ \quad \times (1 - \frac{18}{d_{2D}^{(m,n)}})] \times & \\ \quad (1 + \frac{5}{4}e^{-6} \times C^{'}(h_n) {d_{2D}^{(m,n)}}^3 & \\ \quad \exp \frac{-d_{2D}^{(m,n)}}{150}) & d_{2D}^{(m,n)} > d_{2D}^{Th} \end{cases} \tag{12.1}$$

where,

$$C^{'}(h_n) = \begin{cases} 0, & h_n \leq 13 \\ \left(\frac{h_n - 13}{10}\right)^{1.5}, & 13 < h_n \leq 23 \end{cases} \tag{12.2}$$

$h_n \in \left[h_n^{min}, h_n^{max}\right]$ denotes the height of user n with meter as a unit, while h_n^{min} and h_n^{max} denote the minimal and maximal height of a user, respectively. $d_{2D}^{(m,n)} = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}$ is the horizontal distance between UAV-BS m and user n. d_{2D}^{Th} is a 2D distance threshold and its value is 18 meters. The path loss between UAV-BS m and user n in the case of LOS propagation and NLOS propagation can also be derived based on [243]:

$$PL_{LOS}^{(m,n)} = 28 + 22log_{10}\left(d_{3D}^{(m,n)}\right) + 20log_{10}(f_c) \tag{12.3}$$

$$\begin{aligned} PL_{NLOS}^{(m,n)} = 13.54 + 39.08log_{10}\left(d_{3D}^{(m,n)}\right) \\ +20log_{10}(f_c) - 0.6\,(h_n - 1.5) \end{aligned} \tag{12.4}$$

where $d_{3D}^{(m,n)} = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2 + (h_m - h_n)^2}$ denotes the distance between the antennas of UAV-BS m and user n, while f_c is the carrier frequency. Hence the average path loss between UAV-BS m and user n can be denoted as:

$$\begin{aligned} PL_{MN}^{(m,n)} = Pr_{LOS}^{(m,n)} \times PL_{LOS}^{(m,n)} + \\ \left(1 - Pr_{LOS}^{(m,n)}\right) \times PL_{NLOS}^{(m,n)} \end{aligned} \tag{12.5}$$

Similarly, $PL_{DN}^{(d,n)}$ denotes the average path loss between macro-BS d and user n, while $PL_{DM}^{(d,m)}$ denotes the average path loss between macro-BS d and UAV-BS m. To indicate whether a sub-channel is occupied by a UAV-BS/macro-BS to serve the users, an occupy indicator is defined, and setting c_i^k as 1 implies that the sub-channel k is occupied by UAV-BS/macro-BS i. Meanwhile, another indicator is defined where $\alpha_m^n = 1$ indicating that user n is served by UAV-BS m. Hence, the SINR between UAV-BS m and user n on sub-channel k can be denoted as:

$$\Upsilon_{(n,m)}^k = \frac{c_m^k \alpha_m^n \left(P_m - PL_{MN}^{(m,n)}\right)}{N_0 B_k + \sum_{i \neq m} c_i^k \left(P_i - PL_I^{(i,n)}\right)} \tag{12.6}$$

where P_m and P_i represent the transmit power of UAV-BS m and interfering node i, respectively. N_0 is the power spectral density of the additive Gaussian noise. When the UAV-BSs are serving users, the interference may not only come from the other UAV-BSs serving users but also from the macro-BSs serving other UAV-BSs/users. Therefore, $PL_I^{(i,j)}$ generally denotes the path loss between user/UAV-BS j and interfering node (UAV-BS/macro-BS) i. Based on the above-mentioned expressions, the achieved throughput for MC user can be obtained by:

$$C_n = \sum_{k}^{K} \lambda_n B_k log_2 \left(1 + \Upsilon_{(n,m)}^{k}\right) \tag{12.7}$$

where λ_n is the user drop indicator. The user n with an SINR lower than Υ_{min} will be dropped and its corresponding user drop indicator λ_n equals zero. For the drop rate of MC users which will be used in the following sections, it is defined as:

$$\beta_{MC} = \frac{\sum_{n}^{N_{MC}} \lambda_n}{N_{MC}}, \lambda_n \tag{12.8}$$

where N_{MC} is the number of MC users.

The intention of this paper is to generate a generic model, which can be used to navigate the UAV-BSs to serve MC users in typical MC scenarios. That is why the statistic model is applied in the transmission model mentioned above to calculate the user SINR. In comparison, the real map scenario models the physical objects in a specific environment, which can capture the blockage effect in the network [244]. If there is a need to apply this generic model in a specific scenario, the generic model can be further refined to accommodate such scenario, which is also the next step in our future research.

12.2.3 Problem Formulation

In a multi-network scenario consisting of both existing BSs on the ground and temporarily deployed UAV-BS, the deployment of the UAV-BS play a critical role in guaranteeing the performance of the target users/services (e.g., MC users/services). It can also impact the overall system performance. As the UAV-BS is connected to the core network using wireless backhaul, it is important to ensure the good quality of both the backhaul and access links when performing this system optimization. Furthermore, the optimal solution depends on many factors like network traffic load distribution, quality of service (QoS) requirements and user movements on the ground. Therefore, jointly optimizing these parameters of UAV-BS is a complex system-level optimization problem that needs to be solved in a dynamic changing environment.

In order to best serve target users while also maintaining a good backhaul link quality between UAV-BSs and their donor-BSs, we aim to solve the following research problems: 1) Design an RL algorithm to jointly optimize the

3-D locations of the UAV-BSs. 2) Find the movement strategy of a UAV-BS to accommodate the dynamically changing user distribution.

Based on the system model introduced in the previous sub-section, the target problem we intend to solve is optimizing the 3-D locations of the UAV-BSs to maximize a weighted sum of the following system key performance metrics for the MC users:

- Backhaul link rate for UAV-BS: On one hand, the backhaul link rate reflects the link quality when the UAV-BS is served as a user via its donor-BS. On the other hand, it also affects the end-to-end throughput performance of its associated users since the throughput of UAV-served users is calculated by considering the quality of both the access link and backhaul link.
- The 5-percentile and 50-percentile of the cumulative distribution function (CDF) of MC user throughput: The 5-percentile MC user throughput represents the performance of the cell-edge MC users, i.e., the MC users with the "worst" throughput performance, while the 50% throughput indicates the average MC user performance in the simulation area.
- Drop rate for MC users: The ratio of MC users that cannot be served with the required services. This is an important performance metric for MC scenarios, since for MC users, keeping reliable connectivity broadly is more important than guaranteeing high-demand services for specific users in most MC cases.

Although the performance metrics of normal users is also critical to evaluate the overall performance of the network even in an MC scenario, in this paper, we only focus on improving the performance of MC users because the performance of normal users is nearly not impacted by broken macro-BS and deployed UAV-BSs in the system model.

12.3 ML-based Solution

In this section, we describe how we transform and model the considered use case in an ML environment. Three important components, including the state space, action space, and reward function, are constructed in order to design an RL algorithm to jointly optimize the 3-D position of multiple UAV-BSs in an IAB network.

12.3.1 Modeling of ML Environment

12.3.1.1 State Space

In our case, a UAV-BS state at a given time instance t has three dimensions, namely a UAV-BS's 3-D position.

We use $\mathcal{P}_t = \{x_t, y_t, z_t\}$ to denote the 3-D position of a UAV-BS at time t. Then, a UAV-BS's state at a given time instance t is denoted as $s_t = \{x_t, y_t, z_t\}$. Table 12.1 shows the candidate values for each UAV-BS:

Table 12.1: Candidate values for each UAV-BS in the simulation environment

3-D position $\mathcal{P}$ Space	UAV1 Candidate Values
x	$[85, 257, 428, 600]$ meters
y	$[-514, -342, -171, 0]$ meters
z	$[10, 20]$ meters
3-D position $\mathcal{P}$ Space	**UAV2 Candidate Values**
x	$[-600, -428, -257, -85]$ meters
y	$[-514, -342, -171, 0]$ meters
z	$[10, 20]$ meters
3-D position $\mathcal{P}$ Space	**UAV3 Candidate Values**
x	$[-428, -257, -85, 85, 257, 428]$ meters
y	$[171, 342, 514]$ meters
z	$[10, 20]$ meters

It should be noted in Table 12.1 that the available height range for all UAV-BSs is limited between 10 m and 20 m, rather than deploying the UAV-BSs into a higher altitude. The reason is that, in the scenario considered in this paper, the UAV-BSs tend to stay at a lower height to maintain good backhaul links to on-ground donor-BS and also provide better access links to serve on-ground MC users, which makes the current height range selection reasonable.

The candidate values of 2-D space location x and y axis cover the disaster area shown in Figure 12.2. The location options are selected by three deployed UAV-BSs. The 2-D MC area has been divided into 3 parts, with each UAV-BS covering one part of the area. For UAV1, x and y axis options are $[85, 257, 428, 600]$ and $[-514, -342, -171, 0]$ meters. For UAV2, x and y axis options are $[-600, -428, -257, -85]$ and $[-514, -342, -171, 0]$ meters. For UAV3, x and y axis options are $[-428, -257, -85, 85, 257, 428]$ and $[171, 342, 514]$ meters. And the height options for all three UAV-BSs in the z axis are $[10, 20]$ meters. As a result, the total number of state combinations in this environment is 18928. The computation complexity will be linearly increased $O(n)$ based on the total number of input states combination.

12.3.1.2 Action Space

In order to enable a UAV-BS to control its state, for each state dimension, we defined three potential action options and the UAV-BSs chose an action from three candidate options. These three alternative action options are denoted by the three digits: $-1, 0, 1$, where “-1" indicates that a UAV-BS decreases the status value at this state dimension by one step from its current value; “0" indicates that a UAV-BS does not need to take any action at this state dimension and keeps its current value; "1" indicates that a UAV-BS increases the status value at this state dimension by one step from its current value.

For example, if the x-axis value of the UAV1 (i.e. the value of the x_t dimension) equals 257 meters, an action coded by “-1" for this dimension means that the UAV-BS will select an action to reduce the position value to 85 meters, an action coded by “0" implies that the UAV-BS will hold the current position (257 meters), and an action coded by “1" implies that the UAV-BS will increase the position value to 428 meters. The same policy is applied to all dimensions of the state space.

Since there are three action alternatives for each space state, the action pool for 3-D position space, the pool has 27 action candidates that may be programmed to an action list $\mathscr{P}$ =[(-1, -1, -1), (-1, -1, 0), (-1, -1, 1), (-1, 0, -1) ..., (1, 1, 1)]. As a result, if we combine the action of the 3-D position space, at any given moment t, a UAV-BS can thus choose an action a_t from these 27 alternatives. Figure 12.3 depicts a state transition from the specified state $s_t = \{257, -342, 10\}$ meters.

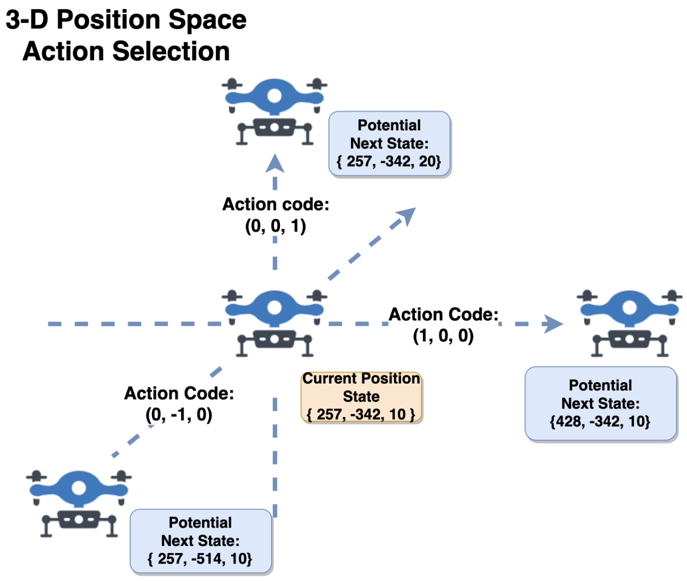


Figure 12.3: Example of the UAV1's state transition from current state $\{257, -342, 10\}$ meters

12.3.1.3 Reward Function Design

It is more critical to serve as many MC users as possible with appropriate service quality than to maximize the peak rate of a subset of MC users. In the MC context, ensuring a seamless and reliable communication service for all users is paramount. The user experience is intricately tied to two key performance metrics: drop rate and throughput, each addressing distinct aspects of communication quality.

1. Drop Rate (β): The drop rate metric is a critical indicator of connection reliability. It reflects the percentage of users who remain connected without disruptions in both uplink (UL) and downlink (DL) communication. In MC scenarios, where ubiquitous connectivity is essential, minimizing the drop rate is synonymous with ensuring that every user remains connected to the communication network. A low drop rate implies a higher level of reliability and availability of the communication service. This is particularly crucial in MC scenarios where universal access takes precedence over-optimizing the communication quality for a specific subset of users. By minimizing the drop rate, the model prioritizes the requirement of connecting every user in the disaster area.

2. Throughput (α): Throughput metrics, on the other hand, provide insights into the quality of the communication service. The 50th percentile and 5th percentile throughput values represent the average and "worst" performance of MC users, respectively, in both UL and DL. These metrics delve into the actual service quality experienced by users. In a mission-critical context, optimizing communication quality is crucial to meet the diverse needs of users. Throughput metrics ensure that not only are users connected, but the quality of their communication experience is also considered. This is particularly relevant when users in the disaster area may have continuous communication demands, and the network must adapt to dynamically changing conditions.

Together, drop rate and throughput metrics provide a holistic view of the user experience in mission-critical scenarios. A low drop rate ensures universal connectivity, meeting the fundamental requirement of MC scenarios, while throughput metrics delve into the aspects of service quality. Balancing both aspects is essential for delivering a comprehensive and reliable user experience that aligns with the needs of MC users in disaster areas. It's worth noting that there may be trade-offs between connection reliability and service quality. Striking the right balance ensures that the communication network not only connects all users but also provides satisfactory service quality, acknowledging the dynamic nature of MC scenarios. Therefore, a reward function is constructed for measuring the overall user experience of current service settings.

As the reward function reflects the overall user experience of the MC users, the aggregated reward metrics are produced for the reward function design of the reinforcement learning algorithm to take into account both the impact of other drones' actions as well as the quality of services at the local drone. The reward is calculated using the average of the performance indicators of local and neighbouring agents. We have selected six key performance metrics for each local agent to highlight the local quality of service for MC users, including:

- The drop rates of MC users for UL and DL (β_{ul}, β_{dl}), which reflect the percentage of unserved MC users.
- The 50% throughput values of MC users for both UL and DL ($\alpha_{ul-50\%}$, $\alpha_{dl-50\%}$), which represent the average performance of the MC users, and
- The 5% throughput values of MC users for both UL and DL ($\alpha_{ul-5\%}$, $\alpha_{dl-5\%}$), which represent the "worst" performance of the MC users.

The choice of performance metrics in our study is linked to the unique challenges and priorities inherent in MC scenarios, where the primary objective is to establish and maintain reliable communication services for all users within the disaster area. Unlike conventional scenarios that may prioritize maximizing the peak rate for specific users, our focus is on universal service delivery and quality. This distinctive prioritization is a direct response to the critical nature of MC scenarios, where seamless communication can be a matter of life and death. The selected metrics serve as crucial indicators to address the specific needs of emergency situations and ensure the effective deployment of UAV-BSs.

The reward function is built as a weighted sum of these six feature values to balance these critical performance indicators, as shown below. The reason why the backhaul link rate is not considered here is that the values of the six features all rely on the quality of the backhaul link between the UAV-BS and its donor-BS. Before the model, all characteristics are normalized using min-max normalization, thus the values are constrained within the range $[0, 1]$.

$$\begin{aligned} R_s = & \omega_1 \times \frac{(1-\beta_{dl}) + (1-\beta_{ul})}{2} + \omega_2 \times \frac{(\alpha_{ul-5\%} + \alpha_{dl-5\%})}{2} \\ & + \omega_3 \times \frac{(\alpha_{ul-50\%} + \alpha_{dl-50\%})}{2} \end{aligned} \tag{12.9}$$

Furthermore, we set weighting coefficients $\omega_1 + \omega_2 + \omega_3 = 1$ to normalize the reward value such that R_s is between $[0, 1]$. To emphasize the significance of supporting all MC users, we assign higher weights to user drop rates and 5% MC-user throughput metrics. This is because, in the MC use cases, we must first prioritize that all users have access to the communication service rather than focusing on optimizing the communication quality of a small subset. In this paper, our method uses the weight values $\omega_1 = 0.5$, $\omega_2 = 0.3$ and $\omega_3 = 0.2$. The weighting of these metrics in the reward function emphasizes our commitment to supporting all MC users, as opposed to optimizing the communication quality for a select group. This intentional emphasis aligns with the core principle that in MC scenarios, every user's access to communication services is of paramount importance. This normalization ensures that the reward values are representative and comparable across diverse scenarios. The reward function's formulation involves a balance of weights assigned to key performance metrics to optimize the algorithm for mission-critical scenarios. Drop rate (ω_1) bears the highest weight, underscoring its critical role in minimizing service interruptions and prioritizing universal access. The weight for 5th percentile throughput (ω_2) is selected to address the trade-off between reducing drop rates and ensuring quality for the worst-performing users. Similarly, the weight for 50th percentile throughput (ω_3) is determined to strike a balance between providing quality for the majority and not compromising the performance of the most disadvantaged users. The optimal combination of these weights is identified through grid search, ensuring the reward function aligns with the unique priorities of mission-critical scenarios and achieves peak system performance. In order to know the influence of each UAV-BS, the reward function will also aggregate the reward values of the neighbour agents. Hence, the following is the reward function applied in the algorithm:

$$R_s = \frac{\sum_{c=1}^{C} \mathcal{M}^c + \mathcal{M}}{len(C) + 1} \tag{12.10}$$

Assuming that C is the set of register neighbours, $\mathcal{M}$ represents the current agent's local system performance, and the $\mathcal{M}^c$ indicates the local system performance of its neighbour ID c.

12.3.2 RL Algorithm Design

In this section, we design an RL algorithm to solve the optimization problem of the considered use case. RL is distinct from supervised and unsupervised learning in the field of ML in that supervised learning is performed from a training set with annotations provided by an external supervisor (task-driven), whereas unsupervised learning is typically a process of discovering the implicit structure in unannotated data (data-driven). RL is suitable for this case since the method provides a unique feature: the trade-off between exploration and exploitation, in which an intelligence agent must benefit from prior experience while still subjecting itself to trial and error, allowing for a larger action selection space in the future (i.e., learning from mistakes).

In order to achieve better self-control decisions for our scenario, we applied deep Q-network (DQN) as our base RL algorithm. The algorithm was first proposed by Mnih et al. in [203][199] by combining convolutional neural networks with Q-learning algorithms [206] in traditional RL. The approach has been frequently used in gaming and static environments. However, the original approach is incapable of adapting to our MC situation due to environmental changes. To address these issues, we have proposed two significant schemes in our autonomous UAV-BSs control algorithm (Algorithm 7): adaptive exploration control and value-based action selection.

12.3.2.1 Adaptive Exploration (AE)

Because of the environmental changes, the original DQN model needs to be updated to accommodate feature value changes. As a result, we create a dynamic exploration probability triggered by a substantial decline in reward value. Following the completion of each learning iteration, the final reward value is checked and compared to the pre-defined reward drop and upper reward thresholds. Based on the outcome, the exploration probability ϵ will

Algorithm 7: Deep Reinforcement Learning in each UAV-BS with adaptive exploration and value-based action selection

```
Initialize the agent's replay memory Buffer $\mathcal{D}$ to
capacity $M$
Initialize action-value function $Q$ with two random sets
of weights $\theta, \theta'$
Initialize exploration probability $\varepsilon$ to 1
Set previous reward value $r_p$ to 0
for $Iteration = 1, N$ do
  for $t = 1, T$ do
    $\mathscr{T}_t, \mathscr{P}_t \leftarrow$ Action_Selection($r_p$, $r_t$, $\varepsilon$)
    $a_t = \{\mathscr{T}_t, \mathscr{P}_t\}$
    Set $r_p = r_t$
    Decode $a_t$ to action options in four state
    dimensions and execute the actions
    Collect reward $r_t$ and observe the agent's next state
    $\mathcal{P}_{t+1} \leftarrow \{x_{t+1}, y_{t+1}, z_{t+1}\}$
    Set $s_{t+1} = \{\mathcal{T}_{t+1}, \mathcal{P}_{t+1}\}$
    Store the state transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
    $\mathcal{A}_s$, $\mathcal{A}_o \leftarrow$ Action_Grouping($a_t$)
    Sample mini-batch of transitions $(s_j, a_j, r_j, s_{j+1})$ from buffer $\mathcal{D}$
    if $s_{j+1}$ is terminal then
      Set $y_j = r_j$
    else
      Set $y_j = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta')$
    end if
    Perform a gradient descent step using targets $y_j$
    with respect to the online parameters $\theta$
    Set $\theta' \leftarrow \theta$
    $\varepsilon \leftarrow$ Adaptive_Exploration($r_p$, $r_t$, $\varepsilon$)
  end for
end for
```

be adjusted.

Each UAV-BS initially explores the state space and then performs Q-value iterations at each training episode. When deciding whether to take an action that gives the maximum reward value or randomly explore a new state, a ϵ-greedy exploration is used. The parameter ϵ determines the likelihood of exploration. Each training step's data is saved in a replay batch $\mathcal{D}$. Each row of $\mathcal{D}$ holds the tuple (s_t, a_t, r_t, s_{t+1}), which represents the current state, action, reward, and next state for a training step. Samples will be chosen at random and used to update the Q value model.

The most recent reward value is reviewed and compared to a pre-defined reward-drop threshold and an upper reward threshold. Then, the exploration probability is updated by checking the following three conditions: (Algorithm 8):

- If the most recent reward value is less than the prior reward, and the difference is greater than the reward drop threshold, the exploration probability is increased to 0.1.
- If the most recent reward value exceeds the higher reward threshold, we can conclude that the algorithm has already located the optimal zone capable of delivering a reliable connection to MC users. The likelihood of exploration will be matched to the probability of completion.
- Otherwise, the exploration probability will multiply by an exploration decay and fall linearly after each learning cycle.

12.3.2.2 Value-Based Action Selection (VAS)

Although the ϵ-greedy algorithm can strike a reasonable balance between exploration and exploitation, in some cases the approach utilized for exploration is redundant and time-consuming. The algorithm will choose actions at random throughout the searching stage, which may lengthen the search time. However, when dealing with a large action and state space, random action selection is clearly not an effective strategy and may cause decision-making to be delayed, which is unacceptable in most time-critical businesses. Therefore, we propose a novel value-based action selection strategy (Algorithm 9) which can lead to fast decision-making for a UAV-BS when determining its 3-D space location.

Algorithm 8: Adaptive Exploration Algorithm (AE)

```
Set restarting exploration probability $\varepsilon_{Restart}$
to 0.1
Set ending exploration probability $\varepsilon_{End}$ to 0.0001
Set exploration decay $\varrho$ to 0.995
Function Adaptive_Exploration($r_p$, $r_t$, $\varepsilon$):
    if $r_p$ - $r_t$ > Drop threshold then
        Set $\varepsilon = \varepsilon_{Restart}$
    else if $r_t$ > Upper reward threshold then
        Set $\varepsilon = \varepsilon_{End}$
    else
        Set $\varepsilon = \varepsilon \times \varrho$
    end if
    return $\varepsilon$
```

As described in the previous section, an agent's 3-D position state at a given time instance t is denoted as $\mathcal{P}_t = \{x_t, y_t, z_t\}$. Since each position state has three dimensions and each state dimension has three action options, the action pool contains in total 27 action candidates that can be programmed to a list of action space $[(-1,-1,-1),(-1,-1,0),(-1,-1,1),...(1,1,1)]$. Each element in this list can then be regarded as an action vector.

Figure 12.4 depicts a probable set of next actions with the same or opposite consequence. The consequence is defined as the reward value (or monitored performance metrics) change after an action has been executed. The algorithm will analyze the outcome of past actions. If the prior action decision has a positive outcome (the reward value increases or monitored performance metrics become better) as defined above, the algorithm will choose actions from a pool of following actions with the same consequence. The dot product between two action vectors determines the result. If the dot product is larger than 0, this action vector can be assumed to have the same outcome as the prior action option.

If the previous action decision results in a negative consequence (the reward value decreases or monitored performance metrics become worse), the algorithm will select actions from the pool consisting of potential next actions with the opposite consequence. The opposite consequence is determined by the dot product of two action vectors that is smaller than or equal to 0. The actions in this pool will result in an opposite consequence compared with the previous action decision. Assume that the previous action vector is $\vec{\mathscr{P}}_t$ while

Algorithm 9: Value-based action selection (VAS)

```
Set grouping threshold $\beta$ to 0
Function Action_Grouping($a_t$):
    $\mathscr{P}_t \leftarrow a_t = \mathscr{T}_t, \mathscr{P}_t$
    for all potential next action $\mathscr{P}_{t+1}$ do
        if $\vec{\mathscr{P}}_t \cdot \vec{\mathscr{P}}_{t+1} > \beta$ then
            Append $\vec{\mathscr{P}}_{t+1}$ to $\mathcal{A}_s$
        else
            Append $\vec{\mathscr{P}}_{t+1}$ to $\mathcal{A}_o$
        end if
    end for
    return $\mathcal{A}_s$, $\mathcal{A}_o$

Function Action_Selection($r_p$, $r_t$, $\varepsilon$):
    if $r_t \geq r_p$ then
        Select a random action $\mathscr{P}_t$ with probability $\varepsilon$
        from the same consequence 3-D position
        action pool $\mathcal{A}_s$
    else
        Select a random action $\mathscr{P}_t$ with probability $\varepsilon$
        from the opposite consequence 3-D position
        action pool $\mathcal{A}_o$
    end if
    Otherwise, select $a_t = \arg\max_a Q(s_t, a; \theta)$
    return $\mathscr{T}_t, \mathscr{P}_t$
```

the next potential action vector is $\vec{\mathscr{P}}_{t+1}$:

$$\begin{cases} \vec{\mathscr{P}}_t \cdot \vec{\mathscr{P}}_{t+1} > 0 & \text{Same consequence as previous} \\ \vec{\mathscr{P}}_t \cdot \vec{\mathscr{P}}_{t+1} \leq 0 & \text{Opposite consequence as previous} \end{cases} \tag{12.11}$$

In Figure 12.4, the red vector represents the previous action decision. The angle between the previous action vector (red vector) and the green vectors is less than $\frac{\pi}{2}$, which can be represented by a dot product greater than zero. As a result, the green vectors represent actions that may result in the same consequence as the red vector. Similarly, the angle between the previous action vector (red vector) and the brown vectors is greater than or equal to $\frac{\pi}{2}$, which is represented by a dot product value less than 0. As a result, the brown vectors may have the opposite consequence.

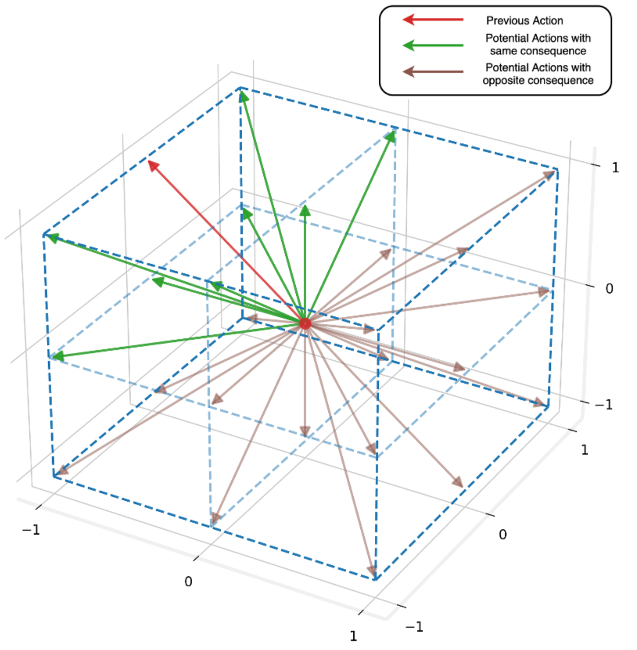

Previous Action
Potential Actions with same consequence
Potential Actions with opposite consequence
1
0
−1
1
0
−1
−1
0
1

Figure 12.4: Diagram of a potential set of next actions with same or opposite consequence

During the UAV-BSs deployment, the algorithm monitors a set of critical system performance values (the reward value). Based on the current and a set of previous performance values, the algorithm will evaluate the consequences caused by the previous action. The algorithm will thus select the action set which will potentially result in positive consequences.

12.3.3 Decentralized Reinforcement Learning

In some circumstances, a single UAV is not capable of being extended to cover a larger area. As shown in Figure 12.1(a), multiple UAV-BSs are deployed to work together to service the MC users. An extensible decentralized method for deploying numerous UAV-BSs is therefore designed. The concept is illustrated in Figure 12.5 where we relocate the central server operation function from the central entity and attach it to the edge entity on UAV, as opposed to the typical single-agent reinforcement learning algorithms, to achieve decentralized characteristics.

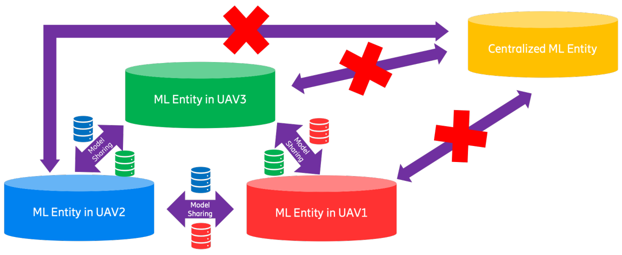


Figure 12.5: Decentralized Architecture for Multi-UAV Coordination

The system has two different kinds of data for exchanging information, namely the system-related data (including location information and system KPIs) and the model data. The location will communicate with nearby drones regarding the connection performance and UAV-BS system-related data. These kinds of data can assist each drone in understanding how their movements affect the others and in being aware of one another's surroundings. Following each UAV-BS decision, the information will be continuously exchanged and used as a guide for the subsequent choice. The local model of each UAV-BS will be shared with its neighbours via the model data channel. Each UAV-BS has a separate procedure to train, communicate, and receive model weights and service metrics during the learning process. Each UAV-BS will share its learning experiences as a result, and the others can gain information from the experiences of the others. Information is exchanged asynchronously through active listening to neighboring UAV-BSs for receiving models and service metrics, rather than requesting them. This push-based communication mechanism helps avoid interruptions from malfunctioning or slow neighbors. After multiple training epochs, the UAV-BSs can swap their model with their neighbours under the control of a frequency parameter. The process is described in Algorithm 7. The procedures can be summarized as follows:

Step 1: Each training episode will begin with each UAV-BS exploring and locating its neighbours before moving on to exploring the environment and doing Q-value iterations. When deciding whether to choose the best action or to randomly explore the new state, a ϵ-greedy exploration is used. The parameter ϵ specifies the likelihood of exploration.

Algorithm 10: Transmission functions of the decentralized reinforcement learning algorithm (DecRL)

```
for $Iteration = 1, N$ do
  for $t = 1, T$ do
    After action selections:
    for each client $c \in C$ in parallel do
      send $\{s_{t+1}, \mathcal{M}\}$;
      receive $\{s^c_{t+1}, \mathcal{M}^c\}$
    end for$s_{t+1} = \{(x_{t+1}, y_{t+1})^c \; for \; c \; in \; C\}$
    Store the state transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
    Learning in each UAV-BS: $f_{AE\&VAS}(s_t, a_t, r_t, s_{t+1})$
  end for
  if $t$ mod $f$ == 0 then
    for each client $c \in C$ in parallel do
      send $\theta'_{t+1}$;
      receive ${\theta'_{t+1}}^c$;
    end for
    $\theta'_{t+1} \longleftarrow \sum_{c=1}^{C} \frac{1}{len(C)} {\theta'_{t+1}}^c$;
  end if
end for
End Function
```

Step 2: After making a choice, each UAV-BS will notify its neighbours of the state and local performance indicators. The agent will simultaneously listen to the other neighbours, and get ready to receive their states s^c_{t+1} and local performance metrics $\mathcal{M}^c$. A global system metric value that can direct each UAV-BS to take future actions will be formed when all metrics have been received and the reward has been calculated based on the reward function R_s.

Step 3: A replay batch $\mathcal{D}$ contains the data for each training stage. The tuple (s_t, a_t, r_t, s_{t+1}), or the current state, action, reward, and next state for a training step, is contained in each row of $\mathcal{D}$. For the purpose of updating the Q value model, samples will be chosen at random. The current state reward pairs will also be distributed to the other agents after each decision round.

Step 4: A UAV-BS will send the updated model results, θ', to its registered neighbours for model aggregation after it has reached the predetermined exchanging iteration. Each UAV-BS will simultaneously listen to its neighbours in order to receive models and service metrics instead of requesting models from the others, which results in a push-based communication mechanism to avoid interruptions from malfunctioning neighbours.

Step 5: Each node executes aggregation by averaging all updated models depending on the aggregation function, $\theta'_{t+1} \longleftarrow \sum_{c=1}^{C} \frac{1}{len(C)} {\theta'_{t+1}}^{c}$;, after receiving all the models from the registered neighbours.

Step 6: The updated model is used by the edge device to replace the outdated one and to carry out additional local training. We'll repeat the steps from above.

12.3.4 Complexity and Robustness Analysis

The Adaptive Exploration algorithm takes three parameters: r_p (prior reward), r_t (current reward), and ε (exploration probability). The function contains three conditional branches based on reward comparisons. Each branch contains constant time operations: setting ε to a constant value or updating it using multiplication. The time complexity of this function is constant, i.e., O(1).

The Action Selection algorithm takes three parameters: r_p (prior reward), r_t (current reward), and ε (exploration probability). It performs conditional branching based on reward comparisons and selects actions accordingly. The time complexity is O(1) since the operations inside each branch are constant.

Both AE and VAS algorithms have constant time complexities for their core functions. The overall complexity is dominated by the number of iterations in the main training loop, which is specified as N. Therefore, the overall time complexity of the algorithms is O(1) for each iteration, and O(N) for the entire training process.

The overall time complexity of the DecRL algorithm is influenced by the number of iterations N, the number of clients C, and the frequency parameter f. The transmission functions contribute a significant portion of the complexity, especially when considering parallel communication with each client. The model update and aggregation steps also have a complexity that depends on the number of clients and the frequency of model updates. The algorithm's complexity is not fixed and can vary based on the specific values chosen for parameters.

Furthermore, the Decentralized Reinforcement Learning (DecRL) algorithm's robustness against communication failures and potential UAV-BS malfunctions is underpinned by the push-based communication approach. This approach ensures dynamic adaptation as UAV-BSs actively listen to updates from the network, avoiding the situation that one needs to wait for the response from a malfunctioning edge, which can avoid accidental disconnections and maintain system engagement. In the face of communication disruptions, the push-based strategy facilitates adaptive reconfiguration, allowing UAV-BSs to adjust positions and communication parameters for continuous connectivity. Additionally, the monitoring function can be enabled by the push-based model aids in detecting UAV-BS malfunctions, prompting dynamic decision-making to redistribute tasks or optimize resource deployment.

12.4 Simulation Results and Analysis

In this section, the simulation configuration and scenario deployment are introduced firstly. We then investigate the impact of the 3-D location of multiple UAV-BSs on the performance of MC users in terms of backhaul link rate, throughput, and drop rate based on system-level simulations. Finally, we present the results of proposed RL algorithms for autonomous UAV-BS navigation.

12.4.1 Simulation Configuration

To evaluate the performance of the proposed RL algorithm in solving the formulated problem, we build a multi-cell scenario by considering the predefined system model, and a simulation is executed with a system-level simulator. With the output of the simulation, the proposed RL algorithm can be applied for UAV-BS to build a well-trained model, based on which optimal UAV-BS position and antenna configuration can be found rapidly.

In the simulation, we drop 500 users in the area as shown in Figure 12.2. The circle area with a 350m radius around the UAV-BS is defined as the MC area. The users located in the MC area are marked as MC users, while the

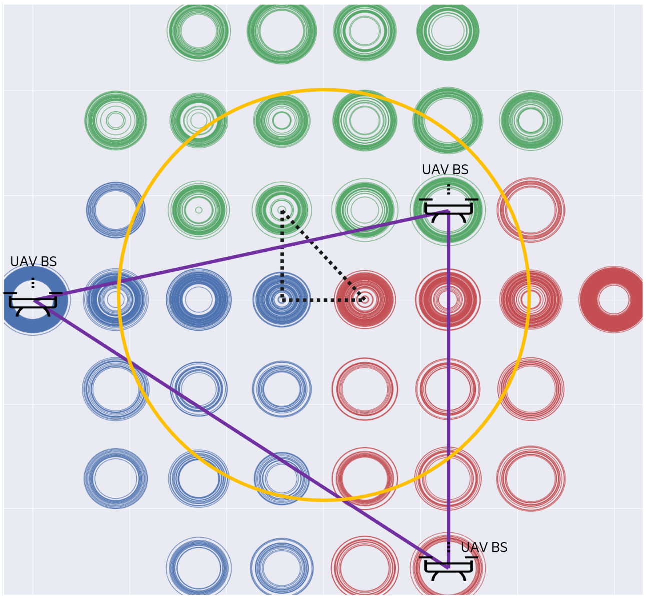


Figure 12.6: Impact of UAV-BSs' Positions on average backhaul link rate

others are normal users. All users follow an arrival model and only arrived users can be considered as activated.

To investigate how a well-trained RL model performs in a dynamic environment, we design a set of different user distributions to simulate the case of slow-moving users.

The detailed simulation parameters are shown in Table 12.2. Specifically, the typical inter-site-distance (ISD) is 500 meters for the mid-band urban-macro scenario in NR. To create a coverage hole and clearly show the impact of introducing UAV-BSs to cover bad-quality MC users, the ISD is set to 1000 meters and a macro-BS is removed from the center of the map due to malfunction. For the radius of the MC area, 350 meters is selected to create an area that is large enough for three UAV-BSs to jointly serve but not too large to introduce excessive candidate positions which significantly impacts the simulation efficiency. The user arriving rate per simulation area is used to control the system load in the network, and its value is selected by keeping the drop rate of the users in a reasonable range between 0% and 10%. The simulation time denotes the time duration between the network beginning and stopping to serve users with one set of configuration parameters, which comprises the 3D positions of three UAV-BSs. Hence, in this paper, one epoch means running a 2-second simulation with one specific combination of positions of three UAV-BSs. During this time, the users are activated based on a predefined arrival rate, as defined in the system model. The value of other parameters is selected based on the typical setting used in a mid-band urban-macro scenario.

12.4.2 System-level Performance Evaluation

To evaluate the impact of UAV-BSs' positions on the user performance, Figure 12.6 shows the range of achieved average backhaul link rates when the three UAV-BSs are deployed at all possible combinations of different candidate positions. Around the MC area denoted by orange circle, 40 candidate 2D-positions of UAV-BS (colored stars shown in Figure 12.2) are selected and mapped to the centers of the colored circles in Figure 12.6. Each colored UAV-BS can only be deployed at the position marked with the same color. The radius of each circle denotes the normalized average backhaul link rate of three UAV-BSs who are located at current positions. For one specific candidate position in blue where one UAV-BS is deployed, the other two UAV-BSs can be placed at all possible combinations of green and red candidate positions. This explains why there are multiple circles at each candidate position. Hence the smallest and largest circle radiuses at one candidate position denote the lower and higher bounds of the backhaul link rate when the UAV-BS is placed at the current location. If connecting the centers of three circles with the largest radiuses which represents the highest backhaul link rate, one triangle marked with a solid purple line can be observed. As shown in Figure 12.6, if three UAV-BSs are deployed at the vertexes of the purple triangle, the average backhaul link rate in the system is the highest. Based on the distribution pattern of three UAV-BSs, it seems that the optimal UAV-BSs' positions to maximize the backhaul link rate tend to be near the edge of the MC area. This is because the UAV-BSs can keep good backhaul link quality when located near the donor-BSs. Similarly, if three UAV-BSs are deployed at the vertexes of the black triangle, the lowest average backhaul link rate is reached.

Since the backhaul link rate decides the level of user throughput, the users can not be served well if the backhaul link quality is poor. Thus the UAV-

Table 12.2: Simulation Parameters

Parameter	Value
Carrier Frequency	3.5 GHz
Bandwidth	100 MHz
Duplex Mode	TDD
TDD DL/UL Configuration	DDUD
Inter-Site-Distance (ISD)	1000 m
Radius of MC Area	350 m
Number of BSs	Macro-BS: 6; UAV-BS: 3
BS Transmit Power	Macro-BS: 46 dBm; UAV-BS: 40 dBm
Noise Figure	7 dB
BS Height	Macro-BS: 32 m; UAV-BS: 10-300 m
Number of Sectors per Site	3
Number of MC&Normal Users	500
User Arriving Rate per Simulation Area	270 users/s
User Speed	3 km/s
Minimum Distance between BS and Users	30 m
Simulation Time	2 s

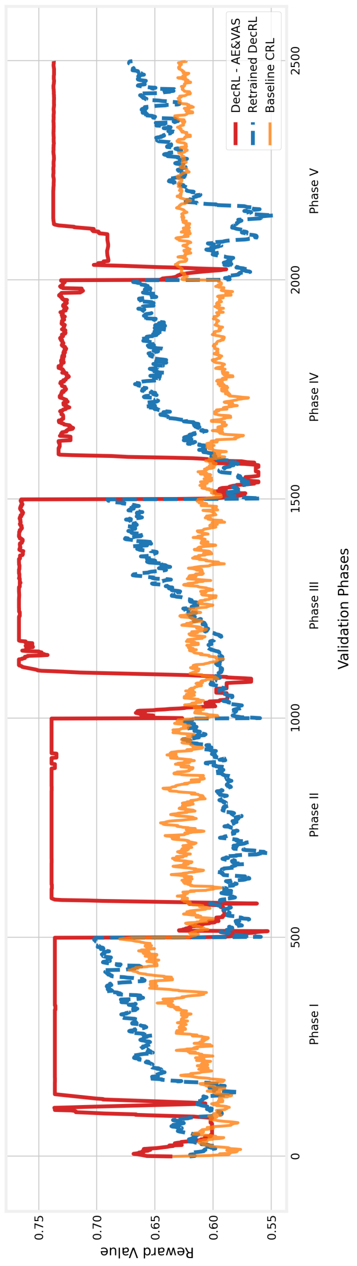


Figure 12.7: Reward value with the number of learning iterations in five consecutive validation phases

BS can not only pursue a high backhaul link rate while ignoring the user throughput, and vice versa. By considering the weight for each metric in the reward function, the proposed RL algorithm can provide optimal locations of UAV-BSs to achieve the highest reward value. For example, if the weight of 5th percentile throughput or drop rate of MC users is high, the UAV-BSs navigated by the proposed RL algorithm tend to move to locations where the performance of low-quality users can be improved as much as possible.

Based on the above observation, the optimal UAV-BSs' positions are different if different performance metrics are considered with different weights in the reward function. Hence, optimizing the performance metrics in the reward function to achieve global optimization by adjusting UAV-BSs' positions is a complicated problem, for which ML-based solutions can be applied to find the implicit structure from the collected data.

12.4.3 Machine Learning Performance Evaluation

In this section, we present the experimental results of Decentralized Reinforcement Learning with Adaptive Exploration and Value-based Action Selection (DecRL-AE&VAS) for autonomously controlling multiple UAV-BSs. For the UAV-BSs deployment, different from the single UAV case introduced in the previous section, more candidate positions are allocated around the MC area for the multi-UAV case. As shown in Figure 12.2, the available positions for each UAV-BS don't overlap with others, which means each UAV-BS covers a certain geographical area with a total of 18928 combinations. The result is evaluated using two criteria: (1) six features (specified in Section IV - Modeling of ML Environment) that demonstrate link service quality, and (2) model learning quality in each phase as demonstrated by system reward value. The results are compared to several baseline models. As we described before, the experiment contains five validation phases incorporating MC user mobility. When entering a new phase, the algorithms will use the data collected during the new phase to learn and update themselves. During the simulation, the DecRL-AE&VAS method trains the model from scratch in the training phase and then continuously improves itself in the succeeding validation phases. The previously learned experience will not be removed from the subsequent sessions. During the model learning, for each agent, the test bed provides an 8-core Intel Xeon Processor with 8 GB memory. The average CPU consumption is lower than $\sim 1\%$ while the memory usage is about 300 MB. The communication overhead of one agent within one epoch is $\sim$ 410 KB while the learning time of each epoch takes $\sim$ 0.3 second, which is suitable for quick response to real-time user movement and can be easily implemented on resource-constraint devices.

There are two baseline methods used to compare the performance of DecRL-AE&VAS and demonstrate the effectiveness of the decentralized architecture and the convergent efficiency in the training phase, namely, baseline CRL and baseline IRL method. For the baseline IRL algorithm, these baseline models are explicitly trained using each edge UAV-BS. There will not be any model or information exchange between the edge and central nodes during training, in contrast to decentralized reinforcement learning. The service quality performance can be compared to the decentralized reinforcement Learning model to demonstrate how it can outperform those locally trained individual models. For the baseline CRL, this baseline model is trained using the centralized learning strategy. Prior to model training, all data from the edge are collected into a single server, and learn the action strategy step by step.

In the validation phase, in order to prove the performance in a dynamic environment, two baselines are utilized, namely, the retrained DecRL and baseline CRL. The retrained DecRL algorithm removes the past information and randomizes the ML model parameters but with the same decentralized setup. When entering a new phase, the algorithm will retrain the model from scratch. The difference between retrained DecRL and DecRL-AE&VAS is the utilization of AE and VAS strategies. These strategies have been proven to be useful when deploying UAV-BSs into dynamic environments. Last but not least, the baseline CRL algorithm in the validation phases will constantly learn and employ the new data when entering the new phase but with a centralized algorithm that controls all deployed UAV-BSs but without improved strategies.

For the learning hyper-parameters of our method, we explored various sets of combinations using random search in order to achieve the best results. Random search efficiently explores a wide range of hyper-parameter combinations, making it more likely to discover effective configurations compared to a more constrained grid search. During the training, the network architecture for a DQN is set to $[4 - 16 - 16 - 81]$. The architecture choice balances the complexity of the model with the capacity to represent the relationship between state and action. The exploration probability decay is set to 0.995. The exploration probability decay determines how fast the exploration probability ϵ decreases over time. A high decay rate is selected in our case to allow the agent to explore more in the early stages of training and gradually shift towards exploitation as training progresses. The learning rate is set to 5×10^{-5}. The learning rate determines the step size during the weight updates in the training process. The selected learning rate strikes a balance between convergence speed and stability. Lastly, the number of learning iterations at the training phase and validation phases equals 1500 and 500, respectively. The number of iterations determines how much exposure the algorithm has to the training data. The values are selected to balance the learning time and the model quality.

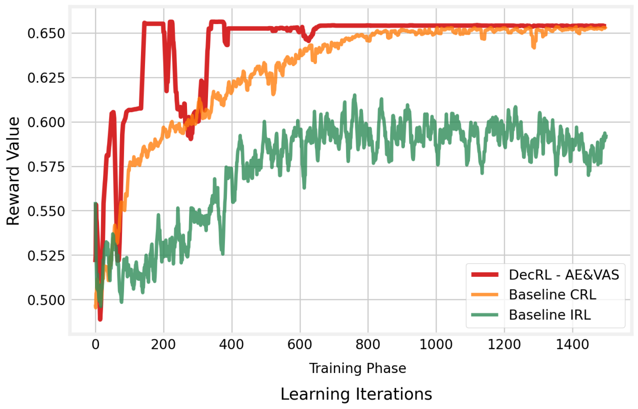


Figure 12.8: Reward value with the number of learning iterations in the training phase

We first analyze the algorithm's convergence performance during the model training phase. Figure 12.8 depicts the reward value as a function of the number of learning iterations. Compared to the baseline CRL algorithm, we can see that the VAS method can help the UAV-BS quickly discover the near-optimal position and converge at a high-quality level.

Training the algorithm in a decentralized way can also be converged and reach a high reward value before 400 training rounds. The convergent reward value of the DecRL-AE&VAS is approximately 5% higher than that of the centralized training method in the initial learning iterations which results in higher training efficiency. When compared to the independent learning

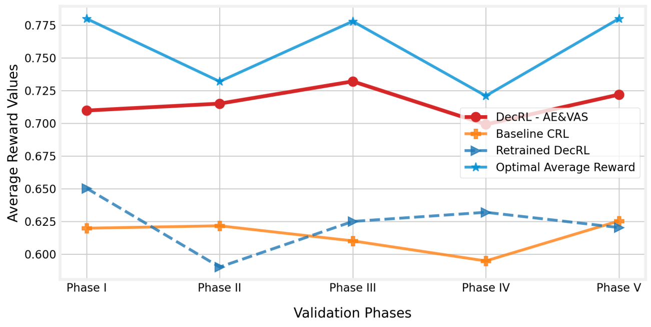


Figure 12.9: Average Reward Value in each validation phase

method, the algorithm does not converge after 1500 learning iterations. It is difficult for agents to share knowledge and collaborate since the algorithm stops them from exchanging information. Throughout the training procedure, the Baseline IRL algorithm performs poorly. Because the reward value represents the overall system performance of the three UAV-BSs, we can conclude that the DecRL-AE&VAS algorithm enables the UAV-BSs to provide the best wireless connection service when compared to the other two frequently utilized baseline approaches in the training phase.

When entering the validation phases (Figure 12.7 Phase I - V), the proposed algorithm can learn from the past and finally reach the optimal state that provides the highest reward value in the validation scenarios using the AE method. Even if the environment has changed and the quality has dropped dramatically, the algorithm can assist the UAV-BS in quickly adjusting and returning to ideal performance. When we look at the baseline CRL approach, the method failed to respond to environmental changes in a short period due to the slow-paced state exploration. When we compared the baseline results to the retrained DecRL and DecRL-AE&VAS, the results showed that our suggested VAS-AE approach can enable UAV-BS to quickly converge in most of the validation phases. However, with the retrained DecRL, a significant impact may occur on the algorithm and service stability if not using previous existing knowledge. When we integrated RL with adaptive exploration and the value-based action selection strategy, the algorithm showed the best performance in terms of convergence speed, the ability to adapt to environmental changes, and stable service quality of our suggested DecRL-AE&VAS method.

The average reward value changes during the validation phases are depicted

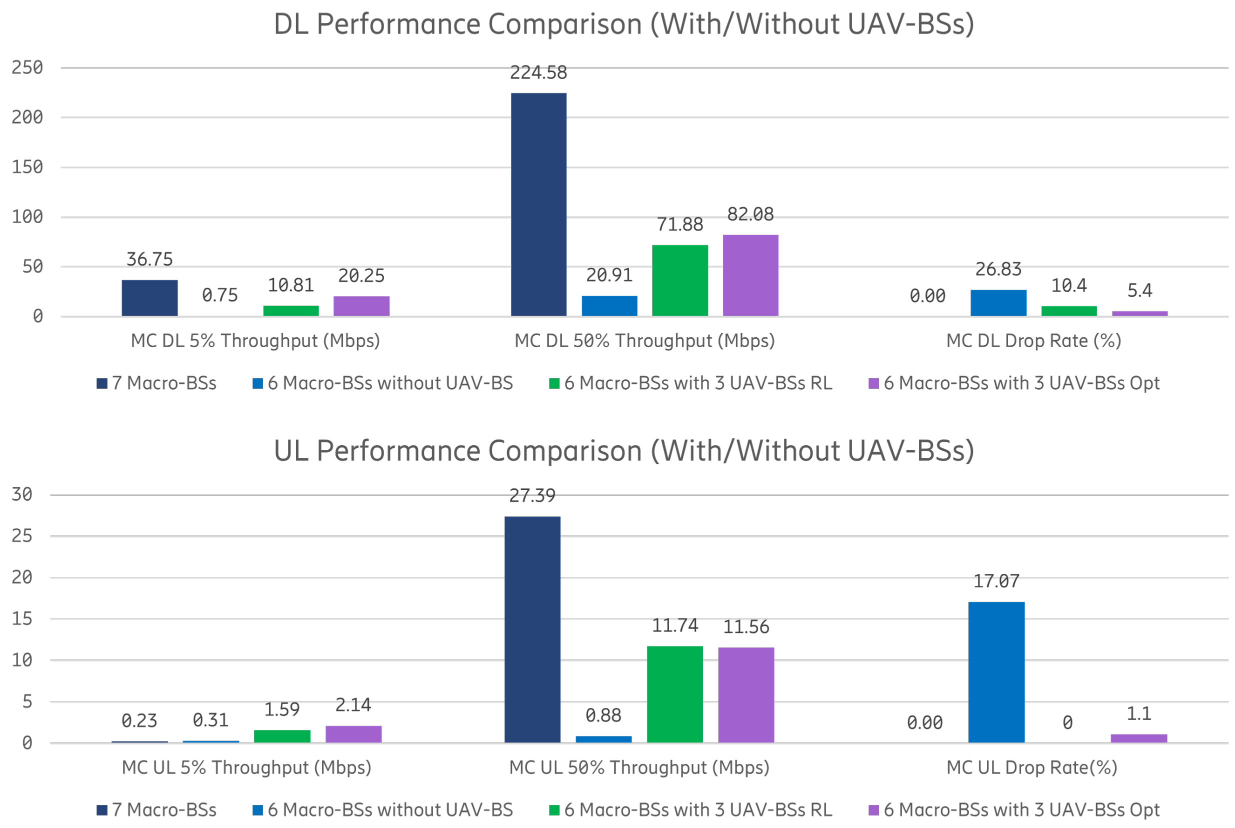


Figure 12.10: DL & UL performance comparison with/without UAV-BSs in the mission-critical use case

in Figure 12.9. Because the reward value encompasses all of the essential criteria that must be evaluated during the deployment of the UAV-BS, the value clearly demonstrates the model's quality at each stage. As shown in the figure, our proposed DecRL-AE&VAS method can assist the UAV-BS in maintaining the ideal service quality, but the baseline model failed to discover a state that can give reliable and satisfactory service to MC users. The suggested DecRL-AE&VAS algorithm gives a reward value (a weighted sum of the six assessed performance measures) of only about 5% to 6% less than the global best solution in each validation phase.

12.5 Discussion

In summary, in this paper, we show that when compared to the baseline RL approaches, the DecRL-AE&VAS model can efficiently assist the UAV-BS in finding the near-optimal location and converging at a high-quality level. Utilizing the model-sharing and information-exchange technique, the proposed algorithm can learn from the past and eventually arrive at the optimal state that provides the best reward value, the proposed DecRL-AE&VAS method can achieve the same or even higher levels of system quality than the Baseline CRL approach in both training and validation phases. Because the reward value shows the system's performance, we may conclude that the DecRL-AE&VAS can help UAV-BS provide better service than the other baseline models. Our findings show that the algorithm has the capacity to link MC users swiftly and adequately, allowing drone systems to self-learn without centralized interaction. In Figure 12.10, we also demonstrate the effectiveness after deploying multiple UAV-BSs in the MC use case. The "7 Macro-BSs" denotes the scenario where 7 marco-BSs are serving the whole scenario, while the "6 Macro-BSs without UAV-BS" denotes the scenario where one macro-BS is broken and no UAV-BS is deployed. The case "6 Macro-BSs with 3 UAV-BSs RL" describes the situation which three UAV-BSs are deployed to fill the coverage hole created in case "6 Macro-BSs without UAV-BS". The results in case "6 Macro-BSs with 3 UAV-BSs Opt" is derived based on grid search to be compared with the RL case. It can be observed that when one macro-BS breaks down, the MC users experience severe performance degradation. Compared with the "6 Macro-BSs without UAV-BS" case, deploying 3 UAV-BSs as in the case "6 Macro-BSs with 3 UAV-BSs RL" can improve about

80% system performance (including 5%, 50% throughput and drop rate) for MC users in both DL and UL. Furthermore, the suggested DecRL-AE&VAS algorithm gives a throughput of 10 Mbps and a drop rate of only about 2% to 3% less than the global best solution indicated in the "6 Macro-BSs with 3 UAV-BSs Opt" case. In conclusion, deploying UAV-BS can greatly improve the performance of MC users who are experiencing coverage loss. However, due to the BS capability difference, deploying 3 UAV-BS, in this case, can not guarantee that the MC users have similar performance in the case when no emergency happens. The only exception is that the UL 5% throughput performance of MC users served by 3 UAV-BSs is better than that served by one macro-BS before the disaster happens. This is because three UAV-BSs can be deployed in dispersed locations which increases the probability for an MC user to be close to its serving BS and the UL 5% throughput performance of MC users is improved accordingly. The 5th percentile, 50th percentile, and drop rate of normal users are also investigated, but they are nearly not impacted by broken macro-BS and newly deployed UAV-BSs. That is because most normal users are not served by the broken macro-BS before the malfunction. In some cases, deploying UAV-BSs even leads to lower 5th or 50th percentile throughput of normal users compared to the case where there are no UAV-BSs, due to the introduced interference from UAV-BSs.

The proposed model, initially designed for UAV-based communication in mission-critical scenarios, holds promise for adaptation and scalability across diverse disaster types and alternative use cases. Examining these dimensions is pivotal for the model's real-world applicability beyond the specific disaster context considered in this study. When considering different disaster types, such as natural disasters, man-made incidents, and public health emergencies, it is imperative to assess the adaptability of the model. Varied communication needs and environmental challenges associated with each type of disaster may require nuanced adjustments to the model's parameters and strategies. By understanding how the model can flexibly adapt to a range of disaster scenarios, its practical utility can be enhanced.

12.6 Limitations

While our proposed deep reinforcement learning algorithm for the three-dimensional placement of UAV-BSs in MC scenarios demonstrates promising results, it is crucial to acknowledge several limitations and challenges associated with the decentralized nature of the approach. The utilization of decentralized architecture and model delivery mechanism introduces unique considerations that require further research.

Communication Overhead: The independence of edge entities in completing tasks introduces communication overhead among the UAV-BSs. As each UAV-BS makes decisions based on local observations and interacts with others, the need for frequent information exchange arises. This can lead to increased communication latency and potential congestion in scenarios with a large number of UAV-BSs. The need for frequent information exchange arises and follows equation $n^2 + n$ where n equals the number of participated edge entities. This can lead to increased communication latency and potential congestion in scenarios with a large number of UAV-BSs. In our paper, we have parameters to control the number of UAV-BSs communicated in each learning iteration and the frequency of model exchange. However, the trade-off between the control parameter and the quality of the model requires further research.

Coordination Complexity: The decentralized approach requires efficient coordination mechanisms among UAV-BSs to avoid conflicts and optimize their collective behavior, especially when the number of participated UAV-BSs is largely scaled up. Ensuring seamless interaction without a central orchestrator poses challenges, particularly in dynamic environments where the on-ground user movement and network conditions may change rapidly.

In conclusion, while our approach showcases the benefits of decentralized control in UAV-BS placement, the outlined limitations underscore the need for further research and development. Addressing these challenges will be crucial for the practical implementation of our proposed solution in real-world scenarios, particularly in dynamic environments that require a large number of UAV-BSs deployed.

12.7 Conclusions and Future Work

In this paper, we presented a data collection system and machine learning applications for an MC use case, a novel RL algorithm, as well as a decentralized architecture to autonomously pilot multiple UAV-BS in order to offer users temporary wireless access. Two novel strategies, i.e., adaptive exploration and value-based action selection, are developed to help the proposed RL algorithms work efficiently in a dynamic real-world context, incorporating MC user movements and a decentralized architecture to support multi UAV-BSs deployment. Note that the number of participated UAV-BSs can be further extended based on the industrial requirements due to the characteristics of the decentralized architecture. We show that the proposed RL algorithm can monitor the MC service performance and quickly respond to environmental changes via self-adapting exploration probability. In addition, it requires far fewer model training iterations by reusing previous experiences and the value-based action selection strategy. Therefore, the proposed method can well serve the MC users by autonomously navigating multiple UAV-BSs despite environmental changes.

In the future, we will consider separating the configuration for access and backhaul antennas of the UAV-BS, as well as modelling drone rotation in the horizontal domain as an additional parameter for the UAV-BS configuration. We also intend to examine other hyper-parameters and reward function combinations based on different service requirements and further refine the model to accommodate specific real-map scenarios. Last but not least, we will also investigate the energy efficiency problem for deploying UAVs with machine learning components in the real-world context.

CHAPTER 13

Enabling Efficient and Low-Effort Decentralized Federated Learning with the EdgeFL Framework

This chapter has been submitted as:
Enabling Efficient and Low-Effort Decentralized Federated Learning with the EdgeFL Framework
Zhang H., Bosch J., and Olsson H. H.
In *Information and software technology*, p.107600.

13.1 Introduction

Federated Learning (FL), a revolutionary machine learning approach, enables model training across decentralized data sources while upholding the fundamental concepts of data privacy and security. A frequent technique in traditional machine learning is to centralize all relevant data into a single location for model training. However, this traditional strategy frequently confronts formidable obstacles. These obstacles cover a wide range of issues, from data privacy concerns and severe regulatory requirements to computationally complex demands. To address these restrictions, FL provides a novel approach by allowing models to be trained locally on the devices or servers from which the data originated, eliminating the need to send data to a central repository. [99].

The most significant advantage of FL is its data privacy preservation. This strategy assures that data remains on client devices or servers, eliminating any concerns or suspicions about data leakage or the unintended exchange of critical information [57]. Instead of sending raw, unaltered data, FL relies on the communication of only model parameter changes, which are frequently encrypted or anonymised. This decentralized training paradigm empowers organizations, researchers, and individuals to form collaborations in the pursuit of model improvements without sacrificing the integrity and security of their data resources.

In addition to data privacy protection, FL adds a new level to collaborative model training. The approach extracts collective knowledge that is more powerful than the sum of its individual parts by encapsulating data and knowledge scattered across several nodes or devices [245]. The federated method supports the integration of various sources of data, resulting in the construction of models that are not only more accurate but also robust, which can be adaptable to a wide range of real-world scenarios.

FL can be applied to applications in various domains, such as healthcare, finance, automotive, Internet of Things (IoT), and edge computing [116][132][126][246][247][248]. One of its advantages is its ability to enable enterprises to exploit the collective knowledge contained within distributed datasets without violating severe privacy requirements or exposing sensitive information to unnecessary risks. This capacity to combine knowledge while adhering to data privacy regulations is especially important in fields such as healthcare and finance, where patient or financial data confidentiality is crucial [249]. Furthermore, FL appears as a powerful tool for reducing reliance on costly and bandwidth-intensive data transfers.

13.1.1 Challenges of FL development

However, while FL offers significant advantages, it also poses distinctive challenges for AI engineers, as highlighted in the research by Lwakatare et al. [151]. These challenges revolve around the intricate design and implementation of distributed systems. Engineers are tasked with creating systems capable of efficiently managing communication, coordination, and synchronization between numerous clients and a central server. This involves overcoming challenges such as scalability and fault tolerance to accommodate extensive deployments [250].

Existing FL frameworks, whether from academia or industry, tend to be complex to use. FL involves various components like distributed systems, optimization algorithms, privacy techniques, and machine learning models [251][252]. When these elements are combined into a single framework, it can result in intricate systems with numerous dependencies and configurations. This complexity can be challenging for users, especially those without a strong background in distributed systems or machine learning [253].

Additionally, FL is applied in diverse domains and use cases, each with its unique requirements and constraints [254]. Designing a platform or framework that can accommodate this diversity can add to the complexity. It becomes a challenge to strike a balance between providing flexibility for customization and maintaining simplicity for ease of use [255]. As indicated in previous research [136], software engineers must understand and implement FL algorithms and optimization techniques such as Federated Averaging and adaptive learning rate strategies. The development of algorithms that efficiently converge towards high-quality models while consuming the least amount of resources represents an overwhelming task that software engineers must face within the context of FL. These diverse duties highlight the complexities of FL and the skill set required for its successful implementation [256].

Last but not least, software engineers may face difficulties because of the lack of comprehensive tools and frameworks designed expressly for FL. To meet their special objectives, they may need to adapt current tools, create customized solutions, or contribute to open-source initiatives [62][257]. Constructing simplified workflows, putting in place effective debugging mechanisms, and creating monitoring tools that fit FL systems are all challenging tasks that require careful attention and expertise. This diverse landscape emphasizes the complexities of integrating FL with edge devices and IoT contexts, where resource efficiency and adaptability are critical.

In summary, the challenges can be concluded into four aspects:

- Scalability and Decentralization in FL: Existing FL frameworks predominantly rely on centralized aggregation, which introduces a single point of failure and limits scalability. The challenge lies in designing a decentralized architecture that can effectively eliminate the central server while maintaining model accuracy and reducing communication overhead.

- Customization and Flexibility in Aggregation Functions: Current FL frameworks often lack the flexibility needed to customize aggregation functions to suit specific use cases. The gap here involves developing an approach that allows for adaptable aggregation functions, which can be fine-tuned to optimize performance across different domains and data distributions.
- Efficient Model Evolution and Latency Reduction: The latency associated with weight updates and model evolution in FL systems has been a significant bottleneck. The gap is in devising methods that can reduce this latency while ensuring rapid model convergence, especially in scenarios with unequally distributed datasets.
- Complexity of Implementation: Implementing FL systems is often complex due to the need to integrate multiple components such as distributed systems, optimization algorithms, and privacy techniques. The challenge lies in simplifying this implementation process to make FL more accessible to engineers without deep expertise in these areas.

13.1.2 Contributions

To tackle these challenges, in this paper, we introduce EdgeFL, an efficient and low-effort FL framework tailored for edge computing environments. EdgeFL addresses the challenges associated with centralized aggregation by adopting an edge-only model training and aggregation approach. This eliminates the need for a central server and allows for seamless scalability across various use cases. The framework offers a simple integration process, requiring only four lines of code (LOC) for software engineers to add FL functionalities to their AI products. Moreover, EdgeFL supports the customization of aggregation functions, giving engineers the flexibility to adapt them to their specific needs. Therefore, the contributions of this paper are as follows:
1) We identify the challenges by evaluating existing FL frameworks in six categories, including line of code, system design, architecture settings, aggregation type and deployability.

Table 13.1: Comparison between existing FL frameworks and our proposed EdgeFL

	TFF	PySyft	FATE	LEAF	PaddleFL	EdgeFL[1] (Proposed Framework)
Line of Code (LOC) for FL Feature	~50	~150	~100	~350	~100	**4**
Centralized Server required	Yes	Yes	Yes	Yes	Yes	**No**
Asynchronous Node Join	No	No	No	No	No	**Yes**
Heterogeneous Environment Support	No	No	No	No	Limited	**Yes**
Customizable Aggregation	No	No	No	No	No	**Yes**
Containerized Edge Deployability	No	No	Limited	No	Limited	**Yes**

2) To tackle these challenges, we introduce EdgeFL[1], a scalable and low-effort edge-only FL framework. To accomplish easy-implementation and scalable model training capacity, simple API design and learning flow abstraction are built.
3) We propose a decentralized FL architecture and learning method that enables asynchronous model training to speed up industrial FL training and support large-scale node communication along edges. The architecture can be used as a template for future edge-only FL development and research.
4) We validate EdgeFL using two well-known datasets, namely MNIST and CIFAR-10 and compared them with five existing FL frameworks/algorithms. Furthermore, a real-world case, the distributed sentiment analysis on the IMDB Dataset of 50K Movie Reviews, is constructed and used to validate the empirical deployment of the EdgeFL framework.

The remainder of this paper is structured as follows. In Section II, we introduce the background and related work of this study. Section III details our research method, including the implementation, data distribution, machine learning methods applied and evaluation metrics. Section IV presents the system design of our proposed EdgeFL. Section V evaluates our proposed framework and compares it with existing FL frameworks. Section VI outlines the discussion on our observed results. Finally, Section VII presents conclusions and future work.

13.2 Background and Related Work

13.2.1 Federated Learning

FL is a machine learning approach that enables multiple decentralized devices or entities to collaboratively train a shared model without sharing their raw data [51], [137], [145]. In traditional machine learning methods, a central server or data aggregator gathers and stores training data from different sources and then trains a global model using this combined data. However, this centralized approach raises concerns related to data privacy, security, and the practicality of transferring large data volumes to a central location [136].

In the context of the growing use of edge computing and distributed data sources, there is an increasing need to utilize data that is distributed across

[1]https://github.com/HarryME-zh/EdgeFL.git

various devices or entities while respecting data ownership and privacy [258]. FL addresses this challenge by allowing local devices, such as smartphones, IoT devices, or edge servers, to train a shared model using their locally stored data [259]. This process is illustrated in Figure 13.1 and typically involves the following steps:

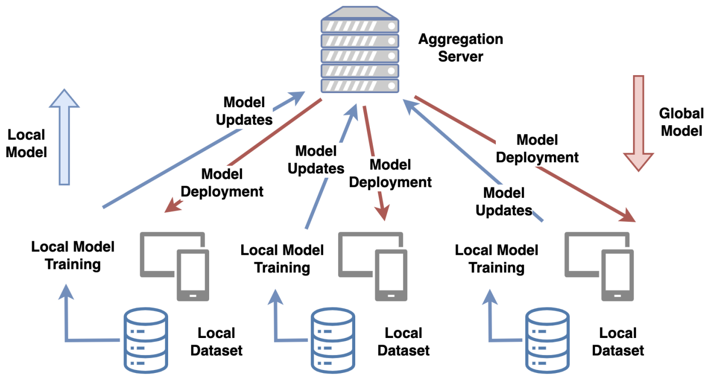


Figure 13.1: Diagram of FL training process

1) Initialization: In the first step, the central server kicks off the process by initializing a global model and distributing it to the participating devices or entities.

2) Local Model Training: Each device or entity then takes the initiative to train the global model using its own local data independently. Importantly, this is done without sharing the raw data. The local model undergoes training using techniques like gradient descent or other optimization methods. This involves updating the model parameters based on the device's individual data.

3) Model Aggregation: Following the local training phase, the devices or entities transmit their locally computed model updates, which can include gradients, to the central server. The central server collects and aggregates these updates using methods such as Federated Averaging or Secure Aggregation, resulting in an improved global model.

4) Iterative Process: The model training and aggregation process iterates over multiple rounds, allowing the global model to improve over time. Each round typically includes local model training, model aggregation, and communication between devices and the central server.

FL utilizes the power of collaborative learning from a diverse range of devices or entities, each with its unique data distribution and characteristics. This diversity plays a crucial role in enhancing the global model's generalization and robustness by capturing a more comprehensive representation of the data.

13.2.2 Existing FL Frameworks

There are several existing FL frameworks available that facilitate the development and deployment of FL systems:

1. TensorFlow Federated (TFF): Google's open-source research-centric framework that is designed to extend TensorFlow's capabilities into the field of FL. It excels in providing a flexible programming model and a rich set of APIs, making it a strong choice for researchers experimenting with FL algorithms. [260]. However, TFF's focus on research and simulation may limit its practical deployment in industrial settings, particularly in edge environments where scalability and ease of deployment are crucial.

2. PySyft: PySyft focuses on privacy-preserving FL and secure multi-party computation and is built on top of PyTorch. Its high-level API is excellent for leveraging differential privacy and secure aggregation, making it suitable for applications where data privacy is important. PySyft facilitates collaborative learning with remote data and models while preserving privacy [261]. Nevertheless, PySyft's complexity in setting up and its emphasis on privacy might make it less straightforward for developers who prioritize ease of use and broad scalability.

3. FATE: FATE is an industrial-grade FL framework developed by Webank Research, that provides a secure and adaptable infrastructure for collaborative model training across distributed entities. This framework places a strong emphasis on data privacy. It is particularly well-suited for industries like finance and healthcare that require strict data privacy controls. However, FATE's comprehensive feature set and its focus on data privacy can add to its complexity, potentially making it challenging to integrate with simpler, lightweight FL applications, especially in edge computing scenarios. [262].

4. LEAF: LEAF, developed by NVIDIA, is an experimental FL framework that provides tools for evaluating FL algorithms. While LEAF is valuable for benchmarking and experimenting, it lacks the production-ready features required for deploying FL in real-world, large-scale environments. The absence of strong support for edge deployments and limited flexibility in customization can be considered as weaknesses in comparison to our proposed EdgeFL framework. [263].
5. PaddleFL: PaddleFL is a deep-learning framework that supports large-scale FL models with a focus on distributed infrastructure. It offers various optimization algorithms and communication protocols, making it a strong candidate for large-scale deployments. However, PaddleFL may not be as straightforward to implement in heterogeneous environments or where the need for custom aggregation functions is critical. [264].

Compared to the existing frameworks, EdgeFL is designed specifically to address practical challenges that are not fully resolved by the above-mentioned frameworks:

- Scalability and Ease of Integration: Unlike TFF and FATE, which may be complex to deploy in large-scale, real-world applications, EdgeFL is designed to be lightweight and easily integrable, requiring only four lines of code to implement.
- Decentralization and Edge Suitability: While frameworks like PySyft and FATE emphasize privacy, they often rely on centralized components for aggregation, which can be a bottleneck. EdgeFL eliminates the need for a central server, enhancing scalability and making it more suitable for edge environments where decentralized operations are necessary.
- Customization and Flexibility: Unlike PaddleFL and LEAF, which may have limited support for custom aggregation functions and edge deployments, EdgeFL offers customizable aggregation mechanisms and supports asynchronous node joining, making it adaptable to a wide range of use cases.

While many decentralized FL models discussed in the literature focus primarily on aggregation algorithms and model training strategies [265][266][258], EdgeFL distinguishes itself by offering a comprehensive, implementation-ready framework. Unlike these models, which often lack complete infrastructure support, EdgeFL provides a robust API and end-to-end infrastructure that facilitates node communication, decentralized learning, and real-world deployment. This focus on practical application and ease of integration makes EdgeFL a valuable tool for software engineers, allowing them to implement FL systems tailored to their specific requirements, while also addressing challenges related to scalability, fault tolerance, and efficient model updates.

13.2.3 Problem Description

Despite the various functionalities of these FL frameworks, the current FL frameworks available in academia and industry are often complex to use, demanding a thorough understanding of FL concepts. The majority of these systems demand a deep understanding of FL concepts and require the implementation of over 100 lines of code (LOC) to deploy an FL application. This often limits flexibility in terms of customizing aggregation functions and lacks support for asynchronous communication schemes, as summarized in Table 13.1. These constraints raise considerable challenges for software engineers aiming to seamlessly integrate FL into production environments.

Moreover, the complexity of existing FL frameworks extends beyond just the initial deployment phase. Maintenance and adaptation become obstacles as well. With numerous dependencies and configurations, updating or modifying an existing FL system can be a daunting task [253][250]. This complexity not only increases the risk of introducing errors or vulnerabilities but also impedes the agility needed to respond to evolving requirements or emerging research advancements in the FL domain.

Additionally, the steep learning curve associated with these frameworks can deter potential adopters, particularly those without a strong background in distributed systems or machine learning [267]. The intricate interplay between various components such as distributed systems, optimization algorithms, and privacy techniques necessitates a deep understanding of FL concepts, making it challenging for newcomers to navigate and harness the full potential of FL technology.

Furthermore, the lack of standardization across existing FL frameworks exacerbates the problem. Each framework may have its own set of APIs, data formats, and communication protocols, further complicating interoperability and hindering the development of reusable components or libraries [267]. This fragmentation not only fragments the community but also impedes collaboration and knowledge sharing, slowing down the overall progress in the field of FL.

13.3 Research Method

In this research, we adopted the empirical methodology and learning procedure outlined by Zhang [92] to conduct a comprehensive quantitative measurement and evaluation of our proposed EdgeFL framework in comparison to existing FL frameworks [66]. The empirical methodology is particularly well-suited for evaluating the performance and practical utility of software frameworks. It allows for systematic observation, measurement, and comparison in controlled environments, ensuring that the results are both robust and generalizable. The methods we employed includes a combination of experiments and a case study [268][269][80]. These methods were chosen as they offer a comprehensive way to evaluate the performance and practicality of EdgeFL in both controlled and real-world settings. Firstly, we conducted a series of controlled experiments to systematically evaluate EdgeFL's performance against existing FL frameworks. The experiments were designed to simulate real-world conditions, focusing on key metrics such as accuracy, communication overhead, computational efficiency, and latency. These experiments allowed us to quantitatively measure and compare the scalability, efficiency, and ease of use of EdgeFL and its competitors. The choice of these metrics is critical because they directly reflect the core challenges in FL deployment, particularly in edge computing environments. To complement the experimental findings, we implemented a case study involving a real-world machine learning task—sentiment analysis on the IMDB dataset. The case study was chosen to demonstrate how EdgeFL performs in a practical application, highlighting its integration process, usability, and overall effectiveness in a real-world scenario. The combination of these methods provides a robust evaluation process. The experiments offer precise, repeatable measurements in a controlled setting, while the case study provides practical insights into EdgeFL's applicability and effectiveness in real-world situations. Together, they ensure that our findings are both rigorous and practically relevant. In the subsequent sections of this paper, we present the details of the selection process and criteria for the selected frameworks. We provide a detailed insight into our implementation approach. This includes our methodology for dataset partitioning and distribution, which is crucial for conducting heterogeneous simulations. We also present the metrics that were utilized to evaluate the performance of our framework. Additionally, we provide an in-depth exploration of the machine-learning techniques employed during our experimental analysis.

13.3.1 Search Process

We employed a structured search strategy to identify all existing FL frameworks. Initially, we conducted a literature review, searching academic papers, conference proceedings, technical documentation, and industry reports related to FL technologies [270]. Key search terms included "Federated Learning Frameworks," "FL Platforms," "Distributed Machine Learning," and "Collaborative Learning Systems." This search strategy aimed to capture a diverse range of frameworks used in both research and industry settings.

Building upon insights gained from the literature review, we formulated criteria and metrics to guide the evaluation process. These criteria were designed to assess the functionality, privacy preservation capabilities, scalability, ease of use, and adoption/community support of FL frameworks but also the comprehensiveness of their infrastructure and implementation readiness. Those frameworks demonstrating robust features, privacy preservation mechanisms, scalability, user-friendliness, substantial adoption in research or industry, and integration with other application development and implementation environments were prioritized for further evaluation [271]. Additionally, emphasis was placed on evaluating frameworks that provided a complete infrastructure with APIs that allow software engineers to implement FL solutions tailored to specific requirements, including node communication, decentralized learning, and deployment. By considering these key aspects, we aimed to ensure that the selected frameworks would be suitable for addressing the research objectives effectively and the comparison with our proposed EdgeFL framework.

13.3.2 Implementation

To comprehensively assess the performance and capabilities of the EdgeFL framework, we conducted a series of experiments using two machine learning applications: digit recognition and object recognition. For these experiments, we leveraged the MNIST and CIFAR-10 datasets, which are extensively applied in the research field. To enable deep learning training and testing, the PyTorch backend is utilized. One of the features of the EdgeFL framework is its simplicity in integration. With the integration of EdgeFL, the FL functionality is seamlessly integrated into these machine-learning applications with the addition of only four lines of code (LOC). Furthermore, we containerized the applications, enabling easy deployment on edge devices while maintaining their functionality and performance.

The flexibility of the EdgeFL framework allows it to be constructed on various container orchestration clusters, such as Kubernetes, Docker Swarm, etc. For the purposes of this study, we adopted Docker Swarm as our preferred cluster management solution [272]. Docker Swarm provides an efficient and scalable environment for handling containerized applications. The services within Docker Swarm facilitate smooth communication among containers, and an internal DNS resolver ensures seamless peer node service communication. By leveraging Docker Swarm's capabilities, we were able to establish a robust and scalable deployment environment for the EdgeFL framework, ensuring its suitability for edge devices and distributed computing scenarios.

13.3.3 Dataset Distribution

For the purpose of this study, we used two kinds of edge data distribution to analyze system performance for heterogeneous simulation.

13.3.3.1 Uniform Distribution

Within this experimental setup, the training data samples were distributed among the edge nodes following a uniform distribution. This distribution ensured an equal likelihood of data samples from each target class.

13.3.3.2 Normal Distribution

Within this configuration, the number of samples in each class within each edge node follows a normal density function. Mathematically, this can be expressed as:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

where μ and σ are defined as:

$$\mu = \frac{k \times N}{K},\ \sigma = 0.2 \times N$$

In the above equations, k represents the ID of each edge node, K denotes the total number of edge nodes, and N corresponds to the total number of target classes in the training data. This configuration aims to provide varied distributions and different numbers of samples among different edge nodes, allowing each class to have a probability of having the majority of samples in a specific node.

13.3.4 Machine Learning Method

The implementation of the models in this study utilized Python and relied on the following libraries: torch 1.6.0 [153], torchvision 0.7.0 [154], and scikit-learn [155], which were applied in model construction and evaluation.

To achieve satisfactory classification results, two distinct convolutional neural networks (CNN) [156] were trained for the MNIST and CIFAR-10 datasets. For the MNIST application, the CNN architecture comprised two 5x5 convolutional layers (with 10 output channels in the first layer and 20 in the second), each followed by 2x2 max pooling. Additionally, a fully connected layer with 50 units employing the ReLU activation function and a linear output layer were included.

For the CIFAR-10 application, the CNN architecture featured four 5x5 convolutional layers (with 66 output channels in the first layer, 128 in the second with a stride of 2, 192 in the third, and 256 in the fourth with a stride of 2). Furthermore, two fully connected layers utilizing the ReLU activation function, with 3000 and 1500 units respectively, were incorporated along with a linear output layer.

13.3.5 Evaluation Metrics

To assess the effectiveness of EdgeFL, three key metrics were selected: weights update latency, model evolution time, and model classification performance [271][273]. The choice of these metrics is grounded in their ability to provide comprehensive insights into the performance and efficiency of the system while addressing the characteristics of EdgeFL's decentralized architecture.

13.3.5.1 Weights update latency

Weights update latency was chosen as a primary metric due to its direct relevance to the network and communication aspects of the FL process. In traditional FL systems using centralized architectures, where a central aggregation server collects and processes model updates, the measure of latency is relatively straightforward. However, in the case of EdgeFL, which adopts a decentralized architecture, the aggregation function is moved to the edge, and a peer-to-peer model update exchange is facilitated. Measuring the time it takes for these updates to be transmitted across edge nodes is a key indicator of the system's efficiency and responsiveness.

This metric offers insights into the network conditions, communication overhead, and overall efficiency of different FL architecture options. It helps determine how quickly the system can adapt to new data and how effectively it can disseminate model updates. By calculating the average weights update latency across all edge nodes during a training round, EdgeFL's performance in a dynamic, decentralized environment is effectively evaluated. This metric is measured by comparing the sending and receiving timestamps in all model receivers, shedding light on the speed at which EdgeFL can accommodate changing data patterns.

13.3.5.2 Model Evolution time

Model evolution time was included as a critical metric to gauge the speed at which local edge devices can adapt and update their knowledge. In the context of FL, it's essential for systems to quickly evolve their models to adapt to rapidly changing environments or data distributions. This is especially crucial for real-time applications and industries where up-to-date information is important.

Similar to weights update latency, the average model evolution time across all edge nodes during one training round is measured. This metric provides insights into the responsiveness of EdgeFL, indicating how swiftly the deployed models at the edge nodes are updated. By examining the timestamps of model deployment, the system's ability to keep pace with evolving data patterns is effectively quantified.

13.3.5.3 Model Classification Performance

Model classification performance serves as a fundamental metric to assess the quality of the trained model, a vital consideration in machine learning applications of our testbed.

This metric evaluates the percentage of correctly recognized images among the total number of testing images. The evaluation is performed on each edge device using their updated models, ensuring that the test sample distribution aligns with the training samples. The average classification performance across all edge nodes is reported, offering a comprehensive measure of how well EdgeFL's decentralized learning approach fares in terms of model quality and accuracy.

13.4 System Design of EdgeFL

In this section, we offer a complete and in-depth explanation of the system design of EdgeFL, providing a full insight into its architectural design and operational structure. This part also includes a thorough presentation of the extensive set of APIs and functions available to EdgeFL users. It goes into the EdgeFL learning life-cycle, explaining the steps and processes that define its functioning, and providing a comprehensive understanding of the framework's usefulness and adaptability.

13.4.1 System Design

Within the FL process, EdgeFL provides a seamless path to achieving scalability, fault tolerance, and customised flexibility. This framework is made up of two main components: FL Edge Nodes and Registration Nodes, inspired by the architectural principles that underpin successful systems like Skype [274]. The FL Edge Nodes allow users to connect directly, thus minimizing the reliance on a central server for data transmission. In EdgeFL, the FL Edge Nodes collaborate seamlessly, harnessing their collective computing power and data while preserving the privacy of each individual contributor. Parallel to this, the Registration Nodes act as critical coordination centers, facilitating the connection and interaction of FL Edge Nodes. They enable the decentralized architecture of EdgeFL by providing the necessary mechanisms for Edge Nodes to identify and communicate with each other independently.

This decentralized architecture enables Edge Nodes to identify and interact with one another independently, enabling a durable and flexible ecosystem for FL processes.

- FL Edge Nodes: The FL Edge Nodes, which are strategically deployed on edge devices, play an important function within the FL framework, with each Edge Node serving as an active participant contributing to the efficacy of the FL process. These nodes play an important role in executing the FL training process, permitting model exchanges with other nodes, and performing critical local model updates. In practice, the FL Edge Node code is developed using the Flask framework, which is well-known for its efficiency and dependability. This code architecture provides machine-learning model files on demand, enabling smooth and efficient interactions across all peer nodes.
- Registration Nodes: Registration Nodes serve as essential coordinators for participated FL edge nodes. Their primary responsibility is to manage a catalogue of active peers, as well as to provide important services for the registration, unregistration, and retrieval of relevant peer information. With their functionalities, these Registration Nodes act as catalysts, allowing FL edge nodes to discover and engage with one another in a fully decentralized manner. The registration nodes are built on a solid architecture with the Flask framework, which is well-known for its dependability and efficiency. This infrastructure exposes a variety of APIs, each precisely designed to streamline and improve the FL process, hence strengthening the framework's operational efficiency and effectiveness.

13.4.2 APIs and Services

Table 13.2 summarises the most important APIs and services for EdgeFL, including edge node registration and registration, peer information retrieval and model file serving.

- Registration API: FL Edge Nodes initiate the registration process by sending a registration request via this API to the Registration Nodes. The hostname of the FL Edge Node is included in the request. After successful registration, the Registration Nodes add the newly added peer information to their lists of active peers.

Table 13.2: APIs and services of EdgeFL

Name	Endpoint	Method
Registration API	/register	POST
Unregistration API	/unregister	POST
Peer Information Retrieval API:	/peers	GET
Model File Serving API	/latest_model	GET

- Unregistration API: When FL Edge Nodes stop participating in the FL process, they use this API to send unregistration requests to the Registration Nodes. The hostname of the outgoing FL Edge Node is specified in these requests. In reaction, the Registration Nodes remove the related peer information from their lists of active participants immediately.

- Peer Information Retrieval API: FL Edge Nodes can employ this API to query the Registration Nodes for the list of active peers. The Registration Nodes promptly respond with the desired active peer information. This functionality empowers FL Edge Nodes to seamlessly discover and establish communication channels with their peers.

- Model File Serving API: FL Edge Nodes expose this API to handle requests for machine-learning model files. When a peer requests the latest model, the responding FL Edge Node delivers the requisite machine-learning model file via an HTTP response, ensuring the efficient sharing of critical model data.

The example function usage has been listed in Appendix 13.9. The EdgeFL framework simplifies the integration of FL capabilities into AI applications, requiring only four lines of code (LOC) for implementation. This ease of use allows software engineers to quickly deploy FL functionality without substantial code modifications or complex re-engineering. The framework provides a streamlined process, beginning with the creation of a Peer instance, which manages configuration, registration, and communication with other FL nodes. The peer instance facilitates key functions such as model aggregation through $peer.aggregation\_func()$ and ensures orderly participation and withdrawal from the FL system via $peer.start()$ and $peer.unregister\_peer()$, which allows seamless participation in the EdgeFL ecosystem.

13.4.3 EdgeFL Learning Life-Cycle

The life cycle of the EdgeFL framework contains several essential steps for an individual edge node to seamlessly become part of the collaborative process, including joining, model training, knowledge sharing, model aggregation, and optionally departing from the FL process. Algorithm 11 outlines the detailed FL learning process of an individual edge node. Here's a detailed description of each step:

1. Edge Node Joining: The process begins with the edge node initializing itself by creating an instance of the Peer class and a background instance for handling model requests. The node then connects to the registration nodes, essentially announcing its presence and status as an active participant. In return, it receives information about other peers and their participation in the FL process.

2. Model Training: With its registration complete, the edge node dives into the FL training process. It begins model training using its locally stored dataset. Through iterative updates, the node refines its local model to enhance its performance.

3. Sharing Models: The FL client node takes an active role in knowledge sharing, retrieving models from fellow peers within the FL framework. It identifies these peers through the registration nodes, fetching the most recent models. These models are then seamlessly integrated into their own local model updates. Notably, this model retrieval process operates asynchronously, ensuring that ongoing local model training remains uninterrupted. This asynchronous model aggregation mechanism allows the FL client node to simultaneously contribute to and benefit from the collaborative learning process.

4. Model Aggregation: The FL client node performs an aggregation function, which combines locally updated models with models provided by other peers. The aggregation function combines several models to create a new aggregated model that incorporates the collective knowledge of all participating nodes. It is crucial to note that the EdgeFL framework's aggregation function can be tailored to individual analysis and case requirements, giving software engineers the freedom to define and

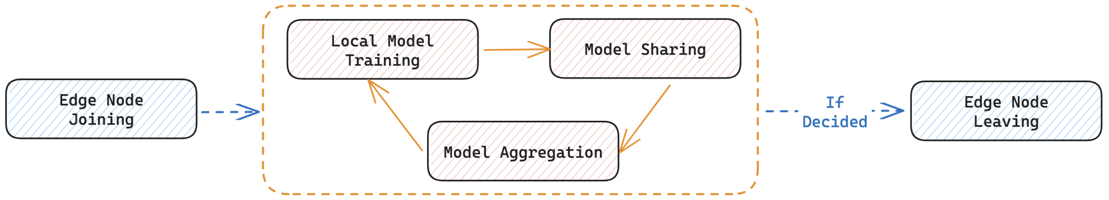


Figure 13.2: The Learning Life-Cycle of EdgeFL, including joining the FL process, model training, sharing, aggregation, and eventual node leaving. These stages collectively define the operational flow of EdgeFL within an edge node.

implement various aggregation algorithms that meet their specific requirements. For general performance analysis, this study employs a default averaging function.

5. Edge Node Leaving: In the event that an edge node decides to exit the EdgeFL system, the FL client node notifies the registration node of its intent to leave, providing its hostname for identification. The registration node promptly updates the active participant list, removing the departing edge node. It's worth noting that some edge nodes may choose to stay in the system even after completing their learning process. By choosing to stay, they contribute by providing their completed learning models, which proves beneficial for newcomers entering the EdgeFL framework. This approach ensures that the system retains the availability of fully learned models, simplifying the onboarding process for new participants.

This life cycle is repeated as additional edge nodes join the FL framework, contribute to the training and aggregation processes, exchange their models, and eventually leave when they decide to discontinue their involvement. While protecting the privacy and autonomy of individual edge nodes, the EdgeFL enables continual collaborative learning and model development.

13.4.4 Containerization and Scalable Deployment

To facilitate effortless deployment on a wide range of edge devices, the EdgeFL framework underwent containerization, employing the robust containerization technology known as Docker. This process involved the encapsulation of all essential components, dependencies, and configurations of EdgeFL into a lightweight and portable container image. This containerization approach delivers the flexibility for EdgeFL to be deployed seamlessly across diverse edge devices, regardless of the specific underlying operating system or hardware architecture.

The architectural diagram showcased in Figure 13.3 provides a representation of the streamlined and scalable deployment architecture of the EdgeFL framework. Within this architectural framework, each edge node container includes a range of services, such as model training, model aggregation, and model serving. Concurrently, the registration node container is equipped

Algorithm 11: EdgeFL - In the system, α represents the ratio of aggregated peers; C is the active peer list; B is the local mini-batch size; E represents the number of local epochs, and γ is the learning rate.

```
Initialize $w_0$
Initialize $\alpha$ as the ratio of aggregated peers
Initialize $C$ as the active peer list
Function Server_Function():
  for each round t = 1, 2, ... do
      Node_Training(w_t)
      Retrieval active peer list $C$
      $m \longleftarrow max(len(C) \times \alpha, 1)$;
      $N_t \longleftarrow$(random set of $m$ peers from $C$);
      for each node $k \in N_t$ do
         Threads.start()
         Fetch $w^k_{t+1}$
         Threads.end()
      end for
      $w_{t+1} \longleftarrow \sum_{k=1}^{K} \frac{1}{K} w^k_{t+1}$;
  end for

  Function Node_Training($w$):
     $\beta \longleftarrow$(split $P_k$ into batches of size $B$);
     for each local epoch $i$ from 1 to E do
         for batch $b \in \beta$ do
            $w \longleftarrow w - \gamma \nabla l(w; b)$;
         end for
     end for
  return $w$ for model sharing
```

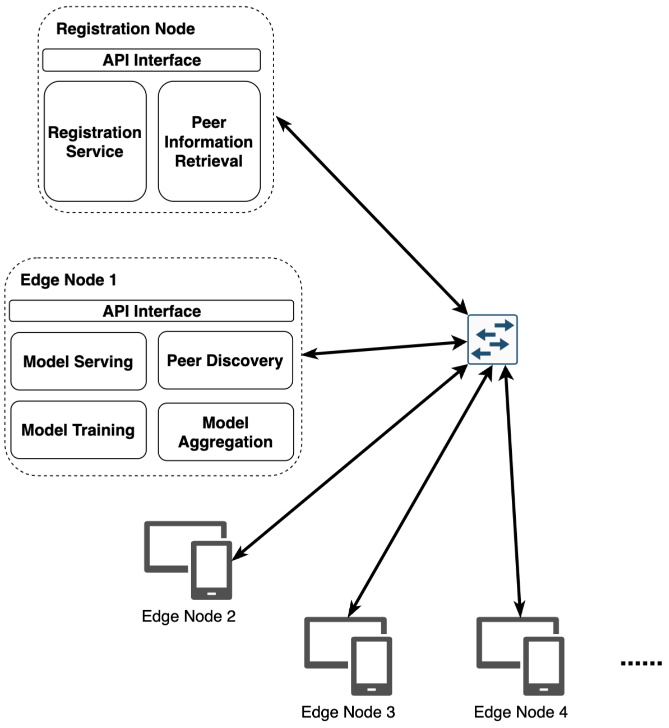


Figure 13.3: Containerized architecture for seamless and scalable deployment of the EdgeFL framework

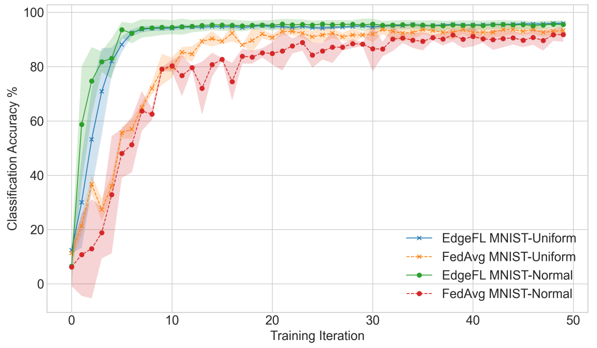
with services for handling registration and peer discovery tasks. This interconnected setup ensures smooth communication among all nodes within the EdgeFL framework. An important feature to highlight is the ability to expand the number of registration nodes in alignment with the growth of participating edge nodes. This scalability enables efficient coordination and management within the EdgeFL framework, ensuring smooth coordination.

By containerizing EdgeFL, software engineers benefit from the ease of distributing and deploying the framework on edge devices, alleviating concerns about intricate installation procedures or compatibility issues. The containerized EdgeFL image includes all the necessary software libraries, frameworks, and configurations, creating a self-contained environment for running the FL client nodes. Furthermore, containerization ensures that the EdgeFL framework remains isolated and independent, effectively preventing conflicts with other software components residing on the edge device. This approach not only simplifies deployment but also improves the stability and reliability of the EdgeFL framework in diverse edge computing environments.

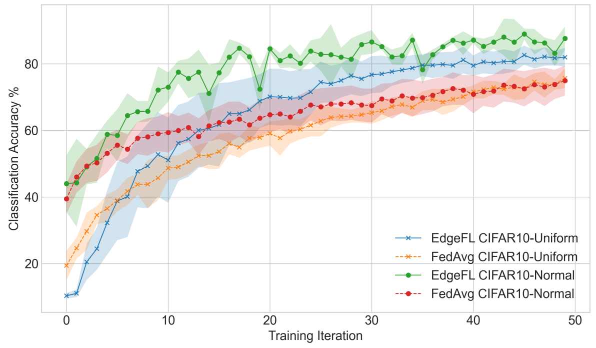
13.4.5 Comparison to Existing FL Framework

EdgeFL employs a decentralized architecture where FL Edge Nodes collaborate directly, eliminating reliance on a central server and significantly enhancing fault tolerance. This decentralized approach reduces the risk of single points of failure and provides a robust, flexible framework suited to dynamic edge environments. In contrast, frameworks like TensorFlow Federated (TFF) and FATE rely on a central server for managing model aggregation and coordination, which can become a bottleneck and a potential single point of failure. PySyft supports decentralized learning but often still involves some central coordination, while LEAF's reference implementations may also be central-server dependent. PaddleFL, while supporting decentralized setups, typically defaults to a central coordination model, which can limit fault tolerance compared to EdgeFL's fully decentralized approach.

In addition, EdgeFL is designed with scalability in mind, allowing edge nodes to operate independently and interact directly with one another. This architecture supports a scalable ecosystem that can grow dynamically with minimal overhead. Registration Nodes facilitate peer discovery and management in a decentralized manner, adapting well to varying network conditions and device capabilities. On the other hand, frameworks like TFF and FATE



(a) MNIST



(b) CIFAR-10

Figure 13.4: The comparison of classification accuracy by utilizing FedAvg (commonly used by existing FL frameworks) and EdgeFL framework

face scalability challenges due to their central server-based models, which can become bottlenecks and affect performance in large-scale, heterogeneous networks. PySyft's scalability may be constrained by its reliance on central coordination, and LEAF's benchmark-oriented design does not inherently address scalability in real-world deployments. PaddleFL's central coordination model can similarly limit its scalability and flexibility compared to the decentralized design of EdgeFL.

EdgeFL is tailored to operate effectively in heterogeneous edge environments, accommodating devices with varying computational power, storage, and connectivity. Its support for asynchronous communication and model aggregation ensures efficient performance across diverse conditions. The containerization of EdgeFL further enhances its adaptability, enabling seamless deployment across different hardware and software platforms. Conversely, traditional frameworks like TFF and FedProx may struggle in heterogeneous settings due to their reliance on synchronous updates and assumptions of homogeneous device capabilities. PySyft's adaptability is enhanced by privacy-preserving techniques but may still face challenges in highly diverse environments. FATE's robustness in privacy comes with the cost of less flexibility in heterogeneous settings, while LEAF's benchmark focus might not address the nuances of real-world adaptability. PaddleFL's adaptability is also affected by its default central coordination model.

Last but not least, EdgeFL emphasizes ease of use with its streamlined implementation and user-friendly APIs, allowing for rapid integration into existing AI solutions with minimal re-engineering. This simplicity is particularly advantageous in fast-paced development environments. In contrast, other FL frameworks often require more extensive setup and configuration. For instance, TensorFlow Federated (TFF) and FATE can involve complex configurations and integration efforts. PySyft, while offering privacy features, may necessitate additional setup for privacy and security configurations. LEAF, being a benchmarking tool, does not prioritize ease of integration, and PaddleFL, despite its robust features, can involve a more involved setup process compared to the straightforward integration offered by EdgeFL.

13.5 Evaluation Results

This section presents the experimental results obtained from the EdgeFL framework, with a specific focus on three aspects outlined in Section 13.3.5. 1)Weights Update Latency: This metric quantifies the time required to transmit model weights from one node to another. 2) Model Evolution Time: This aspect measures the duration it takes to acquire a new version of the model. 3) Classification Accuracy: The evaluation of model performance on the edge dataset is a key element of this analysis. To ensure that our experiments provide robust and meaningful results, we conducted simulations with a total of 10 nodes. In these simulations, all nodes actively participated in the training procedure for both MNIST and CIFAR10 applications. This setup was designed to ensure an adequate number of samples on each edge node, facilitating a comprehensive analysis and evaluation of the EdgeFL framework and offering insights into the real-world performance and effectiveness of EdgeFL in practical scenarios.

Firstly, we examine two performance metrics: weights update latency and model evolution time. Our experimental findings, as presented in Table 13.3, showcase the comparisons among the EdgeFL framework and existing FL frameworks across both MNIST and CIFAR10 applications.

In the domain of weights update latency, EdgeFL consistently demonstrates its efficiency. It exhibits reduced delays in transmitting model weights between edge nodes, underscoring the efficacy of its decentralized architecture. These improvements in weights update latency can be attributed to the streamlined communication processes inherent to EdgeFL. The decentralized nature of the framework allows for rapid sharing of updated models among nodes.

EdgeFL also excels at achieving rapid model evolution in scenarios with unequally distributed datasets. EdgeFL leverages its decentralized architecture and a pull-based model-sharing mechanism to facilitate quick model evolution. This mechanism empowers edge nodes to promptly enhance their local models, effectively adapting to the available data. Consequently, EdgeFL reduces the time required for model evolution, especially when dealing with imbalanced dataset distributions.

These findings demonstrate the advantages of EdgeFL over traditional FL methods. Its better performance in terms of model delay and evolution time can be attributed to its efficient communication and aggregation processes. By utilizing the potential of edge computing, EdgeFL minimizes network overhead

Table 13.3: Comparison of weight update latency and model evolution time by leveraging existing FL frameworks and our proposed EdgeFL framework

FL Frameworks	MNIST		CIFAR10	
	Weights Update Latency (sec)	Model Evolution Time (sec)	Weights Update Latency (sec)	Model Evolution Time (sec)
TFF⋆	-	7.76	-	16.372
PySyft	0.0247	9.05	0.0311	18.292
FATE	0.0326	13.868	0.0473	31.689
LEAF⋆	-	10.239	-	27.362
PaddelFL	0.0258	11.667	0.0412	25.581
EdgeFL	**0.0092**	**5.093**	**0.0148**	**10.753**

⋆ TFF and LEAF frameworks do not include actual server and client implementations but rather provide simulations of the FL process. Therefore, measuring weight latency is not feasible within these frameworks.

and streamlines the model update process, leading to quicker and more responsive knowledge sharing across the edge network. The observed enhancements in model delay and evolution time carry substantial implications for real-world applications. Reduced delays enable the quick exchange of updated models, ensuring timely access to the latest knowledge throughout the edge network. Furthermore, the reduced evolution time enables edge devices to promptly adapt to evolving data characteristics. These characteristics make EdgeFL well-suited for use cases that demand rapid model evolution and responsiveness to dynamic environmental changes.

In addition to evaluating weights update delay and model evolution time, we conducted extensive accuracy comparisons between EdgeFL's decentralized averaging and the widely used FedAvg algorithm [145] found in existing FL frameworks. Figure 13.4 illustrates the accuracy comparisons. The results demonstrate that EdgeFL's decentralized averaging approach outperforms the centralized FedAvg method when it comes to evaluating models on edge devices. The average accuracy achieved through EdgeFL's decentralized averaging is approximately 2% higher for MNIST and 5% higher for CIFAR-10 datasets. These findings underscore the accuracy improvements made by EdgeFL's decentralized averaging approach. The observed increase in accuracy demonstrates the effectiveness of EdgeFL's decentralized averaging mechanism in enhancing model performance. By utilizing the collective knowledge and insights gained from distributed edge devices, EdgeFL facilitates improved model convergence.

Figure 13.5 illustrates the model distribution latency of the decentralized EdgeFL algorithm in comparison to other algorithms as the number of edge nodes increases from 10 to 100. From the figure, PySyft and PaddleFL exhibit slightly higher latency, particularly with more nodes. FATE records the highest latency, especially in larger networks, which results in greater communication overhead. EdgeFL's latency shows a gradual rise with node expansion, indicating manageable overhead for larger networks due to decentralized characteristics. This scalability is achieved because each node can establish equal connections, distributing server workload more evenly to the edge, and effectively balancing updating traffic. Consequently, EdgeFL demonstrates the most modest growth rate among the compared algorithms, showcasing its scalable performance, particularly in large-scale machine learning applications and edge computing scenarios.

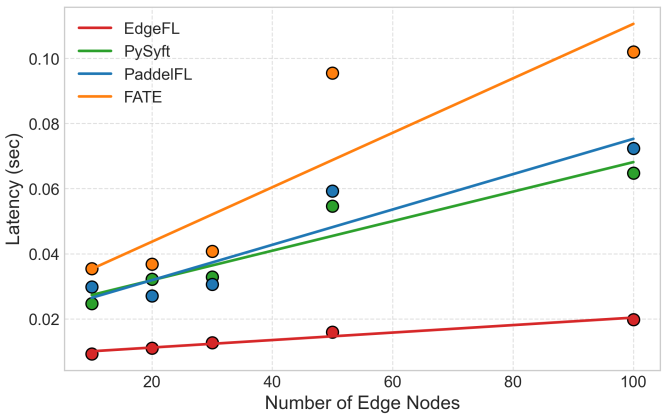


Figure 13.5: Model distribution latency with the increasing number of edge nodes

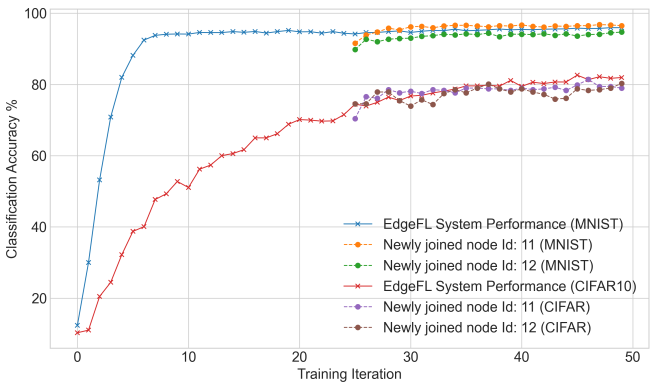


Figure 13.6: Midway joined node classification performance in both MNIST and CIFAR-10 application

Moreover, our study demonstrates the effectiveness of EdgeFL's asynchronous join feature, which enables new nodes to seamlessly integrate into the existing system and swiftly acquire the latest knowledge without necessitating a full retraining process from scratch. As depicted in Figure 13.6, our observations reveal that when new nodes (node id: 11, 12) join the system midway through the training process, they rapidly achieve the same level of accuracy as the overall system. This outcome illustrates EdgeFL's capability to facilitate efficient knowledge transfer and rapid model convergence for newly joined nodes.

The asynchronous join functionality within EdgeFL offers advantages in terms of scalability and time-to-adaptability. By allowing new nodes to immediately benefit from the collective intelligence of the system without the need for extensive training, EdgeFL reduces the computational burden and time required for onboarding new participants. This attribute proves its value in dynamic environments where nodes frequently join and depart from the system.

This demonstration of the asynchronous join capability within EdgeFL underlines its potential for real-world deployments, especially in scenarios where rapid knowledge transfer and swift integration of new nodes are crucial. By leveraging the existing knowledge base and facilitating the seamless assimilation of new nodes, EdgeFL empowers FL systems to adapt and evolve efficiently over time. These findings underscore the advantages of EdgeFL's asynchronous join mechanism, highlighting its potential to enhance the scalability and flexibility of FL within dynamic edge computing environments.

13.6 Case Study

In addition to the comparisons, we conduct a case study to apply the EdgeFL framework to a practical machine learning application. In this case study, we leverage the EdgeFL framework to tackle the task of sentiment analysis on the IMDB Dataset of 50K Movie Reviews, a well-known benchmark for binary sentiment classification [275]. This dataset comprises 50,000 movie reviews for training and testing, providing a substantial amount of data for our analysis. Our objective is to perform sentiment classification, distinguishing between positive and negative sentiments within the movie reviews. To address this task, a Long Short-Term Memory (LSTM) network is utilized, which is known for its ability to capture sequential dependencies and nuances

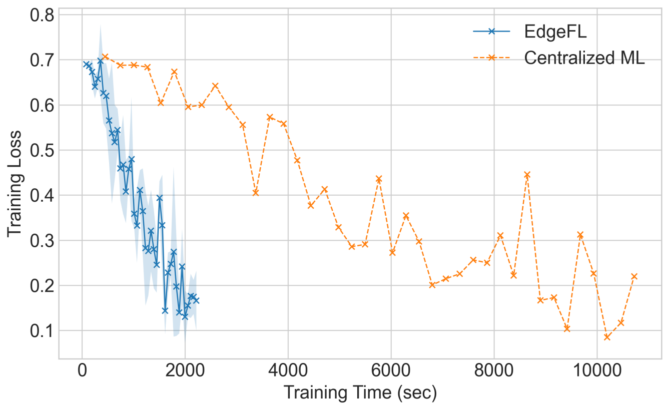


Figure 13.7: Training Loss with training time consumed by EdgeFL and Centralized ML

within textual data [276][277]. LSTM networks are particularly well-suited for natural language processing tasks like sentiment analysis.

We simulate ten distinct movie review sites, each represented by an individual user end node. This decentralized setting not only allows for parallel processing and training on diverse data sources locally. The decentralized architecture of EdgeFL ensures that each server node trains an LSTM model locally, making updates based on the reviews available at its respective sites. These local models are then collaboratively aggregated with the function "$peer.aggregation\_func()$" to create a global model that encapsulates insights from all nodes. This FL process enables the collective model to benefit from the diversity of reviews across different movie sites, resulting in a more robust sentiment analysis model.

Figure 13.7 and 13.8 illustrated the result of the sentiment analysis application with the EdgeFL framework compared to traditional centralized machine learning. For the result of training loss, the figure illustrates a notable efficiency gain achieved by the EdgeFL framework in comparison to traditional centralized machine learning. Specifically, EdgeFL demonstrates approximately five times greater efficiency in terms of training time. This efficiency is evident in the faster convergence of the training loss curve, indicating that the model improves significantly in less time.

Figure 13.8 depicts the comparative performance of sentiment analysis train-

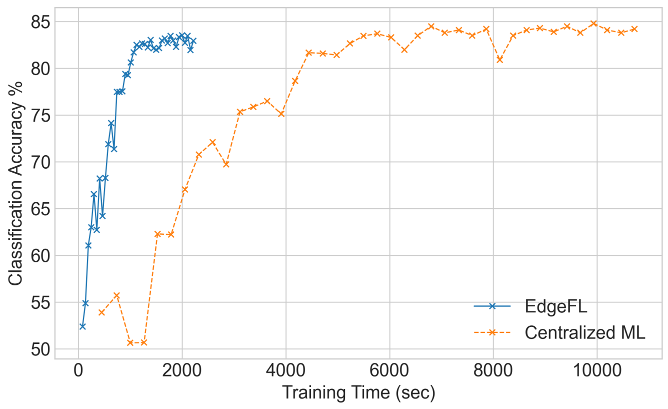


Figure 13.8: Test Accuracy change with training time consumed by EdgeFL and Centralized ML

ing using two approaches: EdgeFL and traditional centralized machine learning. The test accuracy of EdgeFL is the average value among all participated end nodes. The results illustrate that EdgeFL achieves faster training times, approximately five times more efficient when performing rapid model training and inference, indicating its superiority in scenarios involving decentralized data sources or resource-constrained environments.

Through decentralized training across ten server nodes, we achieve robust sentiment classification while respecting data privacy and security, making this approach highly suitable for large-scale, distributed natural language processing tasks, especially in scenarios involving user-generated content and edge computing environments. In this case, we build a practical prototype of a decentralized FL system with the EdgeFL framework that can be seamlessly deployed to container orchestration platforms such as Docker Swarm, Kubernetes, etc.

13.7 Discussion

This paper focuses on analyzing and interpreting the results obtained from the experiments conducted with the EdgeFL framework. We evaluated EdgeFL using a range of metrics. Firstly, the EdgeFL outperformed existing FL frameworks in terms of weights update delay. EdgeFL managed to reduce weights update delay by roughly 50%, surpassing centralized alternatives. When dealing with unevenly distributed datasets, EdgeFL demonstrated its benefit as well. EdgeFL's decentralized architecture addresses key limitations of traditional FL frameworks, such as centralized aggregation and scalability challenges. Unlike FedAvg and similar approaches that rely on a central server, EdgeFL employs decentralized model training and aggregation, enhancing fault tolerance and reducing latency. This approach aligns with the broader shift towards decentralized systems in distributed computing, offering a new perspective on how FL can be implemented in edge environments. The implications of this work include a rethinking of model aggregation strategies in FL, moving away from server-dependent methods to a more resilient, peer-to-peer system. EdgeFL's pull-based model-sharing mechanism introduces a dynamic way for nodes to access the latest updates, contributing to faster convergence and adaptation in non-IID environments, which could inspire further research into adaptive and asynchronous FL methods that better accommodate the heterogeneity of real-world data and network conditions.

Compared to previous FL frameworks, EdgeFL's contributions lie in its ability to achieve faster weight updates and model convergence without sacrificing accuracy. The framework's decentralized averaging technique consistently outperforms FedAvg, particularly in scenarios with unevenly distributed datasets, demonstrating that EdgeFL can achieve more efficient and accurate learning in diverse edge environments. Furthermore, EdgeFL enables faster model training and evolution, making it a solution for applications where data is generated and processed locally, such as IoT devices, smart cities, and autonomous systems. The ability of EdgeFL to seamlessly integrate new nodes without requiring complete retraining further underscores its scalability and adaptability, which are crucial for real-world deployment. Additionally, the case study on sentiment analysis with the IMDB dataset demonstrates how EdgeFL can be practically applied to enhance the performance and efficiency of machine learning tasks, particularly in scenarios with decentralized data sources.

However, throughout our studies, it's important to note that, despite the multiple benefits provided by EdgeFL, we observed two major limits that deserve consideration.

Firstly, we observed a notable increase in bandwidth cost as the number of nodes in our decentralized network grew. This escalation in connectivity demands is a natural consequence of decentralization, and while it can be partially managed through parameters such as the connection parameter α (which determines the number of models shared in each round), it's important to note that adjusting this parameter may have an effect on the final quality of the model. Addressing this trade-off is crucial because excessive bandwidth consumption can limit the scalability of EdgeFL and increase operational costs, particularly in environments with constrained network resources. Research focused on optimizing communication protocols, such as developing efficient data compression techniques, adaptive bandwidth management strategies, and novel aggregation methods, is essential. These advancements would enable EdgeFL to maintain high model accuracy while minimizing bandwidth demands, making it more practical and cost-effective for large-scale, real-world deployments.

Second, we encountered computational constraints while training large neural networks, particularly when dealing with resource-constrained edge devices. The inherent limitations of these edge devices, such as low processing power and memory, caused difficulties in carrying out complicated training operations. These constraints can severely impact the performance and feasibility of FL processes on such devices. The need to perform intensive computations and manage large models can lead to inefficiencies and slowdowns, which in turn affect the overall effectiveness of the FL system. These highlight the importance of ongoing research efforts focused on reducing computation complexity and developing novel ways for accommodating resource-constrained contexts. By addressing these computational constraints, EdgeFL can become more versatile and capable of operating effectively on a wider range of edge devices, thus enhancing its practical value and usability in diverse edge scenarios.

13.8 Conclusion

This paper introduces EdgeFL, a novel decentralized FL framework designed exclusively for efficiency and low-effort implementation. EdgeFL effectively addresses scalability, integration, and efficiency challenges by utilizing an edge-only model training and aggregation approach, eliminating the need for a central server. The framework ensures seamless scalability across a diverse range of use cases. The framework offers a straightforward integration process, requiring only four lines of code (LOC) for software engineers to incorporate FL functionalities into their AI products. Furthermore, EdgeFL offers engineers the flexibility to customize aggregation functions to meet their specific needs, enhancing the framework's adaptability and versatility. Our experiments and evaluations demonstrate the strengths and advantages of EdgeFL. In numerous aspects, EdgeFL outperforms existing FL frameworks. It excels in reducing weight update latency and model evolution time by 50%, enhancing classification accuracy by 2% for the MNIST dataset and 5% for the CIFAR dataset compared to other FL frameworks. In a real-world case study, EdgeFL demonstrated remarkable efficiency gains, achieving training times approximately five times faster than traditional centralized machine learning, all while maintaining a comparable level of model quality. These findings highlight EdgeFL's potential in practical applications, particularly in industrial settings where timely and accurate model updates are crucial.

Our future work aims to validate and extend the capabilities of EdgeFL with a broader range of use cases. We also plan to explore resource optimization techniques, such as model compression and quantization, to enhance communication efficiency for edge devices in EdgeFL. Additionally, we will investigate adaptive aggregation strategies that dynamically adjust the aggregation process based on network conditions, device capabilities, and data heterogeneity.

13.9 EdgeFL Function Details and Example Usage

The proposed EdgeFL framework provides software engineers with an easy way to integrate FL capabilities into their AI solutions. In contrast to the complexity commonly associated with FL frameworks, EdgeFL offers a concise implementation that requires only four lines of code (LOC). This ease of use enables software engineers to quickly include FL functionality into their existing AI applications, with no need for substantial code changes or complex re-engineering efforts. The following Listing 1 demonstrates an example with which EdgeFL can be deployed.

Listing 13.1: Usage example of EdgeFL.

```
# --- Continue from node training part ---
# --- Initialize peer instance ---
peer = Peer(config)

# --- Start peer instance ---
peer.start()

for epoch in range(number_of_epochs):

    # --- Pull model from active peers and start
    # aggregation
    w_latest = peer.aggregation_func()

    model.load_state_dict(w_latest)

    train(model, torch.device("cpu"), train_loader,
            optimizer, epoch)
    test(model, torch.device("cpu"), test_loader)
    scheduler.step()

    torch.save(model.state_dict(), "model-latest.pth")

# --- unregister from the registration node if leave
peer.unregister_peer()
```

peer = Peer(configs): The procedure begins with the creation of an instance of the Peer class, a component of the EdgeFL architecture that represents an active participant in the system. During this initialization, an array of configurations is created, which includes elements such as registration node addresses and specific settings for the aggregation function. This phase ensures that the FL edge node is precisely configured, allowing for flawless connection with registration nodes and competent involvement in FL training and aggregation efforts. Once the Peer class instance is established, the Peer class instance serves as a handle, supporting interactions between the FL edge node and its peer entities. The FL edge node gains the ability to do a variety of actions via this peer object, including model retrieval, registration with registration nodes, and model aggregation.

peer.start(): This function initiates the execution for the FL edge node's active participation in the EdgeFL framework. When called, it triggers a series of mandatory steps that prepare the FL edge node for seamless participation in the FL process. These steps include the registration of the FL edge node with the registration nodes, the establishment of connections with its peer entities, and the activation of a background instance ready to handle asynchronous file requests from peers. By calling "peer.start()", the FL edge node actively participates in the collaborative model learning process, while leveraging the computational capabilities in edge devices. By using this function, the FL edge node integrates easily into the EdgeFL ecosystem, establishing its role as a contributor to the system of FL.

peer.aggregation_func(): This function is in charge of orchestrating the aggregation process, which is a cornerstone of the EdgeFL system. When triggered, it starts a multidimensional series of operations aimed at increasing the collective intelligence of the FL system. First, the function requests models from fellow FL edge nodes specified by the registration nodes. Following that, it employs the aggregation method, expertly combining these models into a single updated model. The aggregation function facilitates the collaborative nature of FL. It combines the inputs of various peer entities, harmonizing their knowledge to improve the quality and performance of the model. The FL edge node takes an active role in this iterative model aggregation process by calling "peer.aggregation_func()", effectively becoming a contributor within the EdgeFL framework. This not only encourages the FL system's collective intelligence but also improves the overall quality of the model.

peer.unregister_peer(): This function supports the FL edge node's seamless exit from the EdgeFL framework. When invoked, it performs the role of notifying the registration nodes of the approaching unregistration while providing the necessary information, including the FL edge node's hostname. The function, "peer.unregister_peer()," effectively starts the process of removing the FL client node from the list of active participants maintained by the registration nodes. This action ensures the proper management of participants within the EdgeFL framework and allows for efficient resource allocation and coordination among the remaining active peers, adding to the EdgeFL framework's overall resilience and stability.

CHAPTER 14

Concluding Remarks and Future Work

In this chapter, we will summarize the main findings from the research presented in this thesis, focusing on the key questions we aimed to answer. Our exploration of Federated Learning and Reinforcement Learning in embedded systems has revealed important ways to improve edge intelligence. By looking at the nine papers included in this work, we will answer the research questions and highlight the main contributions of this thesis. These contributions not only deepen our understanding of these techniques but also provide practical approaches for their use in real-world situations.

14.1 Discussion

The research presented in this thesis builds on the foundation of Federated Learning and Reinforcement Learning and explores their application in the context of embedded systems and edge computing. By analyzing various case studies and frameworks, this research provides novel insights into how these learning techniques can be adapted to address the unique challenges of decentralized systems operating in dynamic environments.

Compared to previous work, such as the studies on FL in embedded devices and on RL in real-world environments, this thesis extends the discussion by examining the synergies between the two methods. While previous research [116][117][115][171] primarily focused on enhancing data privacy and communication efficiency through FL, our research delves deeper into how asynchronous FL models can improve the timeliness of updates, especially in scenarios where real-time decision-making is crucial. The asynchronous protocols proposed in Papers D and E show significant advancements in reducing communication overhead while maintaining high model accuracy, offering a more dynamic approach to the traditional FL frameworks described in earlier studies.

Moreover, our exploration of RL complements previous findings by highlighting its value in continuously adapting to real-world challenges. While existing research primarily [203][201][204] demonstrates RL's ability to optimize decision-making in complex environments, this thesis adds to the field by integrating RL with FL, showing how the combination of these techniques can further enhance the performance of embedded systems. For example, in Papers F and G, RL algorithms are applied to autonomous systems, like UAVs, to improve operational efficiency, a departure from more static applications seen in prior studies.

The contributions of this research not only addresses the technical challenges of deploying FL and RL in embedded systems but also offers practical solutions, such as the EdgeFL framework, which simplifies the adoption of decentralized learning in edge environments. In this sense, the research adds to the existing body of knowledge by providing actionable methodologies that developers and researchers can use to improve the scalability, efficiency, and adaptability of embedded systems.

In reflecting on the results, it becomes clear that while the combination of FL and RL holds great promise, there are still areas requiring further exploration. For instance, how these methods can be applied to other domains beyond the current use cases discussed in this thesis remains an open question. Additionally, future research should focus on addressing the limitations identified in edge devices with constrained computational resources, ensuring that the advanced algorithms proposed can be universally deployed without significant trade-offs in performance or latency.

Therefore, the research questions framed in Chapter 3 are addressed as follows:

RQ1. How can Federated Learning be effectively engineered and deployed in embedded systems to enhance data privacy, reduce communication costs, and ensure model accuracy?

Federated Learning can be effectively engineered and deployed in embedded systems by taking advantage of its decentralized structure. This approach allows model training to happen directly on devices, such as smartphones or IoT sensors, without the need to send sensitive user data to a central server. This local training significantly enhances data privacy, as sensitive information remains on the user's device, reducing the risk of data breaches and complying with privacy regulations. Papers A and B emphasize these benefits, demonstrating how Federated Learning not only protects user data but also addresses the growing concerns around data privacy in various applications.

Moreover, Federated Learning helps to reduce communication costs by minimizing data transmission. Instead of sending large datasets over the network, only the updates to the model—derived from local data—are communicated back to a central server. This approach is particularly beneficial in environments with limited bandwidth, where transferring large amounts of data can be slow and expensive.

To further improve model accuracy, asynchronous model aggregation methods can be employed, as discussed in Papers D and E. These methods allow devices to update the central model more flexibly and efficiently. Instead of waiting for all devices to complete their training before aggregating results, updates can be sent as soon as they are ready. This leads to more timely adjustments based on local data and helps the model adapt better to real-time changes in the environment. By combining local insights with global learning, Federated Learning enhances the overall accuracy and robustness of models deployed in dynamic embedded systems.

RQ2. What architectural frameworks for Federated Learning can optimize scalability and performance in edge computing environments, and how do these architectures influence model training efficiency?

The research identifies several architectural frameworks for Federated Learning that are designed to optimize scalability and performance in edge computing environments. Paper C provides a detailed comparison of four main architectures: centralized, hierarchical, regional, and decentralized. It highlights that decentralized frameworks are particularly advantageous, as they reduce the risks associated with single points of failure often found in centralized systems. By distributing the workload across multiple devices, decentralized architectures enhance scalability, allowing for the integration of a larger number of edge devices without compromising performance.

Additionally, Paper C examines the trade-offs between communication latency and model performance. It underscores the importance of selecting the appropriate architecture to maintain high training efficiency, particularly in environments where network conditions may vary. The findings suggest that decentralized architectures not only improve system performance but also effectively manage the limitations inherent in edge devices, such as limited processing power and intermittent connectivity.

Furthermore, Paper I introduces the EdgeFL framework, which aims to address some of the challenges faced by existing Federated Learning implementations. By providing a more efficient and low-effort approach to decentralized Federated Learning, EdgeFL enhances scalability and reduces the complexity often associated with traditional frameworks. This framework supports faster model updates and reduces latency, thus improving overall model training efficiency. Together, the insights from these papers illustrate that choosing the right architectural framework is essential for optimizing the deployment of Federated Learning in edge computing environments, ensuring that the systems remain adaptable, efficient, and responsive to real-world demands.

RQ3. How can Federated Learning and Reinforcement Learning methods enhance the adaptability and efficiency of embedded systems in dynamic environments?

Federated Learning and Reinforcement Learning methods can significantly enhance the adaptability and efficiency of embedded systems by enabling them to learn from their environments in real-time. This capability is especially crucial in dynamic scenarios where conditions can change rapidly, such as during emergencies or in environments with variable network connectivity. Papers D, E, F, G, and H illustrate how these techniques can be effectively applied in such contexts.

Papers D and E highlight the importance of asynchronous Federated Learning methods, which further enhance adaptability by enabling timely updates based on local data. In Paper D, the authors propose an asynchronous model aggregation protocol that allows edge devices to send updates to the central model as soon as they are ready. This flexibility is crucial for maintaining high accuracy in rapidly changing environments, as it reduces the latency between data collection and model improvement. Paper E extends this concept by introducing an asynchronous federated aggregation protocol for deep neural decision forests, showcasing how these approaches can accelerate model training while ensuring high classification performance.

In Paper F, the authors discuss the deployment of an unmanned aerial vehicle (UAV) base station in disaster-stricken areas. Here, deep Reinforcement Learning algorithms are used to autonomously navigate the UAV and optimize its operations based on real-time feedback from the environment. This ability to adapt ensures that the UAV can maintain connectivity for mission-critical users, demonstrating how Reinforcement Learning can improve responsiveness and service quality under challenging conditions.

Similarly, Paper G, H present dynamic Reinforcement Learning algorithms tailored for the telecommunications industry. These research show that the algorithm can adjust the position and antenna tilt of a drone-based base station to adapt to fluctuating user demands and environmental changes. The results indicate a substantial improvement in service performance, underscoring how deep Reinforcement Learning allows embedded systems to evolve in response to complex, real-world challenges.

The integration of Federated Learning and Reinforcement Learning techniques equips embedded systems with the capability to respond effectively to real-world challenges, ultimately leading to improved adaptability and operational efficiency. By leveraging these advanced learning methods, embedded systems can continuously learn and adjust their behaviors, ensuring optimal performance even in dynamic and unpredictable environments.

14.2 Key Contributions

This thesis makes a significant contribution to the field of edge intelligence by providing a comprehensive exploration of Federated Learning and Reinforcement Learning. It highlights the potential of these advanced learning techniques to enhance the functionality and adaptability of embedded systems. By examining various applications and scenarios, the research deepens the understanding of how these methods can be effectively integrated into real-world systems.

A key aspect of this work is the identification and comparison of several architectural frameworks for Federated Learning. The analysis, particularly presented in Paper C, demonstrates the advantages of decentralized architectures. These frameworks mitigate the risks associated with single points of failure while improving scalability and performance in edge computing environments. This contribution is crucial for optimizing the deployment of Federated Learning, ensuring that systems can grow and adapt without compromising their effectiveness.

Another important contribution of the thesis is the introduction of innovative asynchronous model aggregation techniques. Papers D and E explore how these methods enhance real-time learning capabilities and reduce communication overhead. By allowing for more efficient updates, these techniques ensure that embedded systems can respond promptly to changing conditions, thus maintaining high accuracy and performance.

The research also includes practical applications of Federated Learning and Reinforcement Learning in dynamic environments, as demonstrated in Papers F, G, and H. These case studies illustrate how embedded systems can autonomously adapt to real-world challenges, such as disaster response and variable network conditions, ultimately improving service quality and operational efficiency. The development of the EdgeFL framework represents a significant contribution to the field. This framework provides a low-effort, efficient approach to decentralized Federated Learning, addressing key challenges related to implementation and scalability. It serves as a valuable resource for developers and researchers aiming to adopt Federated Learning in practical applications.

Finally, the thesis identifies several open research questions related to the deployment of Federated Learning in embedded systems. By outlining these questions, it sets a foundation for future studies and encourages further exploration in this rapidly evolving area, fostering ongoing innovation and advancement in edge intelligence. In summary, the main objectives of contributions are as follows:

- Investigated the integration of Federated Learning and Reinforcement Learning techniques to enhance the adaptability and efficiency of embedded systems in dynamic environments.
- Analyzed and compared different architectural frameworks for Federated Learning, identifying their strengths and weaknesses in optimizing scalability and performance in edge computing.
- Developed innovative asynchronous aggregation methods that improve real-time learning capabilities while minimizing communication overhead in embedded edge systems.
- Established a practical framework, EdgeFL, that simplifies the implementation of decentralized Federated Learning, addressing challenges related to complexity and scalability, and providing a resource for developers and researchers.

14.3 Future Work

While this thesis has made significant contributions to the fields of Federated Learning and Reinforcement Learning, numerous avenues remain open for further exploration. Future research can build on the findings presented in this work to enhance the adaptability, efficiency, and scalability of embedded systems across various dynamic environments.

One promising area for future investigation is the optimization of Federated Learning algorithms tailored for heterogeneous devices. As edge devices exhibit a wide range of computational power, storage capabilities, and communication bandwidth, developing adaptive learning techniques that can accommodate these differences is crucial. Future studies could focus on refining asynchronous aggregation methods or exploring novel strategies for resource allocation during the training process. By enabling Federated Learning systems to dynamically adjust based on the capabilities of connected devices, researchers can enhance both model performance and energy efficiency.

Another important direction is the integration of Federated Learning with advanced Reinforcement Learning techniques. Exploring hybrid models that leverage the strengths of both learning paradigms could lead to more robust systems capable of effectively responding to real-time changes in their environments. Future research might investigate the development of frameworks that facilitate this integration, enabling embedded systems to make informed decisions based on local data while continually learning from their interactions with dynamic contexts. Empirical studies validating the effectiveness of these hybrid approaches in practical applications will be essential for demonstrating their value.

The EdgeFL framework introduced in this thesis also presents numerous opportunities for further enhancement and application. Future research can dive into its potential in different industries, such as healthcare, finance, and smart cities, assessing its performance and scalability under varied operational conditions. Investigating user-centric approaches to Federated Learning could provide insights into optimizing system design and usability, particularly in sectors that handle sensitive data. Additionally, exploring the role of user feedback in refining Federated Learning models could foster a more adaptive and responsive system.

As privacy concerns continue to escalate in today's data-driven world, there is a pressing need to develop Federated Learning methods that incorporate advanced privacy-preserving techniques. Future research could focus on combining differential privacy and encryption with Federated Learning to ensure that user data remains secure while still enabling effective model training. Such advancements could enhance public trust in these technologies and facilitate wider adoption across industries.

Moreover, exploring the broader societal implications of deploying these technologies will be essential. Future studies could investigate ethical considerations, regulatory frameworks, and the potential impacts on job markets and privacy rights. Engaging with policymakers, industry leaders, and the public can help ensure that advancements in edge intelligence align with societal values and needs, ultimately fostering responsible innovation.

Lastly, conducting longitudinal studies that examine the long-term impacts of Federated Learning and Reinforcement Learning systems in real-world applications could provide valuable insights. These studies could assess how these technologies evolve, adapt, and perform over time, offering critical data for refining existing models and frameworks.

Through these diverse directions, future research can further advance the understanding and application of Federated Learning and Reinforcement Learning in embedded systems. By addressing these challenges and opportunities, researchers can pave the way for innovative solutions that tackle real-world problems and contribute to the development of smarter, more adaptive systems.

References

[1] P. Harrington, *Machine learning in action*. Simon and Schuster, 2012.

[2] S. Wu, "Different machine learning methods for tic-tac-toe prediction," in *2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, IEEE, 2022, pp. 379–383.

[3] D. B. Fogel, "Using evolutionary programing to create neural networks that are capable of playing tic-tac-toe," in *IEEE International Conference on Neural Networks*, IEEE, 1993, pp. 875–880.

[4] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[5] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, "Machine learning for industrial applications: A comprehensive literature review," *Expert Systems with Applications*, vol. 175, p. 114 820, 2021.

[6] M. B. Hoy, "Alexa, siri, cortana, and more: An introduction to voice assistants," *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[7] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: Advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[8] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[9] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, “Machine learning at the network edge: A survey,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[10] X. Liu, J. Yu, J. Wang, and Y. Gao, “Resource allocation with edge computing in iot networks via machine learning,” *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020.

[11] J. Sifakis, “Embedded systems-challenges and work directions,” *Lecture notes in computer science*, vol. 3544, p. 184, 2005.

[12] M. Duranton, “The challenges for high performance embedded systems,” in *9th EUROMICRO Conference on Digital System Design (DSD’06)*, IEEE, 2006, pp. 3–7.

[13] S. Singh, R. Sulthana, T. Shewale, V. Chamola, A. Benslimane, and B. Sikdar, “Machine-learning-assisted security and privacy provisioning for edge computing: A survey,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 236–260, 2021.

[14] K. Malhotra and Y. Kumar, “Challenges to implement machine learning in embedded systems,” in *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, IEEE, 2020, pp. 477–481.

[15] Y. Lu, “Artificial intelligence: A survey on evolution, models, applications and future trends,” *Journal of Management Analytics*, vol. 6, no. 1, pp. 1–29, 2019.

[16] L. Spector, “Evolution of artificial intelligence,” *Artificial Intelligence*, vol. 170, no. 18, pp. 1251–1253, 2006.

[17] S. Angra and S. Ahuja, “Machine learning and its applications: A review,” in *2017 international conference on big data analytics and computational intelligence (ICBDAC)*, IEEE, 2017, pp. 57–60.

[18] D. Dhall, R. Kaur, and M. Juneja, “Machine learning: A review of the algorithms and its applications,” *Proceedings of ICRIC 2019: Recent innovations in computing*, pp. 47–63, 2020.

[19] T. M. Mitchell, “Machine learning and data mining,” *Communications of the ACM*, vol. 42, no. 11, pp. 30–36, 1999.

[20] I. Goodfellow, *Deep learning*, 2016.

[21] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] P. P. Shinde and S. Shah, “A review of machine learning and deep learning applications,” in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, IEEE, 2018, pp. 1–6.

[23] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, “A survey on deep learning in medical image analysis,” *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[24] J. B. Heaton, N. G. Polson, and J. H. Witte, “Deep learning for finance: Deep portfolios,” *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.

[25] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *Journal of field robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[26] R. Rai, M. K. Tiwari, D. Ivanov, and A. Dolgui, *Machine learning in manufacturing and industry 4.0 applications*, 2021.

[27] Z. Ge, Z. Song, S. X. Ding, and B. Huang, “Data mining and analytics in the process industry: The role of machine learning,” *Ieee Access*, vol. 5, pp. 20 590–20 616, 2017.

[28] A. Dogan and D. Birant, “Machine learning and data mining in manufacturing,” *Expert Systems with Applications*, vol. 166, p. 114 060, 2021.

[29] A. Alanazi, “Using machine learning for healthcare challenges and opportunities,” *Informatics in Medicine Unlocked*, vol. 30, p. 100 924, 2022.

[30] M. L. De Prado, *Advances in financial machine learning.* John Wiley & Sons, 2018.

[31] J. Chen and X. Ran, “Deep learning with edge computing: A review,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[32] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[33] F. Oliveira, D. G. Costa, F. Assis, and I. Silva, "Internet of intelligent things: A convergence of embedded systems, edge computing and machine learning," *Internet of Things*, p. 101 153, 2024.

[34] K. Z. Haigh, A. M. Mackay, M. R. Cook, and L. G. Lin, "Machine learning for embedded systems: A case study," *BBN Technologies: Cambridge, MA, USA*, vol. 8571, pp. 1–12, 2015.

[35] M. Nadeski, "Bringing machine learning to embedded systems," *Texas Instruments*, pp. 1–5, 2019.

[36] E. Batzolis, E. Vrochidou, and G. A. Papakostas, "Machine learning in embedded systems: Limitations, solutions and future challenges," in *2023 IEEE 13th annual computing and communication workshop and conference (CCWC)*, IEEE, 2023, pp. 0345–0350.

[37] D. StÎhl and T. MÎrtensson, *Continuous practices: a strategic approach to accelerating the software production system.* Lulu. com, 2018.

[38] T. Mårtensson, D. Ståhl, and J. Bosch, "Continuous integration impediments in large-scale industry projects," in *2017 IEEE International Conference on Software Architecture (ICSA)*, IEEE, 2017, pp. 169–178.

[39] S. Sonko, E. A. Etukudoh, K. I. Ibekwe, V. I. Ilojianya, and C. D. Daudu, "A comprehensive review of embedded systems in autonomous vehicles: Trends, challenges, and future directions," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 2009–2020, 2024.

[40] H. Nam and R. Lysecky, "Latency, power, and security optimization in distributed reconfigurable embedded systems," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2016, pp. 124–131.

[41] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, "An evaluation of edge tpu accelerators for convolutional neural networks," in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, IEEE, 2022, pp. 79–91.

[42] A. Biglari and W. Tang, "A review of embedded machine learning based on hardware, application, and sensing scheme," *Sensors*, vol. 23, no. 4, p. 2131, 2023.

[43] T.-Y. Yen and W. Wolf, "Communication synthesis for distributed embedded systems," in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, IEEE, 1995, pp. 288–294.

[44] M. Liu, Z. Zhuang, Y. Lei, and C. Liao, "A communication-efficient distributed gradient clipping algorithm for training deep neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26 204–26 217, 2022.

[45] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[46] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, vol. 53, pp. 5113–5155, 2020.

[47] A. Albanese, M. Nardello, G. Fiacco, and D. Brunelli, "Tiny machine learning for high accuracy product quality inspection," *IEEE Sensors Journal*, vol. 23, no. 2, pp. 1575–1583, 2022.

[48] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, 2020.

[49] P. H. Feiler, "Challenges in validating safety-critical embedded systems," *SAE International Journal of Aerospace*, vol. 3, no. 2009-01-3284, pp. 109–116, 2009.

[50] P. Skarin, W. Tärneberg, K.-E. Årzen, and M. Kihl, "Towards mission-critical control at the edge and over 5g," in *2018 IEEE international conference on edge computing (EDGE)*, IEEE, 2018, pp. 50–57.

[51] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106 854, 2020.

[52] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[53] S. R. Chalamala, N. K. Kummari, A. K. Singh, A. Saibewar, and K. M. Chalavadi, "Federated learning to comply with data protection regulations," *CSI Transactions on ICT*, vol. 10, no. 1, pp. 47–60, 2022.

[54] P. M. Mammen, “Federated learning: Opportunities and challenges,” *arXiv preprint arXiv:2101.05428*, 2021.

[55] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, “Federated learning review: Fundamentals, enabling technologies, and future applications,” *Information processing & management*, vol. 59, no. 6, p. 103 061, 2022.

[56] P. R. Silva, J. Vinagre, and J. Gama, “Towards federated learning: An overview of methods and applications,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, e1486, 2023.

[57] Q. Li, Z. Wen, Z. Wu, *et al.*, “A survey on federated learning systems: Vision, hype and reality for data privacy and protection,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[58] L. U. Khan, S. R. Pandey, N. H. Tran, *et al.*, “Federated learning for edge networks: Resource optimization and incentive mechanism,” *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[59] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, “Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis,” *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[60] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[61] Y. Li, “Reinforcement learning in practice: Opportunities and challenges,” *arXiv preprint arXiv:2202.11296*, 2022.

[62] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, “Federated learning for internet of things: Recent advances, taxonomy, and open challenges,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[63] P. Peng, W. Lin, W. Wu, *et al.*, “A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches,” *Computer Science Review*, vol. 53, p. 100 656, 2024.

[64] N. C. Luong, D. T. Hoang, S. Gong, *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE communications surveys & tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[65] D. T. Campbell and J. C. Stanley, *Experimental and quasi-experimental designs for research.* Ravenio books, 2015.

[66] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, *et al.*, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on software engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[67] G. Heitink, *Practical theology: History, theory, action domains: Manual for practical theology.* Wm. B. Eerdmans Publishing, 1999.

[68] A. Dresch, D. P. Lacerda, and J. A. V. Antunes, "Design science research," in *Design science research*, Springer, 2015, pp. 67–102.

[69] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.

[70] P. J. Denning, "A new social contract for research," *Communications of the ACM*, vol. 40, no. 2, pp. 132–134, 1997.

[71] P. Y. Papalambros, "Design science: Why, what and how," *Design Science*, vol. 1, 2015.

[72] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[73] C. Wohlin and P. Runeson, "Guiding the selection of research methodology in industry–academia collaboration in software engineering," *Information and Software Technology*, vol. 140, p. 106 678, 2021.

[74] A. Strauss and J. Corbin, *Basics of qualitative research.* Sage publications, 1990.

[75] A. Castleberry and A. Nolen, "Thematic analysis of qualitative research data: Is it as easy as it sounds?" *Currents in pharmacy teaching and learning*, vol. 10, no. 6, pp. 807–815, 2018.

[76] A. Fink, *Conducting research literature reviews: From the internet to paper.* Sage publications, 2019.

[77] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[78] Z. Stapic, E. G. López, A. G. Cabot, L. de Marcos Ortega, and V. Strahonja, "Performing systematic literature review in software engineering," in *Central European Conference on Information and Intelligent Systems*, Faculty of Organization and Informatics Varazdin, 2012, p. 441.

[79] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571–583, 2007.

[80] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, pp. 131–164, 2009.

[81] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons, 2012.

[82] R. K. Yin, *Case study research and applications*, 2018.

[83] D. E. Perry, S. E. Sim, and S. M. Easterbrook, "Case studies for software engineers," in *Proceedings. 26th International Conference on Software Engineering*, IEEE, 2004, pp. 736–738.

[84] R. E. Stake, "Qualitative case studies.," 2008.

[85] C. Robson and K. McCartan, *Real world research: a resource for users of social research methods in applied settings.* Wiley, 2016.

[86] K. L. Barriball and A. While, "Collecting data using a semi-structured interview: A discussion paper," *Journal of Advanced Nursing-Institutional Subscription*, vol. 19, no. 2, pp. 328–335, 1994.

[87] S. Baskarada, "Qualitative case study guidelines," *Baškarada, S.(2014). Qualitative case studies guidelines. The Qualitative Report*, vol. 19, no. 40, pp. 1–25, 2014.

[88] O. Kempthorne, "The design and analysis of experiments.," 1952.

[89] M. Gutbrod, J. Münch, and M. Tichy, "How do software startups approach experimentation? empirical results from a qualitative interview study," in *International Conference on Product-Focused Software Process Improvement*, Springer, 2017, pp. 297–304.

[90] A. M. Christie, "Simulation: An enabling technology in software engineering," *CROSSTALK–The Journal of Defense Software Engineering*, vol. 12, no. 4, pp. 25–30, 1999.

[91] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2017.

[92] D. Zhang and J. J. Tsai, "Machine learning and software engineering," *Software Quality Journal*, vol. 11, no. 2, pp. 87–119, 2003.

[93] "Collaborating partners, software center." (), [Online]. Available: `https://www.software-center.se/` (visited on 03/15/2022).

[94] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[95] L. Bickman, *Validity and social experimentation*. Sage, 2000, vol. 1.

[96] W. R. Shadish, T. D. Cook, D. T. Campbell, *et al.*, *Experimental and quasi-experimental designs for generalized causal inference/William R. Shedish, Thomas D. Cook, Donald T. Campbell.* Boston: Houghton Mifflin, 2002.

[97] L. Bickman and D. J. Rog, *The SAGE handbook of applied social research methods*. Sage publications, 2008.

[98] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[99] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[100] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*, Citeseer, 2014, pp. 48–55.

[101] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[102] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, *et al.*, "Communication efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[103] T. Yang, G. Andrew, H. Eichner, *et al.*, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[104] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[105] R. Feldt. "Isi se journals (ranked)." (), [Online]. Available: `http://www.robertfeldt.net/advice/se%5C_venues/`.

[106] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[107] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1387–1395.

[108] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *arXiv preprint arXiv:1910.06837*, 2019.

[109] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, 2020.

[110] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[111] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[112] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 6341–6345.

[113] M. Ammad-ud-din, E. Ivannikova, S. A. Khan, *et al.*, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[114] Y. Liu, A. Huang, Y. Luo, *et al.*, "Fedvision: An online visual object detection platform powered by federated learning," *arXiv preprint arXiv:2001.06202*, 2020.

[115] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–7.

[116] S. Lu, Y. Yao, and W. Shi, "Collaborative learning on the edges: A case study on connected vehicles," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[117] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," *arXiv preprint arXiv:1909.00907*, 2019.

[118] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated learning in the sky: Joint power allocation and scheduling with uav swarms," *arXiv preprint arXiv:2002.08196*, 2020.

[119] W. Zhou, Y. Li, S. Chen, and B. Ding, "Real-time data processing architecture for multi-robots based on differential federated learning," in *2018 IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, IEEE, 2018, pp. 462–471.

[120] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: A secure, decentralized and privacy-preserving system," *arXiv preprint arXiv:1906.10893*, 2019.

[121] A. B. Sada, M. A. Bouras, J. Ma, H. Runhe, and H. Ning, "A distributed video analytics architecture based on edge-computing and federated learning," in *2019 IEEE Intl Conf on DASC/PiCom/CBDCom/CyberSciTech*, IEEE, 2019, pp. 215–220.

[122] J. Mills, J. Hu, and G. Min, “Communication-efficient federated learning for wireless edge intelligence in iot,” *IEEE Internet of Things Journal*, 2019.

[123] S. Savazzi, M. Nicoli, and V. Rampa, “Federated learning with cooperating devices: A consensus approach for massive iot networks,” *IEEE Internet of Things Journal*, 2020.

[124] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, “Split learning for health: Distributed deep learning without sharing raw patient data,” *arXiv preprint arXiv:1812.00564*, 2018.

[125] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, “Loadaboost: Loss-based adaboost federated machine learning on medical data,” *arXiv preprint arXiv:1811.12629*, 2018.

[126] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, “Federated learning of predictive models from federated electronic health records,” *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[127] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, “Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records,” *Journal of Biomedical Informatics*, vol. 99, p. 103 291, 2019.

[128] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, “Efficient and privacy-enhanced federated learning for industrial artificial intelligence,” *IEEE Transactions on Industrial Informatics*, 2019.

[129] D. Verma, S. Julier, and G. Cirincione, “Federated ai for building ai solutions across multiple agencies,” *arXiv preprint arXiv:1809.10036*, 2018.

[130] M. R. Sprague, A. Jalalirad, M. Scavuzzo, *et al.*, “Asynchronous federated learning for geospatial applications,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2018, pp. 21–28.

[131] K. Sozinov, V. Vlassov, and S. Girdzijauskas, “Human activity recognition using federated learning,” in *2018 IEEE Intl Conf on ISPA/IUCC/BDCloud/SocialCom/SustainCom*, IEEE, 2018, pp. 1103–1111.

[132] B. Hu, Y. Gao, L. Liu, and H. Ma, "Federated region-learning: An edge computing based framework for urban environment sensing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–7.

[133] D. Shultz, *When your voice betrays you*, 2015.

[134] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[135] M. Goddard, "The eu general data protection regulation (gdpr): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017.

[136] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[137] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021.

[138] G. Walsham, "Interpretive case studies in is research: Nature and method," *European Journal of information systems*, vol. 4, no. 2, pp. 74–81, 1995.

[139] R. K. Yin, *Case study research and applications: Design and methods.* Sage publications, 2017.

[140] J. A. Maxwell, *Qualitative research design: An interactive approach.* Sage publications, 2012, vol. 41.

[141] L. Sgier, "Qualitative data analysis," *An Initiat. Gebert Ruf Stift*, vol. 19, pp. 19–21, 2012.

[142] C. Rivas, "Coding and analysing qualitative data," *Researching society and culture*, vol. 3, no. 2012, pp. 367–392, 2012.

[143] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data-ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[144] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation," *arXiv preprint arXiv:1903.07424*, 2019.

[145] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[146] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[147] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[148] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[149] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[150] J. Bosch, I. Crnkovic, and H. H. Olsson, *Engineering ai systems: A research agenda*, 2020.

[151] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A taxonomy of software engineering challenges for machine learning systems: An empirical investigation," in *International Conference on Agile Software Development*, Springer, Cham, 2019, pp. 227–243.

[152] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2019, pp. 74–90.

[153] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[154] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1485–1488.

[155] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[156] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[157] P. J. Navarro, C. Fernandez, R. Borraz, and D. Alonso, “A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data,” *Sensors*, vol. 17, no. 1, p. 18, 2017.

[158] H. Zhang, J. Bosch, and H. H. Olsson, “End-to-end federated learning for autonomous driving vehicles,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021.

[159] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” *arXiv preprint arXiv:1907.02189*, 2019.

[160] T. Park, N. Abuzainab, and W. Saad, “Learning how to communicate in the internet of things: Finite resources and heterogeneity,” *arXiv preprint arXiv:1610.01586*, 2016.

[161] D. Pomerleau, “An autonomous land vehicle in a neural network,” *Advances in neural information processing systems’(Morgan Kaufmann Publishers Inc., 1989)*, vol. 1, 1998.

[162] H. M. Eraqi, M. N. Moustafa, and J. Honer, “End-to-end deep learning for steering autonomous vehicles considering temporal dependencies,” *arXiv preprint arXiv:1710.03804*, 2017.

[163] S. Du, H. Guo, and A. Simpson, “Self-driving car steering angle prediction based on image recognition,” *arXiv preprint arXiv:1912.05440*, 2019.

[164] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[165] F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, “Comparing offline and online testing of deep neural networks: An autonomous car case study,” in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, IEEE, 2020, pp. 85–95.

[166] R. Valiente, M. Zaman, S. Ozer, and Y. P. Fallah, “Controlling steering angle for cooperative self-driving vehicles utilizing cnn and lstm-based deep networks,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2423–2428.

[167] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[168] N. Fernandez, "Two-stream convolutional networks for end-to-end learning of self-driving cars," *arXiv preprint arXiv:1811.05785*, 2018.

[169] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, 2017.

[170] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[171] S. Doomra, N. Kohli, and S. Athavale, "Turn signal prediction: A federated learning case study," *arXiv preprint arXiv:2012.12401*, 2020.

[172] SullyChen. "Collection of labeled car driving datasets." (2018), [Online]. Available: `https://github.com/SullyChen/driving-datasets`.

[173] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, International Society for Optics and Photonics, vol. 281, 1981, pp. 319–331.

[174] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*, Springer, 2003, pp. 363–370.

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

[176] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[177] A. Sjöberg, E. Gustavsson, A. C. Koppisetty, and M. Jirstrand, "Federated learning of deep neural decision forests," in *International Conference on Machine Learning, Optimization, and Data Science*, Springer, 2019, pp. 700–710.

[178] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1467–1475.

[179] L. Rokach, "Decision forest: Twenty years of research," *Information Fusion*, vol. 27, pp. 111–125, 2016.

[180] H. Zhang, J. Bosch, and H. H. Olsson, *Real-time end-to-end federated learning: An automotive case study*, 2021.

[181] FLIR. "Flir thermal dataset for algorithm training." (2018), [Online]. Available: `https://www.flir.in/oem/adas/adas-dataset-form/`.

[182] F. Yu, H. Chen, X. Wang, *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[183] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[184] A. Shah, E. Kadam, H. Shah, S. Shinde, and S. Shingade, "Deep residual networks with exponential linear unit," in *Proceedings of the Third International Symposium on Computer Vision and the Internet*, 2016, pp. 59–65.

[185] P. Larrañaga, D. Atienza, J. Diaz-Rozo, A. Ogbechie, C. Puerto-Santana, and C. Bielza, *Industrial applications of machine learning.* CRC press, 2018.

[186] E. Zhang and Y. Zhang, "Average precision," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 192–193, ISBN: 978-0-387-39940-9.

[187] Scikit-learn. "Scikit-learn: Average precision." (2007), [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html`.

[188] J. Li, K. K. Nagalapur, E. Stare, *et al.*, "5G New Radio for Public Safety Mission Critical Communications," *arXiv preprint arXiv:2103.02434*, 2021.

[189] S. A. R. Naqvi, S. A. Hassan, H. Pervaiz, and Q. Ni, "Drone-aided communication as a key enabler for 5G and resilient public safety networks," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 36–42, 2018.

[190] K. P. Morison and J. Calahorrano. "FirstNet Case Study: How FirstNet Deployables are Supporting Public Safety." (2020), [Online]. Available: `https://www.policeforum.org/assets/FirstNetDeployables.pdf`.

[191] A. Merwaday, A. Tuncer, A. Kumbhar, and I. Guvenc, "Improved throughput coverage in natural disasters: Unmanned aerial base stations for public-safety communications," *IEEE Vehicular Technology Magazine*, vol. 11, no. 4, pp. 53–60, 2016.

[192] L. Ferranti, L. Bonati, S. D'Oro, and T. Melodia, "SkyCell: A prototyping platform for 5G aerial base stations," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, IEEE, 2020, pp. 329–334.

[193] H. Wang, H. Zhao, W. Wu, J. Xiong, D. Ma, and J. Wei, "Deployment algorithms of flying base stations: 5G and beyond with UAVs," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 009–10 027, 2019.

[194] E. Kalantari, M. Z. Shakir, H. Yanikomeroglu, and A. Yongacoglu, "Backhaul-aware robust 3D drone placement in 5G+ wireless networks," in *2017 IEEE international conference on communications workshops (ICC workshops)*, IEEE, 2017, pp. 109–114.

[195] C. T. Cicek, H. Gultekin, B. Tavli, and H. Yanikomeroglu, "Backhaul-aware optimization of UAV base station location and bandwidth allocation for profit maximization," *IEEE Access*, vol. 8, pp. 154 573–154 588, 2020.

[196] N. Tafintsev, D. Moltchanov, M. Gerasimenko, *et al.*, "Aerial access and backhaul in mmWave B5G systems: Performance dynamics and optimization," *IEEE Communications Magazine*, vol. 58, no. 2, pp. 93–99, 2020.

[197] C. Madapatha, B. Makki, C. Fang, *et al.*, "On integrated access and backhaul networks: Current status and potentials," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1374–1389, 2020.

[198] 3GPP, "Study on enhancement for data collection for NR and ENDC," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 37.817, Nov. 2021, Version 0.3.0.

[199] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[200] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[201] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[202] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[203] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[204] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[205] Z. Ding and H. Dong, "Challenges of reinforcement learning," in *Deep Reinforcement Learning*, Springer, 2020, pp. 249–272.

[206] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[207] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[208] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[209] H. Shakhatreh, K. Hayajneh, K. Bani-Hani, A. Sawalmeh, and M. Anan, "Cell on wheels-unmanned aerial vehicle system for providing wireless coverage in emergency situations," *Complexity*, vol. 2021, 2021.

[210] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[211] J. Li, X. Lin, K. K. Nagalapur, *et al.*, "Towards providing connectivity when and where it counts: An overview of deployable 5G networks," *arXiv preprint arXiv:2110.05360*, 2021.

[212] Z. Qi, A. Lahuerta-Lavieja, J. Li, and K. K. Nagalapur, “Deployable networks for public safety in 5g and beyond: A coverage and interference study,” in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 346–351.

[213] S. A. Al-Ahmed, M. Z. Shakir, and S. A. R. Zaidi, “Optimal 3D UAV base station placement by considering autonomous coverage hole detection, wireless backhaul and user demand,” *Journal of Communications and Networks*, vol. 22, no. 6, pp. 467–475, 2020.

[214] A. Fouda, A. S. Ibrahim, Í. Güvenç, and M. Ghosh, “Interference management in UAV-assisted integrated access and backhaul cellular networks,” *IEEE Access*, vol. 7, pp. 104 553–104 566, 2019.

[215] M.-A. Lahmeri, M. A. Kishk, and M.-S. Alouini, “Artificial intelligence for UAV-enabled wireless networks: A survey,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1015–1040, 2021.

[216] A. Ly and Y.-D. Yao, “A review of deep learning in 5G research: Channel coding, massive MIMO, multiple access, resource allocation, and network security,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 396–408, 2021.

[217] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, “Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, 2018.

[218] C. Madapatha, B. Makki, A. Muhammad, E. Dahlman, M.-S. Alouini, and T. Svensson, “On topology optimization and routing in integrated access and backhaul networks: A genetic algorithm-based approach,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2273–2291, 2021.

[219] F. Tang, Y. Zhou, and N. Kato, “Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 12, pp. 2773–2782, 2020.

[220] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, “Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, 2021.

[221] C. Zhou, W. Wu, H. He, *et al.*, “Deep reinforcement learning for delay-oriented IoT task scheduling in sagin,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 911–925, 2021.

[222] S. Tan, C. Dun, F. Jin, and K. Xu, “UAV control in smart city based on space-air-ground integrated network,” in *2021 International Conference on Internet, Education and Information Technology (IEIT)*, 2021, pp. 324–328.

[223] L. Zhang, A. Celik, S. Dang, and B. Shihada, “Energy-efficient trajectory optimization for UAV-assisted IoT networks,” *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[224] S. Yin, S. Zhao, Y. Zhao, and F. R. Yu, “Intelligent trajectory design in UAV-aided communications with reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8227–8231, 2019.

[225] S. Yin and F. R. Yu, “Resource allocation and trajectory design in UAV-aided cellular networks based on multi-agent reinforcement learning,” *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[226] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, “Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 12, pp. 2783–2797, 2020.

[227] Y. Yang and X. Liu, “Deep reinforcement learning based trajectory optimization for uav-enabled iot with swipt,” *Ad Hoc Networks*, vol. 159, p. 103 488, 2024.

[228] C. Deng, X. Fang, and X. Wang, “Beamforming design and trajectory optimization for uav-empowered adaptable integrated sensing and communication,” *IEEE Transactions on Wireless Communications*, 2023.

[229] S. M. Abohashish, R. Y. Rizk, and E. Elsedimy, “Trajectory optimization for uav-assisted relay over 5g networks based on reinforcement learning framework,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, no. 1, p. 55, 2023.

[230] W. Liu, D. Li, T. Liang, T. Zhang, Z. Lin, and N. Al-Dhahir, "Joint trajectory and scheduling optimization for age of synchronization minimization in uav-assisted networks with random updates," *IEEE Transactions on Communications*, vol. 71, no. 11, pp. 6633–6646, 2023.

[231] C. Zhang, Z. Li, C. He, K. Wang, and C. Pan, "Deep reinforcement learning based trajectory design and resource allocation for uav-assisted communications," *IEEE Communications Letters*, vol. 27, no. 9, pp. 2398–2402, 2023.

[232] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[233] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6180–6190, 2020.

[234] H. Huang, Y. Yang, H. Wang, Z. Ding, H. Sari, and F. Adachi, "Deep reinforcement learning for UAV navigation through massive MIMO technique," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1117–1121, 2020.

[235] Y. Zeng, X. Xu, S. Jin, and R. Zhang, "Simultaneous navigation and radio mapping for cellular-connected UAV with deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 7, pp. 4205–4220, 2021.

[236] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[237] S. Ahmed, M. Z. Chowdhury, and Y. M. Jang, "Energy-efficient UAV relaying communications to serve ground nodes," *IEEE Communications Letters*, vol. 24, no. 4, pp. 849–852, 2020.

[238] C. Zhao, J. Liu, M. Sheng, W. Teng, Y. Zheng, and J. Li, "Multi-UAV trajectory planning for energy-efficient content coverage: A decentralized learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3193–3207, 2021.

[239] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9540–9554, 2021.

[240] H. Zhang, J. Li, Z. Qi, *et al.*, "Autonomous navigation and configuration of integrated access backhauling for UAV base station using reinforcement learning," *arXiv preprint arXiv:2112.07313*, 2021.

[241] 3GPP–, "Study on Enhanced LTE Support for Aerial Vehicles," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 36.777, Dec. 2017, Version 15.0.0.

[242] 3GPP—, "Radio Resource Control (RRC) protocol specification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.331, Mar. 2022, Version 16.8.0.

[243] 3GPP-, "Study on channel model for frequencies from 0.5 to 100 GHz," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.901, Jan. 2020, Version 16.1.0.

[244] B. Galkin, E. Fonseca, R. Amer, L. A. DaSilva, and I. Dusparic, "Reqiba: Regression and deep q-learning for intelligent uav cellular user to base station association," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 5–20, 2022.

[245] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[246] H. Zhang, J. Bosch, and H. H. Olsson, "Real-time end-to-end federated learning: An automotive case study," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, 2021, pp. 459–468.

[247] S. Pandya, G. Srivastava, R. Jhaveri, *et al.*, "Federated learning for smart cities: A comprehensive survey," *Sustainable Energy Technologies and Assessments*, vol. 55, p. 102 987, 2023.

[248] A. Rauniyar, D. H. Hagos, D. Jha, *et al.*, "Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions," *IEEE Internet of Things Journal*, 2023.

[249] H. Tao, M. Z. A. Bhuiyan, M. A. Rahman, *et al.*, "Economic perspective analysis of protecting big data security and privacy," *Future Generation Computer Systems*, vol. 98, pp. 660–671, 2019.

[250] Y. Dai, Z. Chen, J. Li, S. Heinecke, L. Sun, and R. Xu, "Tackling data heterogeneity in federated learning with class prototypes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 7314–7322.

[251] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Computing*, vol. 1, no. 1, p. 100 008, 2021.

[252] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[253] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–44, 2023.

[254] L. Zhang, X. Lei, Y. Shi, H. Huang, and C. Chen, "Federated learning with domain generalization," *arXiv preprint arXiv:2111.10487*, 2021.

[255] D. Gao, H. Wang, X. Guo, *et al.*, "Federated learning based on ctc for heterogeneous internet of things," *IEEE Internet of Things Journal*, 2023.

[256] W. Huang, M. Ye, Z. Shi, H. Li, and B. Du, "Rethinking federated learning with domain shift: A prototype view," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2023, pp. 16 312–16 322.

[257] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: Challenges and opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 2020.

[258] E. Gabrielli, G. Pica, and G. Tolomei, "A survey on decentralized federated learning," *arXiv preprint arXiv:2308.04604*, 2023.

[259] G. Liu, C. Wang, X. Ma, and Y. Yang, "Keep your data locally: Federated-learning-based data privacy preservation in edge computing," *IEEE Network*, vol. 35, no. 2, pp. 60–66, 2021.

[260] TensorFlow, "Tensorflow federated, machine learning on decentralized data," 2021.

[261] A. Ziller, A. Trask, A. Lopardo, *et al.*, "Pysyft: A library for easy federated learning," *Federated Learning Systems: Towards Next-Generation AI*, pp. 111–139, 2021.

[262] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 320–10 325, 2021.

[263] S. Caldas, S. M. K. Duddu, P. Wu, *et al.*, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[264] Y. Ma, D. Yu, T. Wu, and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice," *Frontiers of Data and Domputing*, vol. 1, no. 1, pp. 105–115, 2019.

[265] A. Tariq, M. A. Serhani, F. Sallabi, T. Qayyum, E. S. Barka, and K. A. Shuaib, "Trustworthy federated learning: A survey," *arXiv preprint arXiv:2305.11537*, 2023.

[266] L. Yuan, Z. Wang, L. Sun, S. Y. Philip, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet of Things Journal*, 2024.

[267] M. Chahoud, S. Otoum, and A. Mourad, "On the feasibility of federated learning towards on-demand client deployment at the edge," *Information Processing & Management*, vol. 60, no. 1, p. 103 150, 2023.

[268] D. I. Sjoberg, B. Anda, E. Arisholm, *et al.*, "Conducting realistic experiments in software engineering," in *Proceedings international symposium on empirical software engineering*, IEEE, 2002, pp. 17–26.

[269] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, *et al.*, *Experimentation in software engineering.* Springer, 2012, vol. 236.

[270] B. Kitchenham, S. Charters, *et al.*, *Guidelines for performing systematic literature reviews in software engineering*, 2007.

[271] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[272] F. Soppelsa and C. Kaewkasi, *Native docker clustering with swarm.* Packt Publishing Ltd, 2016.

[273] Y. Gao, M. Kim, S. Abuadbba, *et al.*, "End-to-end evaluation of federated learning and split learning for internet of things," *arXiv preprint arXiv:2003.13376*, 2020.

[274] R. Kazman and H.-M. Chen, "The architecture of complexity revisited: Design primitives for ultra-large-scale systems," 2023.

[275] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150.

[276] N. M. Ali, M. M. Abd El Hamid, and A. Youssif, "Sentiment analysis for movies reviews dataset using deep learning models," *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol*, vol. 9, 2019.

[277] S. M. Qaisar, "Sentiment analysis of imdb movie reviews using long short-term memory," in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, IEEE, 2020, pp. 1–4.