THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Towards Real-World Federated Learning: Empirical Studies in the Domain of Embedded Systems

HONGYI ZHANG

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2022

Towards Real-World Federated Learning: Empirical Studies in the Domain of Embedded Systems

Hongyi Zhang

ISSN 1652-876X
This thesis has been prepared using LaTeX.

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

Printed by Chalmers Reproservice
Gothenburg, Sweden, April 2022

To my family

Abstract

Context: Artificial intelligence (AI) has led a new phase of technical revolution and industrial development around the world since the twenty-first century, revolutionizing the way of production. Artificial intelligence (AI), an emerging information technology, is thriving, and AI application technologies are gaining traction, particularly in professional services such as healthcare, education, finance, security, etc. More machine learning technologies have begun to be thoroughly applied to the production stage as big data and cloud computing capabilities have improved. With the increased focus on Machine Learning applications and the rapid growth of distributed edge devices in the industry, we believe that utilizing a large number of edge devices will become increasingly important.

The introduction of Federated Learning changes the situation in which data must be centrally uploaded to the cloud for processing and maximizes the use of edge devices' computing and storage capabilities. With local data processing, the learning approach eliminates the need to upload large amounts of local data and reduces data transfer latency. Because Federated Learning does not require centralized data for model training, it is better suited to edge learning scenarios with limited data and privacy concerns.

Objective: The purpose of this research is to identify the characteristics and problems of the Federated Learning methods, offer new algorithms and frameworks that can assist companies in making the transition to Federated Learning, and empirically validate the proposed approaches.

Method: To achieve these objectives, we adopted an empirical research approach with design science being our primary research method. We conducted a literature review, case studies, including semi-structured interviews and simulation experiments in close collaboration with software-intensive companies in the embedded systems domain.

Results: We present four major findings in this paper. First, we present a state-of-the-art review of the empirical results reported in the existing Federated Learning literature. We then categorize those Federated Learning implementations into different application domains, identify their challenges, and propose six open research questions based on the problems identified in the literature. Second, we conduct a case study to explain why companies anticipate Federated Learning as a potential solution to the challenges they encountered when implementing machine learning components. We summarize the services that a comprehensive Federated Learning system must enable in industrial settings. Furthermore, we identify the primary barriers that companies must overcome in order to embrace and transition to Federated Learning. Based on our empirical findings, we propose five requirements for companies implementing reliable Federated Learning systems. Third, we develop and evaluate four architecture alternatives for a Federated Learning system, including centralized, hierarchical, regional, and decentralized architectures. We investigate the trade-off between communication latency, model evolution time, and model classification performance, which is critical for applying our findings to real-world industrial systems. Fourth, we introduce techniques and asynchronous frameworks for end-to-end on-device Federated Learning. The method is validated using a steering wheel angle prediction case. The local models of each edge vehicle can be continuously trained and shared with other vehicles to improve their local model prediction accuracy. Furthermore, we combine the asynchronous Federated Learning approach with Deep Neural Decision Forests and validate our method using important industry use cases in the automotive domain. Our findings show that Federated Learning can improve model training speed while lowering communication overhead without sacrificing accuracy, demonstrating that this technique has significant benefits to a wide range of real-world embedded systems.

Future Work: In the future, we plan to test our approach in other use cases and look into more sophisticated neural networks integrated with our approach. In order to improve model training performance on resource-constrained edge devices in real-world embedded systems, we intend to design more appropriate aggregation methods and protocols. Furthermore, we intend to use the Federated Learning and Reinforcement Learning methods to assist the edge in evolving themselves autonomously and fully utilizing the computation capabilities of the edge devices.

Keywords: Federated 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.

Other publications by the author, not included in this thesis, are:

[F] **Zhang H.**, Bosch J. and Olsson H.H., “End-to-end federated learning for autonomous driving vehicles”. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.

[G] Dakkak A., **Zhang H.**, Mattos D.I., Bosch J. and Olsson H.H., “Towards Continuous Data Collection from In-service Products: Exploring the Relation Between Data Dimensions and Collection Challenges”. In *2021 28th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 243-252). IEEE.

Personal Contributions

For all publications where I am the first author, my contribution is listed below using the CRediT (Contributor Roles Taxonomy) author statement [1]:

1. Conceptualization – Formulation of overarching research goals and aims
2. Methodology – Development or design of methodology; creation of models
3. Validation - Focus on the overall reproducibility of results
4. Investigation - Conducting a research and performing data collection.
5. Data Curation – Activities to annotate scrub data and maintain research data
6. Writing – Original draft, review, and editing
7. Preparation – Creation and/or presentation of the published work
8. Project Administration – Management and coordination responsibility for research activities.

For the mentioned paper G, in which I am listed as a second author, I made the following contributions:

1. Conceptualization – Formulation of overarching research goals and aims
2. Methodology (partly) – Design of methodology and creation of research models.
3. Investigation – Conducting the research process and performing data collection
4. Data Curation – Activities to annotate scrub data and maintain research data
5. Writing (partly) – Original draft, review, and editing.
6. Project Administration – Management and coordination responsibility for research activities

Acknowledgments

First and foremost, I'd want 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'd want to express my gratitude to my co-supervisor, Helena Holmström Olsson. I'd 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'd want 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'd like to thank my wife, Lida, for her unending support and encouragement in my daily life, as well as my kittens, who provided me with emotional support during my studies.

Last but not least, I'd like to thank the Software Center and company participants 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

Abstract i

List of Papers iv

Personal Contributions vi

Acknowledgements vii

1 Introduction 1

1.1 Artificial Intelligence and Machine Learning 2

1.2 Challenges of the Typical Machine Learning Workflow 5

Data Privacy 5

Communication Efficiency 6

Data Ownership and Storage 7

1.3 Transition towards Federated Learning 8

1.4 Summary of the Included Papers 9

1.5 Contribution of the Research 10

1.6 Structure of the thesis 11

2 Background 13

2.1 Machine Learning 14

Supervised Learning 14

Unsupervised Learning 15
Reinforcement Learning 16
Model Training and Inference 16
2.2 Deep Learning 17
2.3 Embedded Systems 18
2.4 AI-Powered Applications 19
2.5 Federated Learning 21
2.6 Summary 23

3 Research Methodology and Design 25
3.1 Research Questions 25
3.2 Empirical Research 27
3.3 Design Science 27
3.4 Data Collection and Evaluation Methods 31
Literature Review 31
Case Study 31
3.5 Research Design 36
Problem Identification 37
Solution Proposal and Validation 39
3.6 Industrial Collaboration 40
3.7 Threats to Validity 44
3.8 Summary 46

4 Engineering Federated Learning Systems: A Literature Review 47
4.1 Research Method 49
Search Process 49
Inclusion and exclusion criteria 50
Results of the Literature Search Process 50
4.2 Existing Federated Learning Systems 51
4.3 Discussion 55
4.4 Conclusion 56

5 Towards Federated Learning: A Case Study in the Telecommunication Domain 59
5.1 Background 61

5.2 Research Methodology 62
Data Collection and Analysis 63
Case Company 65
Use Cases and Participants 65
Threats to validity 66
5.3 Empirical findings: Towards Federated Learning 67
Benefits of Implementing Federated Learning 67
Transition to Federated Learning 69
5.4 Discussion 71
Benefits 71
Learning Services 71
Challenges 73
Criteria for a Reliable Federated Learning System 78
5.5 Conclusions 80

6 Federated Learning Systems: Architecture Alternatives 81
6.1 Architecture Alternatives 85
Centralized Architecture 85
Hierarchical Architecture 87
Regional Architecture 87
Decentralized Architecture 88
6.2 Research Method 89
Simulation Testbed 89
Training Data Distribution 90
Machine Learning Method 92
Evaluation Metrics 93
6.3 Algorithms used in each architecture alternative 94
Centralized Architecture 95
Hierarchical Architecture 96
Regional Architecture 98
Decentralized Architecture 99
6.4 Evaluation 99
Weights update latency 101
Model Evolution Time 103
Classification Accuracy 104
6.5 Discussion 107
Centralized 108

Hierarchical . . . 110
Regional . . . 110
Decentralized . . . 111
6.6 Conclusion and Future Work . . . 111

7 Real-time End-to-End Federated Learning: An Automotive Case Study 113
7.1 Background . . . 116
7.2 Related Work . . . 117
Steering Wheel Angle Prediction . . . 117
Federated Learning in Automotive . . . 118
7.3 Method . . . 119
Mathematical Notations . . . 119
Data Traces and Testbed . . . 120
Machine Learning Method . . . 123
Evaluation Metrics and Baseline Model . . . 124
7.4 Real-time End-to-End Federated Learning . . . 127
7.5 Results . . . 132
7.6 Discussion . . . 137
7.7 Conclusion and Future Work . . . 137

8 AF-DNDF: Asynchronous Federated Learning of Deep Neural Decision Forests 139
8.1 Background . . . 142
Deep Neural Decision Forests . . . 142
Federated Learning and Asynchronous Aggregation . . . 144
8.2 Method . . . 145
Data Traces and Testbed . . . 146
Machine Learning Method . . . 148
Evaluation Metrics and Baseline Model . . . 149
8.3 AF-DNDF: Asynchronous Federated Deep Neural Decision Forests 152
8.4 Results . . . 156
Classification Performance . . . 156
Model Training Efficiency . . . 158
Bandwidth Utilization . . . 158
8.5 Discussion . . . 161
8.6 Conclusion . . . 162

9 Concluding Remarks and Future Work 163

9.1 Key Contributions 166

9.2 Future Work 168

References 171

CHAPTER 1

Introduction

The sheer volume of data generated by humans and computers nowadays significantly exceeds humans' ability to absorb, comprehend, and make sophisticated judgments based on it. Artificial intelligence (AI) underpins all computer learning and is the future of data-driven and complex decision-making [1]. There are 255,168 distinct moves in tic-tac-toe (the circle and cross-game), for example, 46,080 of which result in a draw. Nonetheless, the majority of people can figure out how to avoid losing the game. Checkers, on the other hand, has nearly 5,000 moves to the power of 18, and therefore only a small number of people with long-term practices can be considered specialists. However, with a sufficient amount of data, computers are extremely efficient at calculating combinations of these moves and generating the best possible response. Artificial Intelligence's capabilities can benefit many aspects of human life and serve as the foundation for future business decisions. [2].

In most industrial applications, AI can enhance the intelligence of the products and improve user experience. AI will not be sold as a standalone application in most circumstances. Instead, AI skills will be employed to augment existing products, similar to how Siri is included in Apple products. Massive amounts of data can be used to improve a variety of home and workplace technologies, ranging from security intelligence to investment analytics. [3]. Furthermore, AI can be adapted by learning algorithms, allowing users' data to continuously improve AI prediction performance. As the result, just as an algorithm can teach itself how to play chess, it can also teach itself which things to recommend to the target users. When presented with new data, back-propagation will allow the model to be updated by training and adding data [4].

1.1 Artificial Intelligence and Machine Learning

Machine learning, as one of the most important general-purpose AI technologies, is frequently regarded by outsiders as a formula or theorem-like abstract basis employed in AI applications. Machine learning is functional in nature, but with its purely algorithmic capabilities, it can still be directly implemented in areas with a good digital foundation, such as finance, industry, medicine, and the internet, providing companies with a variety of services such as intelligent risk control, predictive maintenance and personalized recommendations [5][6][7].

The main value of machine learning is to analyze known data using specialized algorithms, find the possibilities buried in the data, and make predictions and judgments based on this, either independently or with the assistance of users. The existence of a large amount of data available for analysis is a prerequisite for the value of machine learning, particularly in the practical application of enterprises, which requires enterprises to be able to provide continuous and accurate data on various aspects of business such as research and design, production and operation, marketing and customer acquisition, and so on. In order to train, modify, and improve the algorithm model, and then use the model to explore the real value of the data collected by the companies, the data collected will be the fuel of the AI techniques [8]. The degree to which industry or region gets digitized impacts how much machine learning can play a part in it.

Machine learning algorithms are a class of algorithms that analyze data automatically to extract patterns and then use those patterns to create predictions from unknown data [9]. With the help of learning iterations, the method can help industries learn from previously generated dependable calculations, decisions, and outcomes. Although many machine learning techniques have been known for quite some time, the capacity to automatically apply difficult mathematical computations to large amounts of data is the most recent advancement. Computing and storage capability is more economical and powerful compared to the data availability and volume growth. All of the preceding criteria imply that software engineers can develop machine learning models to quickly and automatically analyze much more complicated data, producing faster and accurate results even at an extremely large scale. When there is no human intervention in reality, high-value predictions can lead to better judgments and more informed behavior.

Artificial intelligence is now seen as a must-have technology for modern enterprises. The future of business will be based on AI-driven systems that continually employ modeling and simulation to identify the best next step [10][11]. Simultaneously, as the amount of data generated in workflows grows, machine learning models will also evolve and play a much more essential role. Businesses will use AI technologies to cut costs, speed up decision-making cycles, encourage creativity, and enable greater disruption. Nowadays, AI is playing a critical role in assisting businesses in processing data at a "superhuman" scale and precision. Intelligent systems are progressively taking on the task of processing and preparing data for analysis.

IoT data, user data, and internal corporate data are now the most common types of data linked to industrial AI applications. In many cases, this is simply due to the fact that organizations have access to a vast amount of data when interacting with users and employing IoT devices. Other types of data, on the other hand, can frequently lead to even superior insights, especially when many types of data are fused. However, the important concern is whether organizations and current technology infrastructures can handle large amounts of data. It is critical to recognize that as the second wave of IoT solutions becomes available, solutions will continue to evolve as 5G offers greater bandwidth and reduced latency, and the volume of data continues to grow. For example, IoT sensors incorporated in products will improve the user experience or enable process owners to offer virtual asset monitoring, continuously adjusting for optimal performance and applying data insights from third parties [12]. Clearly, the usage of machines will increase in the future. Many companies are already aware that humans cannot handle the current level of data. They require machine assistance to more efficiently

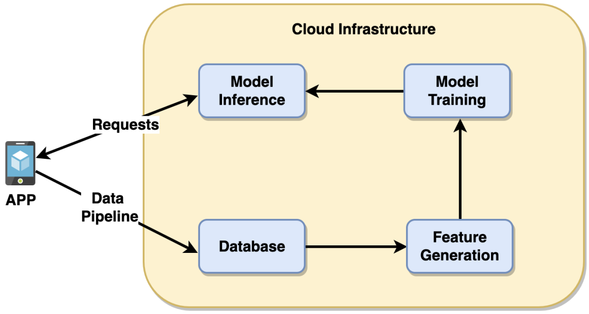


Figure 1.1: Typical machine learning workflow in the domain of embedded systems

organize data and employ machine learning software for databases to make them economically valuable.

Machine learning workflows specify which steps of a machine learning project are carried out. typical processes including data gathering, data pre-processing, model training and refinement, assessment, and deployment to production [9]. Despite the fact that there are several ways to automate the machine learning process in the area of embedded systems, the essence of the processes remains the same [13]. According to [14][15], a typical machine learning workflow in the domain of embedded systems is shown in Figure 1.1. The typical workflow begins with logs collected from the app, server or user devices. The full data set is transferred to cloud storage via the data pipeline, and once the storage has accumulated a particular quantity of data, the data will be cleaned and features will be generated for further model training. After that, the created features and models are saved in the model storage. To answer user requests, the inference module will use the trained models and features for online services. However, the timeliness of the data models produced using the typical training workflow is quite low, and achieving particularly high timeliness of updates is difficult. In addition, the entire training process will require a large amount of bandwidth to gather and send data. Furthermore, the entire training or feature creation procedure is time-consuming, and the processing power of the central cluster is highly demanding. Based on our observations, many companies continue to confront numerous bottlenecks and problems when implementing AI components into their embedded systems [13].

1.2 Challenges of the Typical Machine Learning Workflow

With substantial advancements in speech, picture, and text recognition, as well as language translation, machine learning has slowly revolutionized the way we live, learn, and work. Massive volumes of training data are collected from users by huge firms such as Google, Facebook, and Apple in order to develop large-scale deep learning networks. While the benefit of deep learning is obvious, the training data it employs can have significant privacy implications: photographs and videos of millions of people are collected centrally and stored indefinitely by huge corporations, and individuals have little control over how the data is utilized. Second, photos and videos are likely to contain sensitive information such as people's faces, registration plates, computer screens, and conversations. Large companies have a monopoly on "big data" and they could reap enormous economic gains as a result. [16][17][18]. Despite the fact that there are several obstacles connected with the machine learning field, based on our observations and literature [19][3][20][21], when companies attempt to implement machine learning components into practice, the majority of concerns focus on three areas: data privacy, efficiency, and data ownership and storage.

Data Privacy

With the development of machine learning, AI techniques have been widely used in a variety of fields in order to benefit people's daily life. Currently, applications such as recommendation systems for advertisement (Amazon, Google), computer vision for image recognition and objective detection (Autonomous Driving), natural language processing (Apple Siri, Amazon Alexa) etc, largely rely on machine learning models. However, in order to obtain a high-quality model, numerous user data needs to be analyzed and consumed, which leads to several ethical problems, such as how to utilize users' data, how to protect those data away from non-permitted actions?

In current society, personal information leakage is very common. A lot of companies or application owners may extract a user's personal identity information, religion, hobbies, interests, family, work information in order to benefit their business [22]. Search engines collect search records to provide personalized search services. However, those data may also be used to track and monitor users' behaviour. Various apps get your address book and location information. Online shopping information helps business decision-making, but it can be used for fraud if it is obtained by criminals.

Overall, we all agree that data can be utilized to enhance our quality of life. However, the way to handle and protect users' privacy has become a large ethical problem for researchers and industries. In the early stage of the development of ML, society depended on virtue and deontological ethics to regulate people for the usage of private data [23][24]. They believed that programmers and software service providers have the obligation to protect user information and prevent data abuse. In fact, reality proves that we cannot only rely on virtues or moral character. With the increasing attention on user privacy, concrete laws and policies should be introduced to restrict the usage of the user data, such as the General Data Protection Regulation (GDPR) [25]. It is well understood that as the amount of training data increases, so will the diversity and performance of machine learning models. However, regulations such as the general data protection regulation (GDPR) [25] prohibit the sharing of personal data in many industries. This legislation has established specific standards for privacy measures, which has improved the protection of personal information even more. As a result, researchers in similar businesses can only evaluate and mine data sets owned by their own companies. If a single organization (for example, a specific medical clinic) does not have a significant amount of data and has insufficient diversity, researchers may end up with a less generic model by executing machine learning on such a dataset [26]. The limits of data privacy and confidentiality definitely affect the performance of machine learning in this scenario [27]. Moreover, with the introduction of policies, more and more techniques of privacy protection will also be developed such as anonymous processing, encryption algorithm, etc. The concrete policy will introduce a better privacy-preserving strategy. Better privacy-preserving techniques may lead to more consideration about forming more efficient policies and eventually form a sustainable socio-technical system.

Communication Efficiency

On the other hand, with billions of edge devices globally networked, these endpoints can generate massive amounts of data. These data must be centrally transmitted to a cloud infrastructure for processing in traditional cloud computing designs. Machine learning's utility is largely drawn from the massive amounts of data acquired, and its effectiveness is further increased by the rising diversity of data. However, the growing number of connected devices constantly generating data from the network makes the companies more difficult to keep track of all the data coming in from various sources [28]. This old data collection and model distribution method may increase network traffic, resulting in transmission congestion and data processing delays. As the amount of data grew rapidly, a corporation using the centralized learning model needed to spend more and more money to create a new data center to store the explosive data.

Data Ownership and Storage

Furthermore, the utilization of machine learning with big data is quickly expanding. However, effective data control is still a significant problem and the way to design procedures to verify data quality, and lead the appropriate data application to produce value for the institution and society is still need to be investigated. Property rights (ownership) and data access have also long been a practical concern in big data applications [29][30]. Only by clearly defining property and access rights can data owners preserve data quality and minimize organizational and institutional barriers to data sharing. Many businesses today regard big data as their most valuable asset. However, data is frequently stored and processed in a distributed fashion, which increases the number of nodes and the amount of potential hostile assaults on the data. At this point, there is a lack of developed infrastructure and control mechanisms for data security protection. Data sharing and exchange are critical components of the Big Data sector. Because of a lack of data standards, it is difficult to share and use the many data sources that are locked up in information silos. Data security and privacy concerns are impediments to data sharing and exchange. Traditional data sharing solutions are being put to the test by the sharing of enormous data sources.

Overall, the operation of applications in the enterprise's digitisation generates massive amounts of data. As a result, enterprises are expanding beyond their traditional business intelligence and decision support system foundations to reapply big data generated in the user network to system development, thereby helping to improve the performance of their applications.

1.3 Transition towards Federated Learning

The current trend in artificial intelligence has shifted the emphasis away from AI-based algorithms but toward big data infrastructures that provide security and privacy. This transition in focus has resulted in the development of Federated Learning, which provides users with benefits that traditional machine learning does not [31].

Because of the issues described above, traditional data exchange solutions are unable to satisfy demand. Federated Learning is a novel technology that is becoming more advanced, based on a combination of secure multi-party computing and other cryptographic techniques. It is, in reality, a distributed machine learning technique in which participants can collaborate to create models without revealing the underlying data [32][33]. The emergence of this learning method alters the situation in which data must be centrally uploaded to the cloud for processing and maximizes the utilization of edge devices' computing and storage capabilities. The learning approach eliminates the need to upload large amounts of local data and reduces data transfer latency with local data processing. The advantages of Federated Learning can be summarized as follows:

1) Data is not leaked to the outside world in Federated Learning, addressing users' privacy and data security demands.

2) Federated learning enables the encrypted transmission of information and model parameters while maintaining the independence of all parties involved.

3) It ensures that the system's quality is preserved, negative migration is avoided and federated models outperform fragmented standalone models.

4) It ensures that the status of the parties involved is equal, supporting fair cooperation.

5) Federated learning can eliminate engineering obstacles, avoiding engineering issues such as enormous numbers of user data, costly network connections, sluggish transmission rates, and low transmission security.

5) The capacity to ensure that participating parties transmit encrypted information and model parameters while retaining independence and concurrent growth.

Data isolation and data privacy protection are becoming the next challenges in AI, but federated learning gives new hope. As technology spreads and standards improve, it will tear down boundaries between industries and establish a community where data characteristics and expertise can be securely exchanged, allowing everyone engaged to equitably share the benefits.

As a result, Federated Learning is appropriate primarily for jobs where the training data is sensitive to privacy or is too large to be collected centrally, which has raised the interest of industry participants who wish to benefit from this technique. However, despite its benefits, Federated Learning is still in its early stage in the domain of embedded systems [34]. This type of learning technique may also pose problems such as component failures, inefficient communication, unstable model performance, etc that needs further research when applied in a real-world context.

1.4 Summary of the Included Papers

In Chapter 4 (Paper A: "Engineering Federated Learning Systems: A Literature Review"), we provide a state-of-the-art overview of the empirical results reported in the existing literature regarding Federated Learning. According to the problems they expressed and solved, we then categorize those deployments into different application domains, identify their challenges and then propose six open research questions.

In Chapter 5 (Paper B: "Towards Federated Learning: A Case Study in the Telecommunication Domain"), we perform an interview-based case study to discover the obstacles that industries face while dealing with machine learning problems and why they believe Federated Learning is a viable solution. We collect the insights gathered from our experienced participants and established a strategy for assisting companies in the embedded system areas in their transition to Federated Learning. In addition, we summarize the services that a complete Federated Learning system needs to support in industrial scenarios and then identify the key challenges for industries to adopt and transition to Federated Learning. Finally, based on our empirical findings, we suggest five criteria for companies implementing reliable Federated Learning systems.

In Chapter 6 (Paper C: "Federated learning systems: Architecture alternatives"), we investigate and compare four architecture alternatives for a Federated Learning system, i.e. centralized, hierarchical, regional and decentralized architectures. We investigate the trade-off between communication latency, model evolution time and model classification performance, which is crucial to applying the results to real-world industrial systems.

In Chapter 7 (Paper D: "Real-time end-to-end federated learning: An automotive case study"), we introduce approaches and asynchronous frameworks to end-to-end on-device Machine Learning by utilizing Federated Learning. We validate our approach with important industrial use cases in the field of autonomous driving vehicles, the wheel steering angle prediction.

In Chapter 8 (Paper E: "AF-DNDF: Asynchronous Federated Learning of Deep Neural Decision Forests"), we combine asynchronous Federated Learning aggregation protocol with an advanced machine learning method, the Deep Neural Decision Forest, and evaluate our approach using the road objective recognition problems. The results show Federated Learning can accelerate model training speed and reduce the communication overhead, which proves that this approach has great strength when deploying ML/DL components to various real-world embedded systems.

1.5 Contribution of the Research

The contribution of this thesis is manifold. Firstly, it identifies the limitations of classic machine learning workflow, including the difficulties involved with current learning steps and the challenges that companies experienced (data collection, model training, model distribution, etc.) while attempting to use machine learning to improve service quality. Secondly, it answers the question of how Federated Learning can influence the business and improve the service quality in embedded systems. It provides the reason that industries consider Federated Learning as a possible solution and future learning method. Third, it identifies the main challenges of Federated Learning systems when companies implement them in a real-world context. It listed the challenges that industries are attempting to solve when adopting and transitioning to Federated Learning, summarizes the services and criteria required for a reliable Federated Learning system and the open research questions for Federated Learning. Fourth, it proposes a solution that can help companies to implement Federated Learning components. It introduces four architecture alternatives that have been or can be applied to a Federated Learning system and introduces a real-time end-to-end Federated Learning method for training Machine Learning models in a distributed context. The method also incorporates the asynchronous Federated aggregation protocol and the concept of Deep Neural Decision Forests.

1.6 Structure of the thesis

The following is the structure of this thesis. Chapter 1 explains machine learning, Federated Learning, and its important contributions. The background of this thesis is presented in Chapter 2. Chapter 3 explains the research methods, as well as the research goals and motivations for each of the methods used in the research. The papers A through E contained in this thesis constitute the basis for Chapters 4, 5, 6, 7, and 8. Chapter 4 provides an overview of the existing engineering Federated Learning systems. Chapter 5 discusses the obstacles and restrictions that prohibit industries from using Federated Learning in real-world systems. Chapter 6 addressed various architecture alternatives for the Federated Learning system. Chapter 7 proposes a real-time method for processing data and performing asynchronous Federated Learning model training. Chapter 8 introduces the combination of the deep neural decision forest and the asynchronous Federated Learning technique. Chapter 9 concludes the thesis with a discussion of the main results, contributions and future work.

CHAPTER 2

Background

This thesis studies Federated Learning and the transition from the typical machine learning workflows to Federated Learning in the real-world context, especially in the domain of embedded systems. Therefore, in order to provide the reader with the necessary information that is required to better understand this thesis, this section provides the background information and describe the related work of this thesis. Section 2.1 discusses the concept of machine learning, as well as the three major learning methods, supervised, unsupervised, and reinforcement learning. Section 2.2 introduces the concept of deep learning and how it affects today's intelligent systems. Section 2.3 introduces the concept of embedded systems as well as the current stage of AI component implementation in the domain of embedded systems. Section 2.4 discussed the impact of AI-powered applications and why they are important in today's businesses. Section 2.5 describes the concept of Federated Learning, as well as related works, opportunities, and challenges. At last, section 2.6 summarizes the chapter.

2.1 Machine Learning

Machine learning is a sub-field of artificial intelligence that focuses on allowing machines to learn from past experiences, model uncertainty in data, and make future predictions [8][35][36]. Assume you want to use a computer to automatically split animal photographs into two categories: cats or dogs. To begin, a training sample, in this case, a collection of photos of cats or dogs, must be acquired. Following that, the samples must be studied, and the characteristics must be identified by some abstraction of the descriptions. The data is then modelled in the third stage. Mathematical models are typically necessary to examine the distribution of attributes and labels. The goal is to use an optimization strategy to learn a mapping function, whose input is a vector of sample features and output is the labels (cats or dogs). When the user enters a new sample of animals in the prediction stage, we do the same feature extraction using the mapping function learned to anticipate. The mapping function that was built after the learning process is used to automatically predict the label category of the sample [37][38]. The above example is only one of the common supervised binary classification problems. In the field of machine learning, as Figure 2.1 shows, it is customary to classify algorithms into 3 categories [39]: unsupervised learning, supervised learning, and reinforcement learning. The difference between each method is that they learn from different types of data.

Supervised Learning

Supervised Learning is a function learned from a particular training dataset [40][41]. And the results can be predicted based on this function when new data arrives. Inputs and outputs, often known as features and targets, are part of the training set requirements for supervised learning. The target in the training set is labelled by a human.

Supervised learning is often used to solve the classification/regression problem, in which existing training samples (i.e. known data and their corresponding outputs) are trained to obtain an optimal model[42]. This model belongs to a certain set of functions and optimal means that it is the best under a certain evaluation criterion. Then this model is used to map all the inputs to the corresponding outputs, and to make the judgements on the outputs in order to achieve the classification/regression. This enables the classification

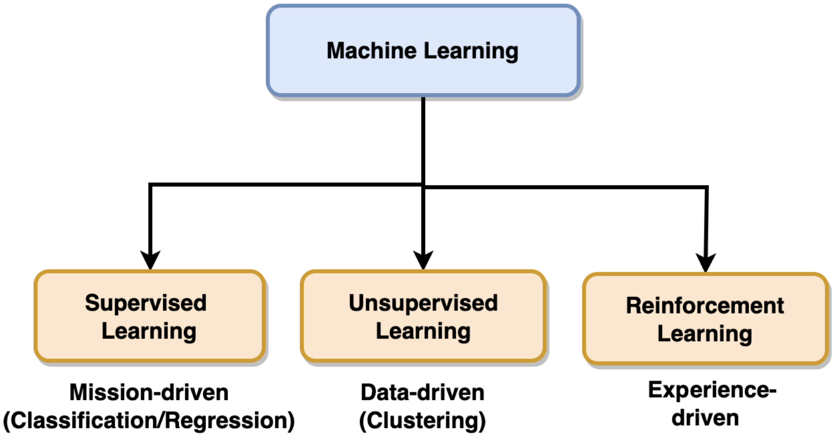


Figure 2.1: Categories of machine learning

of unknown data.

Supervised learning is a typical training method for neural networks and decision trees [43]. Both strategies rely primarily on data provided by a predefined system. The method for neural networks analyzes the information to evaluate network defects and then continuously modifies the network parameters. The system employs decision trees to identify which attributes provide the most information.

Unsupervised Learning

For unsupervised learning, the input data is unlabeled, and there is no clear outcome. The sample data is of an unknown category, hence it is important to classify the sample set based on sample similarity (clustering) in order to minimize the intra-class gap and maximize the inter-class gap [44][45]. This means that in many circumstances, the labels of the samples are unknown in advance, i.e. there is no category for the training examples, and the classifier design must be learned from the original set of unlabeled samples. For example, in a player experience classification problem, game players can be classified based on their level of involvement, and we can allow high engagement players to test the new gameplay while sending help tutorials to low engagement users. We begin by configuring the fundamental attributes: player time, player expenditure, and player level. These three attributes are fed into the Unsupervised Learning model method, which is pre-set and divided into two groups, and the system classifies all players into two groups: highly engaged players and lightly engaged players.

We do not specify specific situations or labels in the unsupervised learning model, such as which players should be highly engaged and which should not. Instead, we define the necessary attributes and offer the attributes to the algorithm, allowing the program to identify the groups naturally. Unsupervised learning is appropriate for situations in which labels are difficult to determine [46].

Reinforcement Learning

The purpose of reinforcement learning is not to direct the computer on what to do, but to allow it to learn on its own. The method is to teach the Agent without explicitly assigning it a classification but to employ some sort of incentive system when it succeeds [47][48]. It is important to highlight that this form of training is typically implemented within the context of a decision-making problem, as the goal is not to create a classification system but to make the most rewarding decision possible. Figure 2.2 gives a basic diagram of reinforcement learning.

This type of reasoning generalizes well to the actual world, where the agent can be rewarded for good behaviour and penalized for bad. Reinforcement learning poses a unique challenge: the trade-off between "trial" and "exploitation", where the intelligence must exploit existing experience to reap the benefits, while at the same time conducting trials that allow for better action choices in the future (i.e. learning from mistakes).

Model Training and Inference

Each of the three learning methods necessitates two phases: training and real-world application (inference phase, validation phase) [49][37]. The two phases of the three learning models differ in many ways: the training phase entails training a policy model from the available data, whereas the real-world phase entails adapting the developed policy model to a new environment.

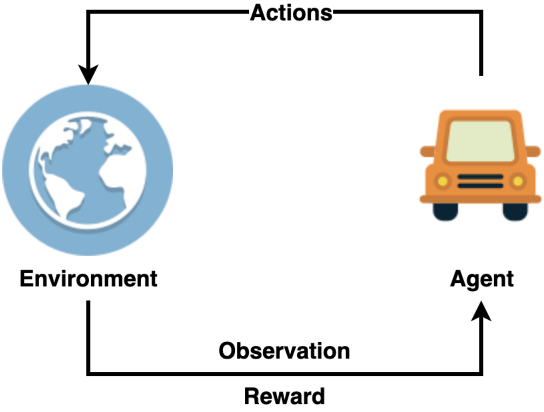


Figure 2.2: Diagram of the reinforcement learning

Specifically:

- Unsupervised Learning: the training phase must divide all samples into target groups, and the real-world phase can reuse the knowledge to divide new samples into corresponding groups [45].
- Supervised Learning: the training phase associates sample attributes with a given label, and the practice phase uses this association to predict whether a sample would fit the given label [40].
- Reinforcement Learning: the training phase achieves an optimized learning strategy through guided experiments, and the practice phase, in which agents employ the acquired approach to observe and complete the task in the new world [47].

2.2 Deep Learning

Humans learn through experience. The more experience you have, the more you will be able to learn. This approach is analogous to the subject of deep learning in the study of artificial intelligence (AI), in that computers powered by AI hardware and software learn from experience. The data from which the machine learns defines the amount of knowledge it can learn, and the quantity and quality of the data determine the amount of information it can learn [50][51]. Deep learning is a subset of machine learning. Unlike many typical machine learning algorithms, which have limited learning capacity and can not consistently increase the overall quantity of knowledge learnt, deep learning systems can enhance performance by accessing more data, which is a machine proxy for "more experienced". Deep learning allows a computer to accumulate enough expertise to be employed for specialized activities such as driving a car, diagnosing a disease, detecting faults, and so on [52].

Deep learning networks learn by detecting complex structures in empirical data. Deep learning networks can represent data at numerous levels of abstraction by constructing computational models with several processing layers [50][51]. For example, a convolutional neural network needs to be trained using a huge amount of image samples. Typically, this type of neural network learns from the pixels in the captured image. It can classify the pixels that represent the physical features of the target in the image and group the pixels that represent these physical traits [53].

Deep learning is substantially different from conventional machine learning. In this case, it would take a significant amount of time for a domain expert to develop a standard machine learning system to detect the physical features. Deep learning, on the other hand, requires only a huge number of samples to be provided to the system before it can learn to create the features. Deep learning systems can beat typical machine learning systems by a considerable margin in various tasks [51]. This is not to claim that developing a deep learning system is simpler than developing a typical machine learning system. Although deep learning performs feature identification independently, we still need to tune hundreds of hyper-parameters to assure the success of deep learning models.

2.3 Embedded Systems

Embedded systems are currently widely used in a variety of applications, including consumer electronics, smart hardware, communication devices, cars, medical devices, personal computers, and mobile phones. An embedded system is a real-time microcomputer system that is part of a machine or equipment [54]. Embedded systems are often compact, low-cost, and energy-efficient. The operation of an embedded system necessitates the execution of processes, tasks, or threads in order to respond to external or internal inputs or to complete normal transactions[55]. These methods must produce accurate findings within specific time constraints. The field of embedded systems is the result of numerous disciplines coming together, including software engineering, operating systems, and electrical engineering. Embedded systems incorporate many concepts from various disciplines which can be improved by adapting, upgrading, and enhancing these concepts and technologies [54][56].

In the context of embedded systems, machine learning enables the use of such data in automated processes to generate more experienced predictions. Machine learning makes use of a huge quantity of historical data to allow the systems to learn independently and apply that information for prediction and decision making [57]. These kinds of devices can help with a variety of jobs in the industries and enable machine learning algorithms to run on low-cost, low-power devices [58].

There are several advantages of using machine learning in embedded devices. It removes the need for data transfer and storage on cloud servers, which decreases data breaches and privacy leaks associated with data transfer. It also decreases intellectual property, personal data, and corporate secrets theft. The use of ML models reduces the need to upload data to a cloud server, saving bandwidth and network resources, which outperform cloud-based systems in terms of efficiency. This is due to the fact that there is no need to transfer a large amount of data to the cloud which contributes to significant network latency [59][60].

2.4 AI-Powered Applications

There's no denying that AI and machine learning are having a significant impact on the tech industry. From corporations to small workshops, AI and ML are everywhere, and possibly more than in our daily lives. AI is having a greater impact on the commercial sector. Machine learning/ Artificial Intelligence has been heavily invested in by tech giants like Facebook and Google, and the technology is already being used in their products. More businesses are looking to incorporate AI components into their products to improve them [61].

As artificial intelligence and machine learning promise to progress in their respective industries in the embedded systems domain, more organizations are switching to systems that support them [62]. Because machine learning and artificial intelligence have the power to transform present work inertia, these changes are bound to occur in a variety of domains [63]. AI has shortened shipping times and reduced shop backlogs in the retail sector. According to current trends, machine learning will be employed in the future to boost productivity and improve staff deficiencies without the need for mass layoffs [64]. Advertising is likewise undergoing a change as a result of artificial intelligence. Marketers will be able to obtain a better understanding of their clients' thoughts and hearts with the help of machine learning, and they will be able to target them with customised communications.

AI-enabled hardware and ML-enabled CPUs are anticipated to be found in mobile phone microchips [65]. These high-performance procedures will give users capabilities such as quick translation and more efficient speech recognition. Computers will also reduce in size as computing and storage capacity improve, but they will become exponentially more powerful [18]. These AI and machine learning-enabled devices will also assist organizations in being more accessible and globally competitive, as well as increasing revenue streams for various types of companies.

However, even though the concept of the implementation of machine learning components has significant benefits, it is sometimes hard for industries and companies to build a reliable and applicable system. We found that the transition from prototype to the production-quality deployment of ML models proves to be challenging for many companies [66].

However, while the concept of combining machine learning approaches is efficient, there are numerous challenges that businesses face as they strive to develop their digital intelligence [67][16][68]. The necessity of data for algorithms is well understood. Without data, even the most advanced algorithm is useless, and while there are a variety of open-source datasets accessible, they simply do not match the needs of an increasingly complex and diverse spectrum of machine learning applications [10]. Access to data and privacy controls also issues for enterprise-level machine learning applications, which can take weeks or months to get the right dataset. The majority of business data is extremely sensitive, particularly when dealing with the government, healthcare, and financial industries [69][60]. When it comes to exchanging data assets, non-disclosure agreements (NDAs) are highly rigorous. Furthermore, it is not uncommon for data to be dispersed throughout an organization, making it extremely difficult to obtain. When the amount of data produced grows at an exponential rate, the question of how to use it wisely and effectively, as well as how to store it, becomes a major concern [70].

2.5 Federated Learning

Machine Learning has piqued the interest of both researchers and the general public. The main challenge is that, while computation capability grows over time, the computational needs of many Machine Learning systems grow even faster [71]. For example, in order to attain acceptable model performance when using deep neural networks, the network must have millions or even billions of neurons, which may result in a longer training time and less model flexibility [42].

A basic diagram of a Federated Learning system is shown in 2.3. Local model training is used in this approach, and data generated by edge devices do not need to be exchanged. Weight updates are instead forwarded to a central aggregation server, which generates a global model. The solution addresses the issue that standard Machine Learning approaches can only train and deploy models on a single central server.

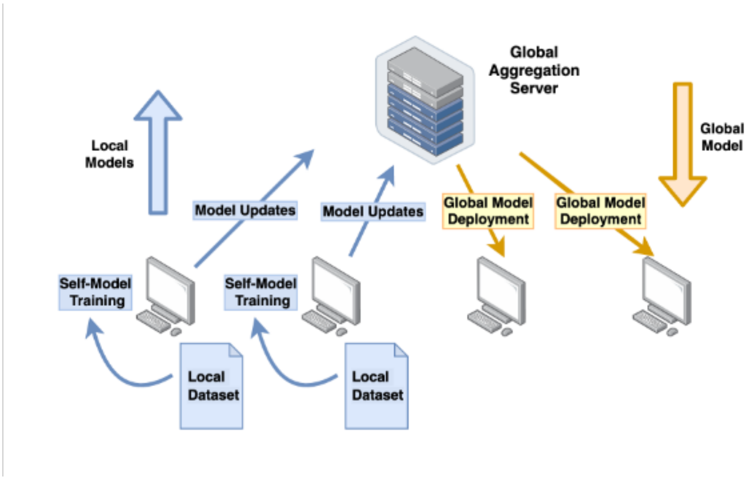


Figure 2.3: A basic diagram of a Federated Learning system

Since Google introduced the concept in 2017 [72], various Federated Learning architectures, frameworks, and solutions have been presented to address real-world concerns [31]. Because models are trained locally, no edge data is transferred in a Federated Learning system, and only weight updates are sent to a central aggregation server to construct a global model. Furthermore, with local training and validation, a Machine Learning model can be quickly verified and deployed, making it more appropriate for a rapidly growing system.

AI engineering has the opportunity to grow to a distributed setting with the concept of cloud computing and decentralized data storage. Federated Learning is proposed to improve existing Machine Learning methodologies by allowing edge devices to train a shared Machine Learning model collectively. Federated Learning theory has been investigated in [33][73]. Its primary goal is to learn a global statistical model from a large number of edge devices. The goal is specifically to minimize the following finite-sum objective function 2.1:

$$\min_{w} f(w), \ \ where \ f(w) := \sum_{i=1}^{n} \lambda_i f_i(w) \tag{2.1}$$

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 ith device. λ_i ($\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$) gives the impact of ith remote device and is defined by users.

In conclusion, the benefit of using Federated Learning is obvious. A Federated Learning system is a privacy-preserving Machine Learning approach due to the process of model training and data dissemination. It can use local computation resources to relieve the central server's computation load. Furthermore, because of the local training manner, the system can give rapid model deployment and development.

Despite its benefits, Federated Learning is still in its early stage in the domain of embedded systems [34]. This type of learning technique may also pose problems when applied in a real-world context. Because of the frequent model interchange, model aggregators and edge devices both require a dependable network connection. A centralized aggregation server may also result in a single point of failure or bottleneck. Different types of learning architectures must be thoroughly evaluated and designed for various industrial contexts. More demand for local computing capacity may need increased investment in hardware development and maintenance by companies. Furthermore, with unevenly distributed data, the global model may fail to converge, resulting in poor model performance. As a result, the method merits further investigation and development by researchers.

2.6 Summary

This chapter presents the background information required for a better understanding of this thesis. The chapter introduces the notion of machine learning as well as the three major learning methods: supervised, unsupervised, and reinforcement learning. Furthermore, it introduces the concept of deep learning and its impact on today's intelligent systems. The discussion of the impact of AI-powered applications, the concept of embedded systems, and the current stage of AI component implementation in the domain of embedded systems demonstrates the significance and provides context for the implementation environment that will be applied in Chapters 4-8. Finally, the overview of the concept, opportunities, and challenges of Federated Learning provides the fundamental knowledge of the topic discussed in this thesis, which is relevant in the subsequent chapters.

CHAPTER 3

Research Methodology and Design

The tactics, processes, or techniques used in the collection of data or evidence for analysis in order to reveal new knowledge or generate a better understanding of a topic are referred to as research methodologies. In this thesis, we explore and discuss the importance of the transitions from commonly used machine learning methods to Federated Learning methods and the way to help the Federated Learning components to be implemented in industrial cases. In this chapter, we present the methodology applied during this research.

3.1 Research Questions

The goal of this research is to identify the barriers and limits that exist in the engineering Federated Learning systems. And we hope to assist industries in overcoming the barriers that impede many industrial organizations from adopting Federated Learning. Despite the fact that we regard Federated Learning as having the ability to deliver solutions to a variety of use cases, and that research and implementation have demonstrated its usefulness, this technique is still in its infancy. To fulfill the research objectives, we adopted an empirical research approach with design science being our primary research method. The study is primarily concerned with the following research questions:

- RQ1. What are the main challenges with existing Machine Learning workflows and how can Federated Learning help in addressing and solving these challenges?
- RQ2. What are the primary constraints and limitations of current Federated Learning systems?
- RQ3. What are the solutions that can help companies in the embedded systems domain build Federated Learning in practice?

The first research question (RQ1) is to explore and gain an initial understanding of Federated Learning. Because the approach is still in its early stages and requires further development, it is critical to understand from an industrial standpoint what sorts of qualities are beneficial for businesses, what kind of challenges they faced and how Federated Learning can assist them in resolving their problems. To address RQ1, we conducted a case study that interviewed experienced machine learning experts, data experts, project managers to gain knowledge of the potential applications of Federated Learning. We also performed a literature review to examine previous research and the findings that these present. Each study paper's topic, methodology, and deployments were taken into account. We identified the specific technical problem addressed in the study based on empirical data and solutions. The second research question (RQ2) is intended to explore the constraints and limitations of existing Federated Learning systems, as well as the obstacles that prohibit organizations from incorporating Federated Learning components into embedded systems. Regarding RQ2, throughout the literature review, we analyzed the improvements and contributions of each work and organized the interview to conclude the problems and limitations of existing Federated Learning systems. The findings indicate numerous research avenues for implementing trustworthy Federated Learning systems in the real world. The third research question (RQ3) is intended to develop solutions including algorithms, frameworks and concepts, as well as to apply and validate our proposed solutions in a real-world scenario. We collaborate with companies in the domain of embedded systems to investigate how to put the Federated Learning theory into practice and realize it in real-world circumstances.

3.2 Empirical Research

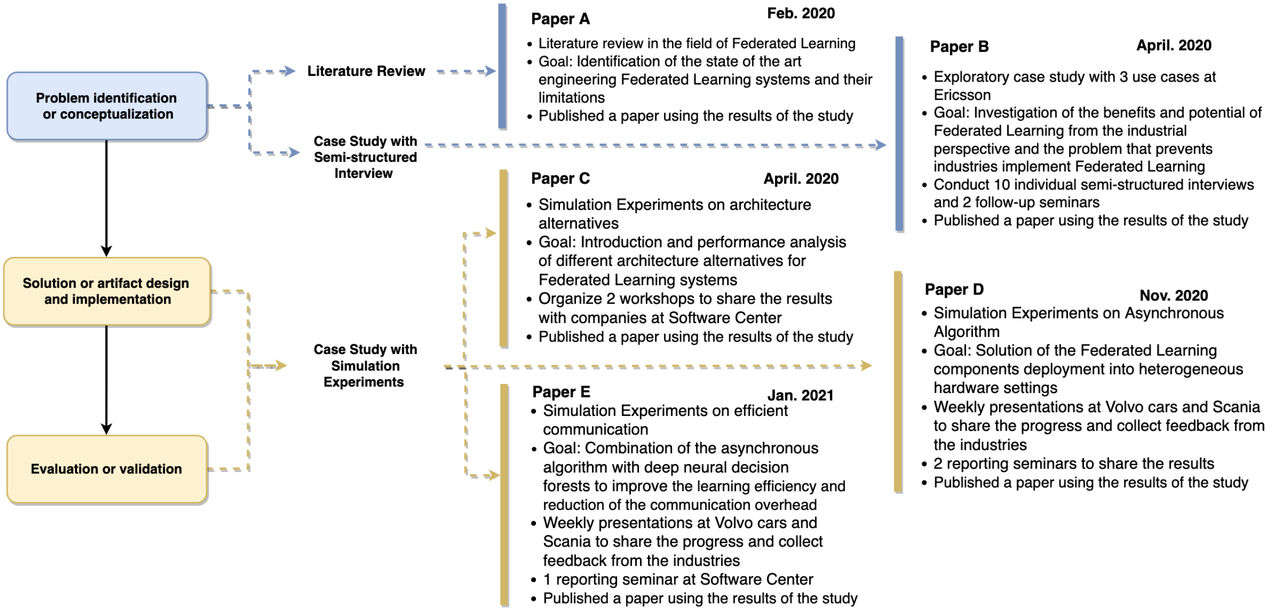
Empirical research [74][75] is research that is based on empirical evidence. It is also an approach to acquiring knowledge through direct and indirect observation or experience. The record of one's actual observations or experiences, known as empirical evidence, can be examined numerically or qualitatively [76]. A researcher can answer empirical questions that are clearly defined and answerable with the evidence obtained by quantifying the evidence or making meaning of it in qualitative or quantitative data. [74].

The approach helps in the improvement, analysis, and evaluation of software development methods and processes. It also serves as a guide for making decisions [75]. In several instances, empirical research is beneficial to the software industry and software engineering researchers. The empirical study may be used to answer inquiries about industry practices and to enhance software development strategies and processes [77]. Empirical research enables software engineers to utilize experiment results and ensure that a set of appropriate procedures and processes are followed at some stage during software development [78]. As a consequence, empirical approaches assist in identifying the best of the resulting software processes and products.

In summary, the purpose of this study was to investigate empirical cases at companies in the domain of embedded systems, investigate the obstacles and constraints they encountered when implementing Federated Learning, and propose corresponding solutions. As a result, empirical research is a useful approach for the phenomenon of this study.

3.3 Design Science

The scientific approach should be used to study design as a science. The scientific method consists of theory and doctrine, the development of experimentally testable conjectures based on theory and doctrine, the systematic analysis of experimentally observed phenomena, and the evaluation of the validity of theories based on observational results [79] [80]. It aims to develop innovations that describe the concepts, methods, technological capabilities, and products that can be used to effectively and efficiently analyze, design, implement and manage the information systems [81][80]. Science is one that has tested theories, research that focuses on narrow and constrained models,


Problem identification or conceptualization
Solution or artifact design and implementation
Evaluation or validation
Literature Review
Case Study with Semi-structured Interview
Case Study with Simulation Experiments

Paper A Feb. 2020

- Literature review in the field of Federated Learning
- Goal: Identification of the state of the art engineering Federated Learning systems and their limitations
- Published a paper using the results of the study

Paper B April. 2020

- Exploratory case study with 3 use cases at Ericsson
- Goal: Investigation of the benefits and potential of Federated Learning from the industrial perspective and the problem that prevents industries implement Federated Learning
- Conduct 10 individual semi-structured interviews and 2 follow-up seminars
- Published a paper using the results of the study

Paper C April. 2020

- Simulation Experiments on architecture alternatives
- Goal: Introduction and performance analysis of different architecture alternatives for Federated Learning systems
- Organize 2 workshops to share the results with companies at Software Center
- Published a paper using the results of the study

Paper D Nov. 2020

- Simulation Experiments on Asynchronous Algorithm
- Goal: Solution of the Federated Learning components deployment into heterogeneous hardware settings
- Weekly presentations at Volvo cars and Scania to share the progress and collect feedback from the industries
- 2 reporting seminars to share the results
- Published a paper using the results of the study

Paper E Jan. 2021

- Simulation Experiments on efficient communication
- Goal: Combination of the asynchronous algorithm with deep neural decision forests to improve the learning efficiency and reduction of the communication overhead
- Weekly presentations at Volvo cars and Scania to share the progress and collect feedback from the industries
- 1 reporting seminar at Software Center
- Published a paper using the results of the study

Figure 3.1: Overview of the research activities

and models that can give scientifically tested predictions [82]. In short, design science should consist of operational theories and models that explain observed events and can anticipate what will occur. Rigour in design science research results from the use of a knowledge base - a theoretical foundation and a research methodology [80]. Success is founded on the researcher's ability to select not just the appropriate approaches for building or developing a theory, but also the appropriate ways for proving that theory or evaluating that output [83]. Design is fundamentally the process of searching for an effective solution to a problem. A solution can be defined as the realization of the desired end by possible means while adhering to the laws of the environment [80]. Design science is inherently a problem-solving process. The fundamental principle of design-science research is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artefact. That is, design-science research requires the creation of an innovative, purposeful artefact for a specified problem domain. Because the artefact is purposeful, it must yield utility for the specified problem. Hence, a thorough evaluation of the artefact is crucial. [80]. The design research involves six concepts [83], namely problem identification and motivation, the definition of the objectives for a solution, solution design, demonstration, evaluation, and communication. According to Wohlin et al.[84], the essence of Design Science is about problem-solving using an artefact as the means to improve a situation. All of these instances encapsulate three generic activities: 1) Problem identification or conceptualization 2) Solution or artefact design and implementation 3) Evaluation or validation

The main data collection and evaluation methods employed in this study are literature review, and case studies combined with semi-structured interviews and simulation experiments. The literature review and interviews were conducted to help the author understand the problem while simulation experiments are applied as a way to evaluate the artefact such as the algorithms and frameworks. Those methods can produce fruitful information which increases the depth of the understanding of the cases. Figure 3.1 shows the overview of the research activities based on design science methodology. In order to identify the problem, we firstly identified the research questions through literature review and semi-structured interview which guided us to further understand Federated Learning and the challenges of typical machine learning workflows in the embedded system domain, summarize the obstacle that prevents the transition of companies to Federated Learning and propose the solutions. Secondly, the processes for validating the topic were designed. In this step, different research techniques and data were considered to help the author answer the research questions and verify the research hypothesis. The third step is to analyze data based on the result collected from different research techniques applied. In this step, based on the simulations we conducted, we summarized the observation, validated the hypothesis, reflect the impact of proposed solutions and raise open questions for future research.

As the fundamental methodology of this research, design science research was used to help us better understand our problem and propose corresponding solutions. The method assists the author in seeing and comprehending the raised research questions, the challenges that the industry encountered, and then proposing the corresponding solutions and performing validation.

The contributions of each individual paper can address one or more of these activities we mentioned above. Through a literature review and case study combined with a semi-structured interview, Paper A, B identified the problems of Federated learning, form the motivation and define the objectives of this research. Paper C, D, E propose the algorithms, frameworks as the solutions to the questions which has been raised by the previous activities and validated them on the empirical datasets through case studies combined with simulation experiments. In addition, in close coordination with the companies from Software Center, several seminars and meetings were set up to help us better understand the difficulties that companies face, exchange progress, and gather input from practitioners with extensive working experience. Following the presentation of the results, discussions on the effect of the new algorithms and frameworks were held to further guide and improve our study.

In summary, the goal of this study was to explore cases at companies in the domain of embedded systems, investigate the challenges and constraints they encountered when adopting Federated Learning methods, and offer an improved practice. Since the purpose of our research is to identify problems and solve them using an artefact as a means to enhance a situation, the design science methodology is appropriate for this study.

3.4 Data Collection and Evaluation Methods

The data collection and evaluation methods are used to help gather empirical data necessary to analyze the actions in the real-world context [85]. The methods applied in this thesis such as literature review, case studies are appropriate for practical problems which can provide an extensive understanding of the behaviour and the influence of the choices for the researchers.

Literature Review

A literature review, according to [86][87], 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. [88]. 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 insufficient and highlight points of focus and innovation in the researcher's own research on the topic. [89]. In our study, we use a literature review to better understand the concept of Federated Learning and the limitations and constraints that exists 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 problem solved by using Federated Learning. We searched papers from various high-ranked journals/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.

Case Study

Case study research is a type of field study. The researcher selects one or more circumstances, collects data and information systematically, and conducts an in-depth study to explore a phenomenon's situation in a real-life context. [90][91]. The benefits of case studies in the software engineering field include complete data collection and analysis of quantitative and qualitative data in the context of the phenomenon and the capacity to catch the complexity of real-life scenarios in order to study the phenomena at a comprehensive level. It is appropriate when the boundaries between the phenomena and the actual environment are ambiguous and difficult to identify, or when the researcher is unable to design precise, direct, and systematically controlled variables to answer research questions, such as "How did it change?", "Why did it become so?", "what was the outcome?", "What is the result?", etc. It also includes a distinct design rationale, distinct data collection methodologies and data analysis methods [91]. To gather information, field observations of behaviour or research documents can be employed. The research is more qualitative in nature and includes distinguishing data gathering and analytic characteristics, such as dependence on numerous sources of evidence, where different sources of evidence must converge in a triangulated fashion to achieve the same conclusion. To guide the path of data collection and the focus of data analysis, there are frequently pre-developed theoretical propositions or problem descriptions. [92]. In contrast to other research approaches, the method provides a clear description and systematic understanding of the dynamic processes of interaction and the context in which they occur, in contrast to other research methodologies, allowing for a more complete and holistic view of a case [90]. We adopted this method combined with semi-structured individual interviews and simulation experiments since we wanted to identify the challenges of the current machine learning workflow in the embedded system domain, determine the benefits of Federated Learning, identify the primary constraints and limitations and propose the solutions for the Federated Learning systems in the real-world context. The exploratory case study allows us to gain a thorough knowledge of the complexities involved in implementing Federated Learning in the embedded systems area. The research techniques, namely interviews and simulation experiments, were described in the following sections.

1) Interviews: Interview-based data collection is a strategy in which the investigator utilizes specific instruments (e.g., interview forms) or assistance (e.g., tape recorders, emails) to ask respondents direct questions verbally, record the responses on the spot, and therefore learn about the real situation [93]. Interview research allows for more in-depth exploration. Questions may be more open-ended, and survey data can be more personalized. Furthermore, the interview approach is adaptable in that the researcher controls the study topic and procedure in real-time, and the information received from the interview is taken from the interviewees' words, allowing for more accurate findings in a specific interview context [94]. During the interview, the investigator can ask follow-up questions on a specific subject and notice the interviewee's words, actions, expressions, and other external appearances, which can be very useful in future study [95]. Interviews can be divided into unstructured, semi-structured and fully structured interviews [94]. Structured interviews have pre-planned and pre-created questions. The same questions are asked to all applicants in the same sequence. A semi-structured interview is one in which the interviewer asks only a few predefined questions and the rest of the questions are unplanned. An unstructured interview is one in which the interviewer asks questions that have not been pre-planned. Instead, in a free-flowing dialogue, questions develop spontaneously, which implies that various candidates are asked different questions. We used semi-structured interviews for our research because they are ideally suited for exploring participants' perspectives and opinions on the problems they experienced during their everyday machine learning work and their concerns about the transition to Federated Learning. Furthermore, the opportunity for face-to-face interaction with the interviewers stimulates the researchers' and interviewees' interests and discussions of the topics outlined in the project and helps to develop rapport between the researchers and interviewees [96].

2) Simulation Experiments: To study the causal relationship between variables using quantitative data, an experiment is em-

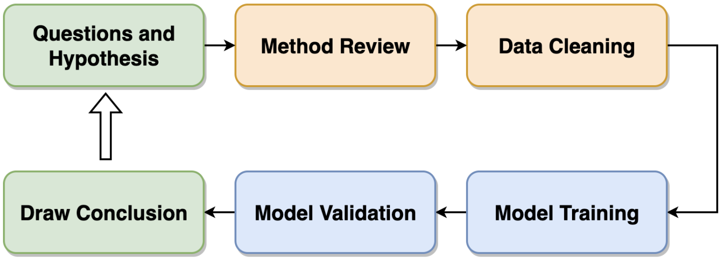


Figure 3.2: The simulation process (cycle)

ployed. In a controlled context, this necessitates careful separation from confusing effects [97][98][99]. For example, if researchers want to investigate whether a certain tool may increase the performance of the software in relation to a specific measure, they might create an experiment in which the system operates with or without the tool. From the formulation of the research idea to the practical implementation of the experiment is a fundamental part of the project. In this section, we present the simulation techniques (Figure 3.2) used in this thesis, which includes questions and hypothesis raising, methods review, model training (hyper-parameter seeking), model validation, and conclusion drafting [100].

1. Questions and Hypothesis: In this step, we present research questions and hypotheses to meet our study's purpose. The questions can be used to guide the technology selection and validation case design for the simulations.

2. Methods Review: After research questions are proposed and the data has been prepared, a method review is conducted in order to evaluate several commonly used machine learning algorithms/methods/frameworks based on those questions. A variety of existing models can be used for different purposes. These models are designed with different objectives in mind.

For example, some models are better suited to working with text, while another model may be better suited to working with images. Different architectures and aggregation protocols are also reviewed based on the case scenario.

3. Data Cleaning: In this part, some data cleaning strategies are used to remove low variance features, strings, constants and outliers. Another major component of data preparation is the division of the dataset. In most cases, the dataset will be firstly equally divided and distributed to the edge devices. In each local edge device, the larger part (approximately 70%) will be used for training the model, while the smaller part (approximately 30%) will be used for evaluation. This is important because using the same dataset for training and evaluation will not allow for a fair assessment of the model's performance in real-world scenarios.

4. Model Training: In this part, models are trained. In order to achieve applicable model quality for the edge clients, multiple hyper-parameters are searched and tried. The strategy used here is the random search. during the model training process, the quality of local edge models will be continuously improved by aggregating models with others.

5. Model Validation: After the model has been trained, it needs to be tested to see if it will work properly in a realistic environment. This is why a portion of the dataset created for evaluation is used to check the proficiency of the model. This will place the model in a scenario where the situations encountered are not part of its training. If the model and system performance fail to reach our expectations, new hyper-parameters are tried. In business applications, evaluation becomes very important. Evaluation allows data scientists to check that

they have set the goals to be achieved. If the results are unsatisfactory, previous steps need to be re-examined in order to identify and pinpoint the root cause of the model's poor performance. If the evaluation is not completed correctly, the model may not perform as well as it needs to for business purposes.

6. Draw Conclusion: The final process is to analyze the metrics and draw conclusions to answer the initially proposed research questions. The conclusion can help us to verify if the initial research goals have been achieved and generate the next steps for the future research direction.

We adopted this method since we wanted to propose new algorithms and frameworks in order to solve the challenges associated with Federated Learning encountered by the companies in the domain embedded system, as well as validated the proposed approaches on empirical scenarios and datasets. The method also allows us to gain a thorough practical knowledge of deploying Federated Learning in a real-world context.

3.5 Research Design

This section describes how we planned our research. The goal of this research is to identify the issues of Federated Learning systems as well as the obstacles that prohibit organizations from incorporating Federated Learning components into their embedded systems. As defined in the design science research methodology, we firstly identified the problems that companies encountered, suggested solutions based on the issues and then validated the algorithms/frameworks on empirical examples. As a result, the research is divided into two phases: Problem Identification and

Solution proposal and validation. Table 3.1 shows the overview of the research activities in these two phases.

Problem Identification

The first phase is to discover the ideas and identify the problems of the current Federated Learning systems. A literature study was first conducted to help the researcher get a sufficient understanding of what has been done in the past in order to learn more about the topic under examination. Throughout the literature review, we determined the first and second research questions, which provide us with a clear picture of what Federated Learning systems/applications are already available. We raise the questions, such as "What are the primary advantages of putting in place a Federated Learning system?", "What are the primary obstacles and limitations of present systems?", etc. To further investigate the barriers that prevent industrial implementation of Federated Learning components in real-world contexts, we chose Ericsson as our exploratory case company and an interview-based case study was conducted in which ten experienced engineers participated. Ericsson is now seeking a way to provide consistent service quality to its users due to its large-scale and distributed customers. We chose the company as our case company because it has broad experience and is investigating the prospect of applying Federated Learning. During the study, we identified the challenges that companies faced when deploying Federated Learning in an industrial context. We collected data primarily through semi-structured interviews with practitioners from Ericsson who design or use machine learning applications, who involve heavily in data engineering, or who are otherwise machine learning experts. Based on their work experience, they offered their opinions on the issues they have when dealing with standard machine learning workflows, as well

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	What are the main challenges with existing Machine Learning workflows and how can Federated Learning help in addressing and solving these challenges?	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
	What are the primary constraints and limitations of current Federated Learning systems?	To identify the limitations of the current Federated Learning implementations			
Solution Proposal and Validation	What are the solutions that can help companies in the embedded systems domain build Federated Learning in practice?	To propose solutions and validate algorithm or frameworks on empirical datasets	Case Studies with simulation experiments	Volvo Cars, Scania	Senior Data Scientist, Analytic System Architect, Data engineer, Software Developer, etc

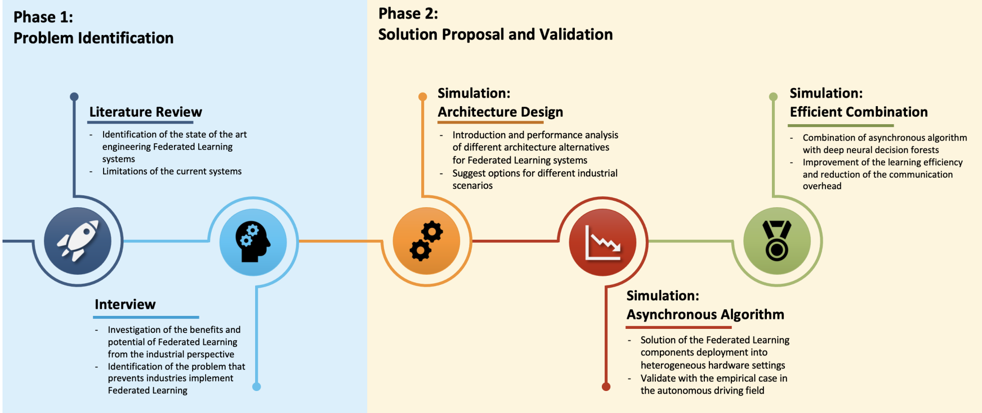

Phase 1:
Problem Identification
Literature Review
- Identification of the state of the art engineering Federated Learning systems
- Limitations of the current systems
Interview
- Investigation of the benefits and potential of Federated Learning from the industrial perspective
- Identification of the problem that prevents industries implement Federated Learning
Phase 2:
Solution Proposal and Validation
Simulation:
Architecture Design
- Introduction and performance analysis of different architecture alternatives for Federated Learning systems
- Suggest options for different industrial scenarios
Simulation:
Asynchronous Algorithm
- Solution of the Federated Learning components deployment into heterogeneous hardware settings
- Validate with the empirical case in the autonomous driving field
Simulation:
Efficient Combination
- Combination of asynchronous algorithm with deep neural decision forests
- Improvement of the learning efficiency and reduction of the communication overhead

Figure 3.3: Overview of the research design

as how they believe Federated Learning can be the solution to future intelligent industrial applications. Furthermore, we explored the potential and challenges of Federated Learning, as well as how organizations in the embedded system field may transition from traditional machine learning to Federated Learning.

Solution Proposal and Validation

Based on the conclusion and ideas gathered from the first phase, several solutions, analyses, and validations were carried out in the second phase to demonstrate how we can assist companies in building Federated Learning components. We examined the performance of several designs that can be utilized in various industrial applications based on the challenges mentioned in the literature review and case study. In this phase, we undertook research in close collaboration with Volvo Cars and Scania. During the collaboration, we constantly shared the most recent findings in a weekly workshop and collect comments from senior data scientists, AI system developers, and architects. We first analyzed the differ-

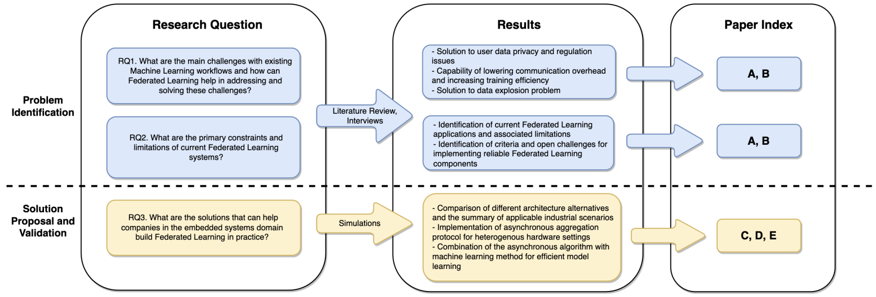

Research Question
Results
Paper Index
Problem Identification
RQ1. What are the main challenges with existing Machine Learning workflows and how can Federated Learning help in addressing and solving these challenges?
RQ2. What are the primary constraints and limitations of current Federated Learning systems?
Literature Review, Interviews
- Solution to user data privacy and regulation issues
- Capability of lowering communication overhead and increasing training efficiency
- Solution to data explosion problem
A, B
- Identification of current Federated Learning applications and associated limitations
- Identification of criteria and open challenges for implementing reliable Federated Learning components
A, B
Solution Proposal and Validation
RQ3. What are the solutions that can help companies in the embedded systems domain build Federated Learning in practice?
Simulations
- Comparison of different architecture alternatives and the summary of applicable industrial scenarios
- Implementation of asynchronous aggregation protocol for heterogenous hardware settings
- Combination of the asynchronous algorithm with machine learning method for efficient model learning
C, D, E

Figure 3.4: Overview of the research questions, methods and results

ent architectures that can be applied in Federated Learning. We summarized their performance and conclude the architecture alternatives that are suitable to be deployed in different industrial scenarios. We presented an asynchronous aggregation protocol to overcome the challenge of deploying Federated Learning systems in real-time instances and avoid the difficulties that arise when the system is comprised of diverse hardware settings. In addition, we studied the method to combine the asynchronous Federated Learning algorithm with several machine learning methods. Simultaneously, by refining the sharing method, we increased model learning efficiency and minimized communication overhead even further. All approaches were validated on empirical datasets, and we chose crucial use cases in the autonomous driving sector to demonstrate the usefulness and efficiency of our suggested strategy when applied in a real-world situation.

3.6 Industrial Collaboration

This research was conducted in close collaboration between academia and industries with the help of Software Center [101]. Software Center is a research collaboration consisting of 17 enterprises and

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 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 Machine 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 previous results during the company workshop	Feedback from the various teams in Volvo Cars, Scania	Participants confirmed the concept of Federated Learning, 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 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	Identification of the current results and the potential improvements for future research

5 colleges with the mission to significantly improve the digitalization capability of the European software-intensive industry. Those Embedded system companies are primarily looking for a promising way to incorporate Artificial Intelligence, particularly machine learning/deep learning, into their systems in order to expedite digitization and improve service quality. Various seminars and workshops were hosted by Software Center during each project sprint to enable researchers to get closer to the industry and develop a better knowledge of the challenges encountered by the companies in the embedded systems domain. In this study, several companies from Software Center took part in our research. We cooperated with Ericsson, Scania, Volvo, and other two companies to identify the difficulties they encountered when using machine learning methods and how Federated Learning could be a solution.

With the case study, we discovered how to deploy Federated Learning components in a real-world context, as well as the issues that engineers must consider. In the problem identification phase, with the assistance of Ericsson, we identified the challenges encountered by the companies and the limitations and constraints of Federated Learning. During the study, 10 semi-structured interviews were conducted to gather opinions from four experienced machine learning engineers, three data experts, two project managers, and one analytic architect. Throughout the interview, we outline the issues, solutions, and enhancements that must be implemented if Federated learning is to be used in a real-world situation. In addition, 8 follow-up meetings with Ericsson participants were scheduled to analyze the data collected and share thoughts about the Federated Learning solution. Finally, a reporting workshop was held in order to make a final presentation of the results. During the solution development and validation phase, we validated our proposed solution, algorithms, and frameworks with real-world automotive cases with the support of Volvo Cars and Scania. During the solution development process, in each project, 16 follow-up meetings were held with Volvo Cars and Scania participants, including six machine learning experts and two project managers. During the meetings, we analyze the simulation results and share the algorithm and frameworks for the Federated Learning solutions. Finally, a reporting workshop was held to make a final presentation of the data and identify current findings as well as recommended improvements for future studies. The details of the activities of case studies combined with interviews and simulation experiments during the collaboration with companies are listed in Table 3.2 and Table 3.3.

3.7 Threats to Validity

Construct validity

While the empirical results are subjective because they represent the experiences of the chosen persons, the probable breadth of the results and their applicability is expanded due to the professional participants' extensive experience. The conclusions reported in this thesis are mostly applicable to the domain and scenarios addressed in this study [102]. However, the ideas and outcomes may be applicable and important beyond the unique issue at hand. The authors and case companies and participants in this thesis have substantial experience with machine learning, Federated Learning, and data engineering. In addition, if structures with specialized terminology within industries or academia were not understood, the authors with experience in both could translate and demonstrate them. As a result, the presence of dangers in idea validity is not recognized.

Conclusion validity

Conclusion validity commonly refers to the particular reasons, methods and procedures we use to draw conclusions about a possible co-variation between variables [103], [104]. Several methods were implemented during the study to ensure that the conclusions' validity was not threatened. The first step was to remove prejudice created by persons who held similar beliefs and publications that reached similar findings. The second research phase involved gathering data in the form of documentation and simulations to supplement the information collected through interviews and literature research, which also served to reduce bias induced by merely collecting data in one format. Further, workshops and seminars were held by Software Center with participants and case companies where the findings were presented by the author and discussed with the participants. The workshop's purpose was to validate the findings ensuring nothing has been misinterpreted or missing. During the workshops, participants largely validated the findings.

External validity

This study was conducted in close collaboration with multiple companies, interviews were collected from participants of different teams, areas and the simulations were conducted on different industrial cases. All the terminologies utilized in the companies were normalized and the implementation was described with necessary details [105]. However, due to the focus on cases in the domain of embedded systems, we can not claim that our results generalize to the entire industry. However, the authors believe that there are many similarities between the case study to other companies in regulated fields.

3.8 Summary

This Chapter discusses three research problems addressed in this thesis. All research methods and techniques are presented. In the research design section, we present all of the research activities we carried out to demonstrate how we answered the questions. The Figure 3.3, 3.4 and Table 3.1 provide a summary of the research process, empirical efforts, and findings. In addition, Table 3.2 and Table 3.3 detail the activities of case studies combined with interviews and simulation experiments during the collaboration with companies. Finally, we discuss the threats to validity and how we attempted to minimize them.

CHAPTER 4

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 [42]. 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 [71], 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 [106]. 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 [33][73]. After the concept was first applied by Google in 2017 [72], 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 4.1, we describe the research method we applied. In section 4.2, we summarize the results from the literature review. In section 4.3, we outline the challenges of current Federated Learning systems. Finally, we conclude the paper in section 4.4.

4.1 Research Method

This research is conducted following the guidelines presented by Kitchenham [107]. 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?

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[108].

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.

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.

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 4.1, 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 ([109][110][111][112]) in telecommunication field, 6 papers ([72][113][114][115][116][117]) relates to mobile applications, 4 papers ([118][119][120][121]) relates to automotive, 5 papers ([122][123][124][125][126]) in IoT and 4 papers ([127][128][129][130]) relates to medical solutions. The rest of the papers ([131][132][133][134][135]) are related to other fields like air quality monitoring, image-based geolocation recognition, etc.

4.2 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.

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. [109] 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. [111] introduce reputation metrics for reliable worker selection in mobile networks. The solution enhances system safety while keeping the same prediction accuracy. Yang et al. [112] 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.

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. [72] and Ramaswamy et al. [113] apply Federated Learning techniques on the Google Keyboard platform to improve virtual keyboard search suggestion quality and emoji prediction. Leroy et al. [115] conduct an empirical study for the "Hey Snips" wake word spotting by applying Federated Learning techniques. Ammand et al. [116] implement a federated collaborative filter for personalized recommendation system. Liu et al. [117] propose “FedVision”, an online visual object detection platform, which is the first computer vision application applied Federated Learning technique.

Automotive

Automotive is a prospective domain for Federated Learning applications. Samarakoon et al. [118] 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. [119] and Saputra et al. [120] 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. [121] 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.

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. [122] propose a real-time data processing architecture of the Federated Learning system on top of differential IoT. Zhao et al. [123] 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. [125] design an advanced FedAvg algorithm which greatly reduces the number of rounds to model convergence in IoT network. Savazzi et al. [126] 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. [124] give a distributed video analytic architecture based on Federated Learning. It allows real-time distributed object detection and privacy-preserving scheme for model updating.

Medical

Federated Learning has propelled to the forefront in investigations of this application domain. Vepakomma et al. [127] propose “splitNN” which enables local and central health entities to collaborate without sharing patient labels. Huang et al. [128][130] present an approach of improving the efficiency of Federated Learning on health records prediction. Brisimi et al. [129] 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.

Other

In our research, we also identified some other application scenarios. Sozinov et al. [134] 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. [133] 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. [132] provide strategies and results in building AI models using the concept of federated AI across multiple agencies. Hu et al. [135] 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. [131] evaluate an efficient and privacy-enhanced Federated Learning scheme for industrial AI solution.

4.3 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 [109] 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 [112][125][131]. 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 [130][119][129], 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 [33] 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 [123].

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?

4.4 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 5

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 [42]. 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 [136]. 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 [137]. However, in many fields, the sharing of personal data is not allowed by regulations, such as GDPR [138]. 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 [31]. 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 5.1 presents the background of this study. In section 5.2, we describe the research method we applied as our basic principle when searching and collecting data. In section 5.3, we summarize the results from the interviews. In section 5.4, we outline the challenges and the criteria when realizing Federated Learning components into industrial systems. Finally, we conclude the paper in section 5.5.

5.1 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 [33][73]. Its major objective is to learn a global statistical model from numerous edge devices.

With the concept first applied by Google in 2017 [72], 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 [139].

5.2 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. [140]. 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 [141], [142]. 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 [143] and where there is a desire to access the interpretations and expectations of people so that a particular context can be richly understood [140]. 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.

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 [91]. 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 [90].

The obtained data were processed using inductive thematic coding technique [144][142]. 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.

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.

Use Cases and Participants

During the research, we studied three different use cases within Ericsson. As we listed in Table 5.1, 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 [28]. 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 5.1.

Table 5.1: 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

Threats to validity

The findings, like any case study, is primarily applicable to the domain and situations that were studied in this research [102]. 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.

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.

5.3 Empirical findings: Towards Federated Learning

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

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

5.4 Discussion

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.

Learning Services

When conducting Federated Learning in actual production environment, we must consider not only the coupling and stability of

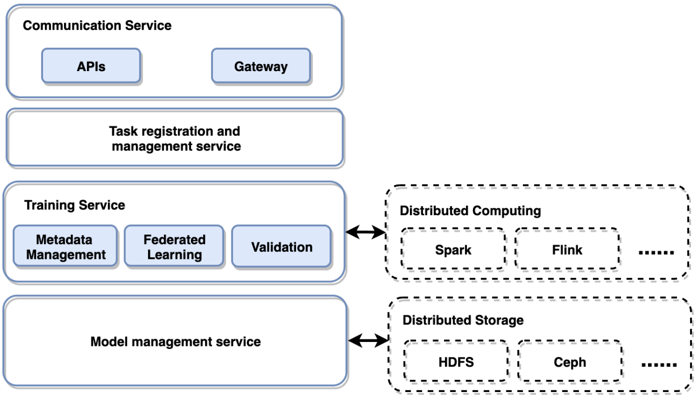


Figure 5.1: Services that a complete Federated Learning system needs to support

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 5.1 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.

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.

Challenges

Based on our empirical results, we derive the challenges for industries stepping towards Federated Learning. Figure 5.2 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.

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.

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

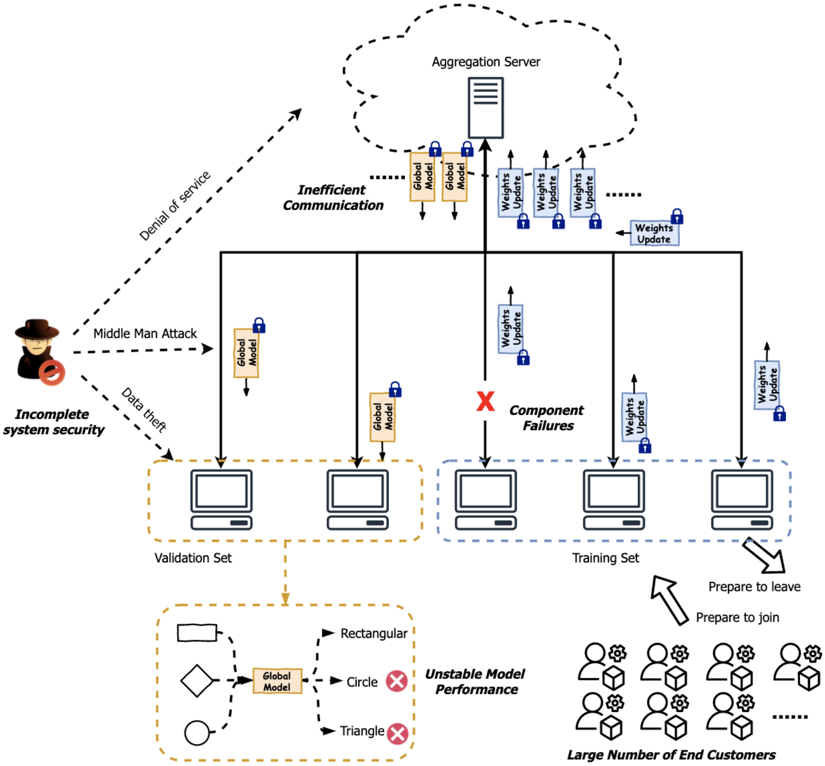


Figure 5.2: Problems and challenges for Federated Learning to be implemented into service-sensitive systems.

keep the connection to 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 [33][145] 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.

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 [31]. 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.

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.

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.

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.

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.

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.

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.

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.

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.

5.5 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 6

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 [146][147][148]. 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 [31].

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 [146]. 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 [32]. 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 [25]). 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"* [149], 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** [71][66]. The situation also applies to Federated Learning systems [150]. Currently, the majority of deployments utilize a single-server centralized architecture which may

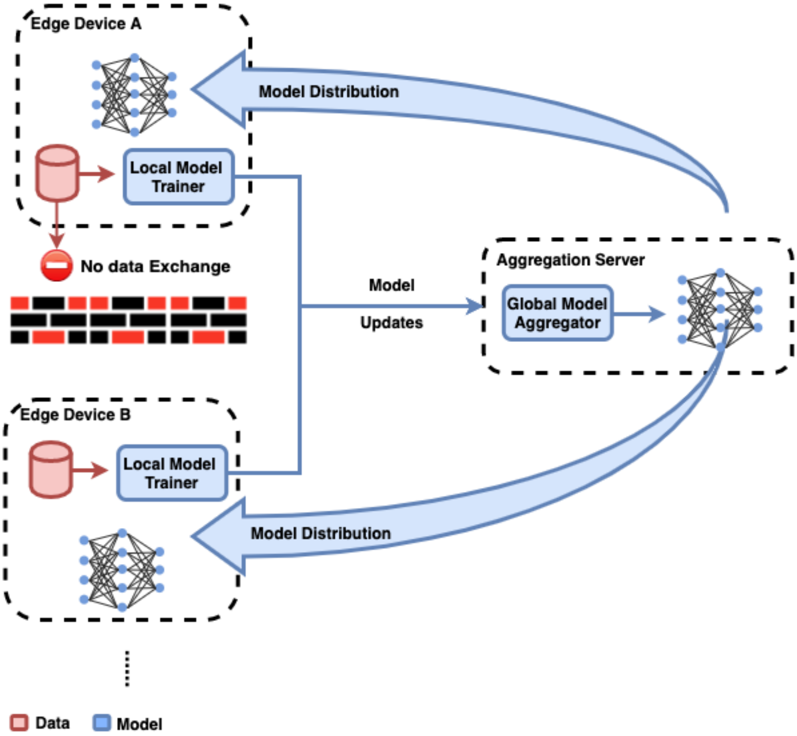


Figure 6.1: System architecture of Federated learning

inevitably face the risk of component failure, system scalability, communication efficiency, etc [31]. 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 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.

6.1 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 [31][146][135][123], 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 6.2 illustrates the concepts.

Centralized Architecture

The centralized architecture is a widely used setting in the majority of current Federated Learning systems[147][129][119]. 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 transmission 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

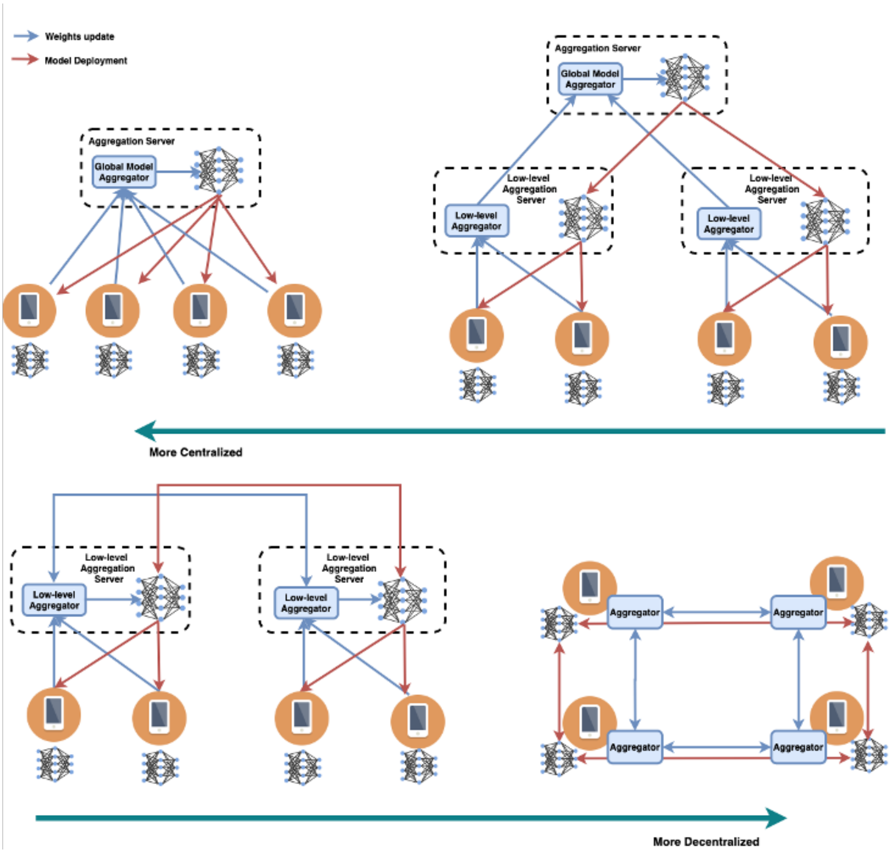


Figure 6.2: Architecture alternatives for Federated Learning systems: centralized, hierarchical, regional and decentralized architecture. For a centralized Federated Learning system ((a)), all the edge nodes are connected to the central aggregation node in order to update local weights and distribute 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.

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.

Hierarchical Architecture

As shown in Figure 6.2 (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 [147], 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.

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 [135]. 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.

Decentralized Architecture

As shown in Figure 6.2 (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 [151]. 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.

6.2 Research Method

In this research, the empirical method and learning procedure described in Zhang [100] 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.

Simulation Testbed

Figure 6.3 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 6.1 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).

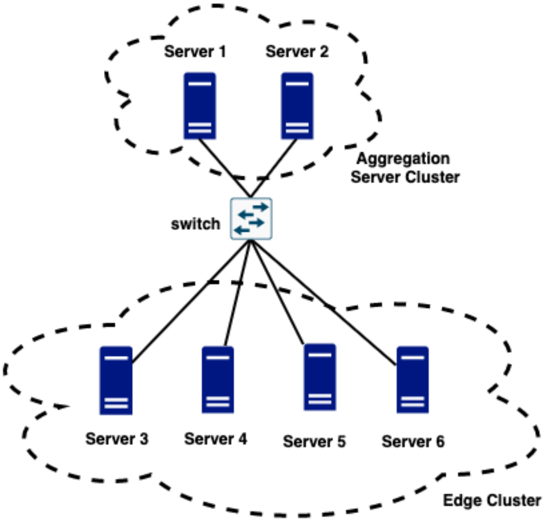


Figure 6.3: Topology of the simulation testbed

Table 6.1: 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.

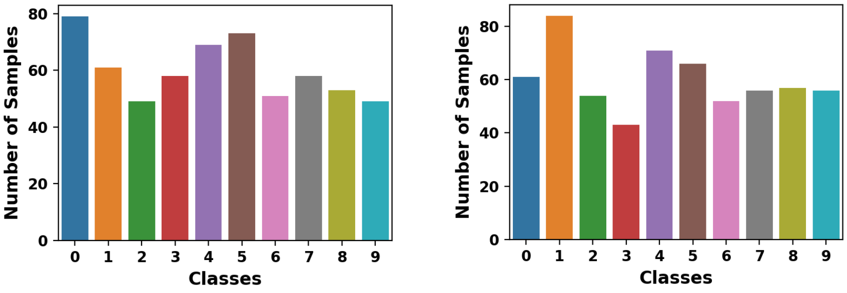
For the regional architecture simulations, the aggregation functions were deployed both on server 1 and 2. Similar to the hierarchical architectures simulation, edge nodes were assigned to one of the aggregation processes once they joined in the system and only communicated with that unique aggregation 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.

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.

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 6.4 outlines the data distribution in two example edge nodes.



(a) Edge 0

(b) Edge 49

Figure 6.4: Uniform training data distribution

Normal Distribution

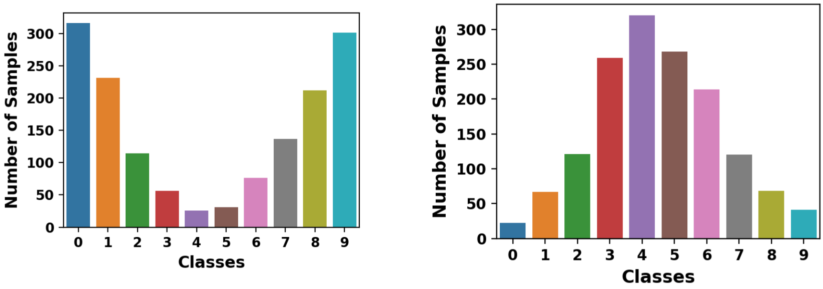
With this setting, in each edge nodes, the number of samples in each class follows the normal density function as shown below.

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

Here, μ and σ are defined as follows:

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

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 6.5 outlines the data distribution in two example edge nodes.



(a) Edge 0

(b) Edge 49

Figure 6.5: Normal training data distribution

Machine Learning Method

The models used in this paper were implemented in Python, using torch 1.4.0 [152], torchvision 0.5.0 [153] and scikit-learn [154] libraries for model building.

In order to achieve a satisfying classification result, two different convolutional neural networks (CNN) [155] 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.

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).

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.

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.

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.

6.3 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) [146] 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 6.1. 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
```

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.

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}.

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.

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

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.

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.

6.4 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 6.2) - (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

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 $t = 1, 2, \ldots$ 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
```

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
```

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.

Weights update latency

Figure 6.6 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 6.2.

The above figure shows that centralized architecture has the

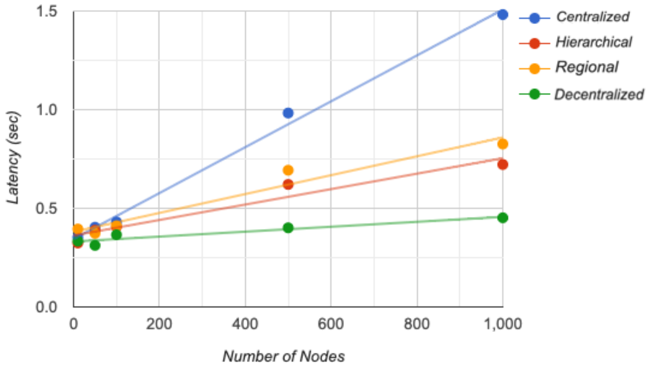


Figure 6.6: Weights latency with different number of nodes in four architecture alternatives

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.

In addition to weights updating latency, the number of retransmission is also measured. From Table 6.3, it can be observed that, when dealing with a large number of edge devices, centralized architecture causes more transmission mistake and less communica-

Table 6.2: 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

tion efficiency than other alternatives. It also proves our findings in weights updating latency.

Table 6.3: 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	-

Model Evolution Time

We then calculated the average model evolution time in all edge nodes, which is outlined in Table 6.4. 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 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 6.4: 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

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 6.5: 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

Uniform Distribution

As described in section 6.2, 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 6.5 for MNIST and Table 6.6 for CIFAR-10.

Table 6.6: 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.

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

Table 6.7: 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 6.8: 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

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 6.7.

Table 6.9: 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 6.8.

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 6.9 and Table 6.10) 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.

6.5 Discussion

According to experiment results, each architectural alternative demonstrates its advantages and disadvantages. In order to help

Table 6.10: 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

industries easily understand the requirements and suitable scenarios for setting up a Federated Learning system, we summarize our findings and suggestions in the following sections.

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 [127][134], 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.

Table 6.11: 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

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 need their system to be scalable and able to tolerant node failure. For example, in mobile applications and wireless systems [148], 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.

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 [156][135]. Furthermore, the system can perform a faster model evolution based on local data but still can partially benefit from global knowledge.

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 [123][110] 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.

6.6 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 6.11 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 7

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 [106]. 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 7.1. 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.

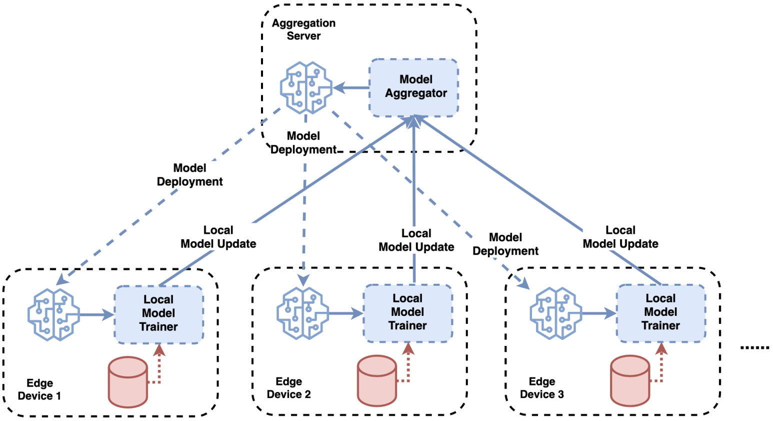


Figure 7.1: A typical Federated Learning System is depicted in the diagram. The light blue components are related to the model, while the red components are related to the data.

This paper builds on our previous research, "End-to-End Federated Learning for Autonomous Driving Vehicles" [157], 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 [158], we realized that it is difficult for businesses to incorporate Federated Learning components into their software products [150]. 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 [159], 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"* [149], 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 [71] [66].

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 7.1, we introduce the background of this study. Section 7.3 details our research method, including the simulation testbed, the utilized machine learning method and the evaluation metrics. Section 7.4 presents the real-time end-to-end Federated Learning approach utilized in this paper. Sections 7.5 evaluates our proposed learning method to empirical data sets. Section 7.6 outlines the discussion on our observed results. Finally, Section 7.7 presents conclusions and future work.

7.1 Background

The first Federated Learning framework was proposed by Google in 2016 [33]. 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 8.6:

$$\min_{w} f(w), \ \ where \ \ f(w) := \sum_{i=1}^{n} \lambda_i f_i(w) \tag{7.1}$$

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. [147] and [148] 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 [150]. 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.

7.2 Related Work

Steering Wheel Angle Prediction

One of the first pioneer research of utilizing the neural network for steering wheel angle prediction is described in [160]. 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% [161]. Shuyang et al. [162] designed a 3D-CNN model with LSTM layers to predict steering wheel angles.

The concept of end-to-end learning was first proposed in [163], 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 [164], 3D-CNN [162], RNN [161], and LSTM [165], the model was first proposed in [166] and applied in [167]. 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.

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 [168][156]. 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 [169].

On top of Federated Learning, Lu et al. [119] test the failure battery for an electric vehicle. Their methods demonstrate the efficacy of privacy serving, latency reduction, and security protection. Saputra et al. [120] 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. [118] 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. [170] 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.

7.3 Method

The analytical technique and research method mentioned in [100] 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.

Mathematical Notations

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

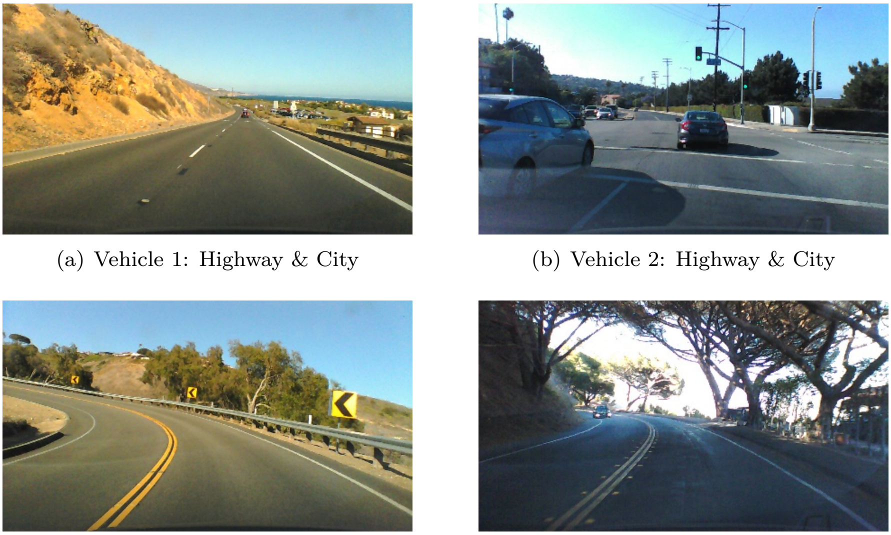
A_t		An image frame matrix at time t
O_t $f(A_t, A_{t-1})$	$=$	An optical-flow matrix at time t
θ_t		Steering wheel angle at time t
$\hat{\theta}_t$		Predicted steering wheel angle at time t

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 [171]. 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.

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 7.2, training datasets for each edge vehicle in our experiment include a variety of driving scenarios.

The data distribution in each edge vehicle is depicted in Figure 7.3. When driving on a hill, the steering wheel angles have a



(a) Vehicle 1: Highway & City

(b) Vehicle 2: Highway & City

(c) Vehicle 3: Hill

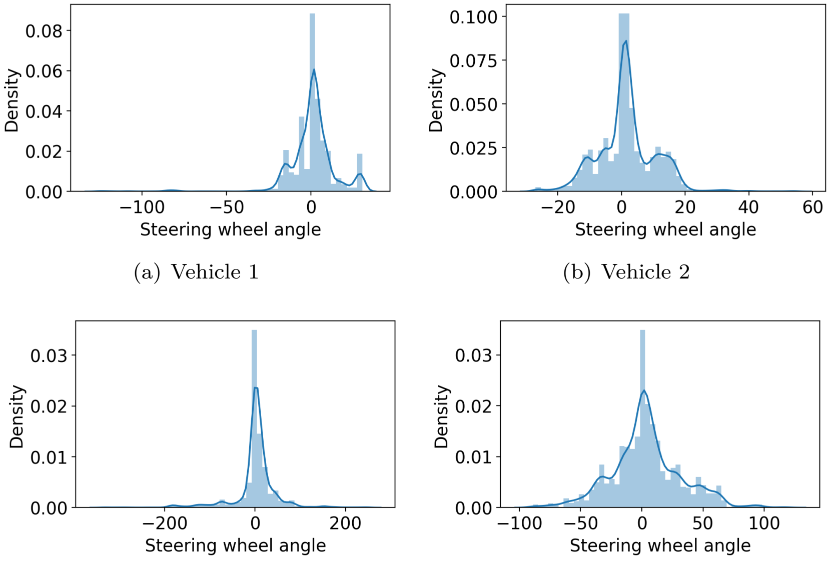
(d) Vehicle 4: Hill & City

Figure 7.2: Driving scenarios in each edge vehicle.

Table 7.1: 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)

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 7.3: 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 7.1. 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

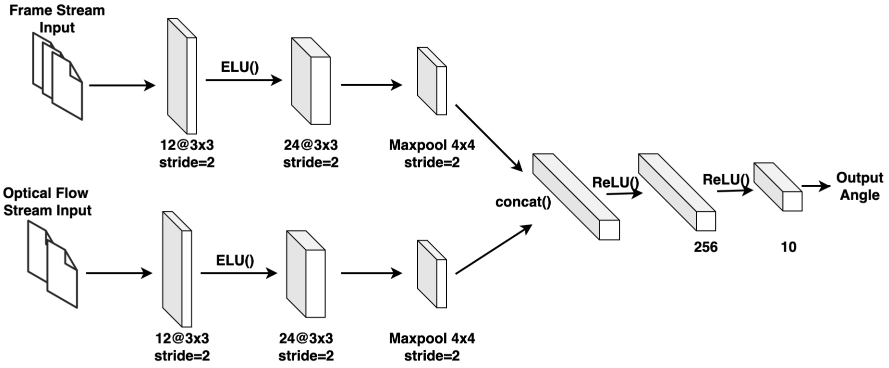


Figure 7.4: The two input branches each have two 3x3 convolution layers in a convolutional neural network. The first layer has 12 output channels that are enabled with the ELU function, while the second layer has 24, which is then followed by 4x4 max pooling. All with stride values of 2 or higher. With the ReLu activation, there are two completely linked layers with 250 and 10 units after concatenating two branches.

4 (Vehicle 1: Tesla V100, Vehicle 3, 4: Tesla T4).

Machine Learning Method

In this paper, steering wheel angle prediction is performed using a two-stream deep Convolutional Neural Network (CNN) [166] [167]. The architecture is described in detail in Figure 7.4. 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 [172]. The optical flow calculation method used in this paper is based on Gunnar Farneback's algorithm, which is implemented in OpenCV [173]. Figure 7.5 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:

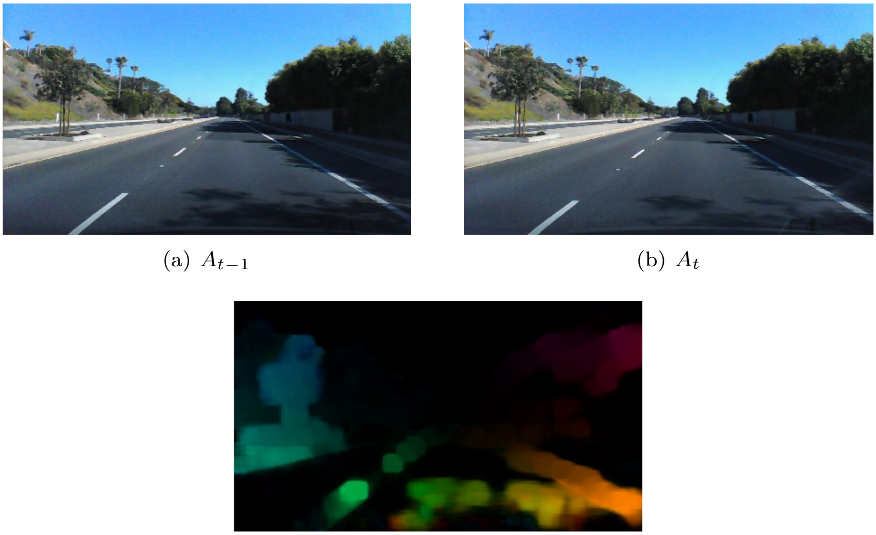
$$Loss = \frac{1}{N}\sum_{t=1}^{N}(\theta_t - \hat{\theta}_t)^2 \tag{7.2}$$

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 [174], with parameters $\beta_1 = 0.6$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$.

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



(a) A_{t-1}

(b) A_t

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

Figure 7.5: Example of the optical flow (a) Previous Frame (b) Current Frame (c) Optical flow of current vision frame.

(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 hyperparameters of this model training are identical to those of Federated Learning, as described in section 7.3. 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.

7.4 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 7.6. 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 version) and $verk$ (local model version of edge vehicle k). The upper limit of the model version difference is represented by a_u,

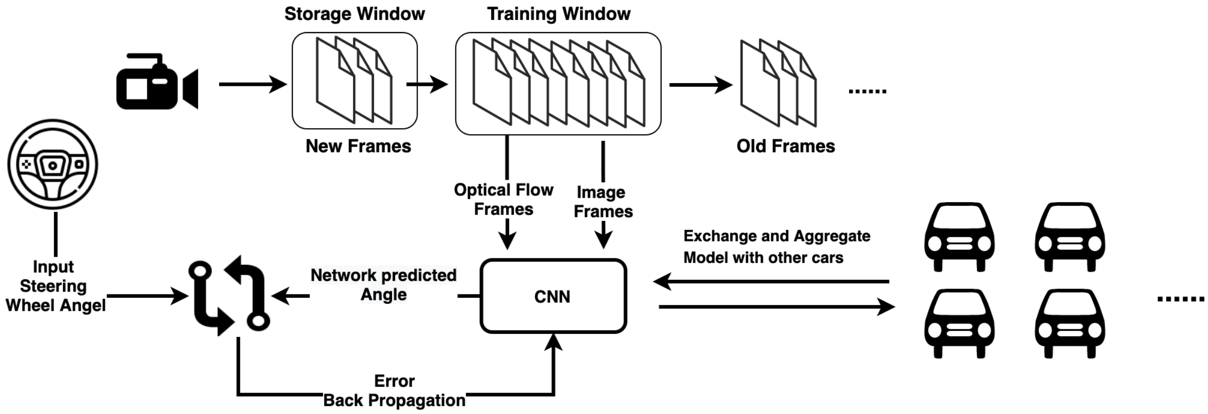


Figure 7.6: Diagram of real-time end-to-end Federated Learning in a single vehicle.

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
        $w_{t+1}^k, ver_k \longleftarrow Client\_Update(w_t, ver)$;
        $w_{t+1} \longleftarrow (1-\alpha) \times w_t + \alpha \times w_{t+1}^k$
        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
```

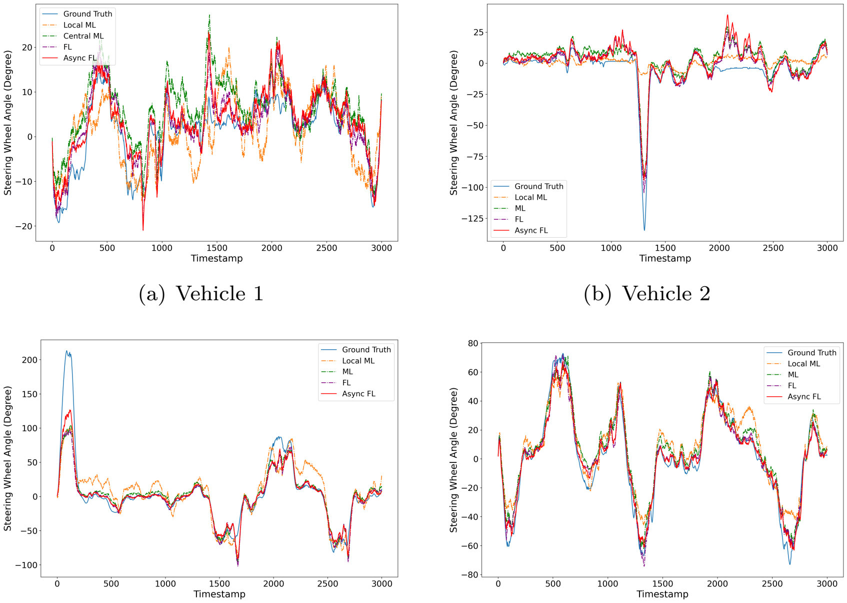
while the lower limit is represented by a_l. There are three conditions:

- 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.)



(a) Vehicle 1

(b) Vehicle 2

(c) Vehicle 3

(d) Vehicle 4

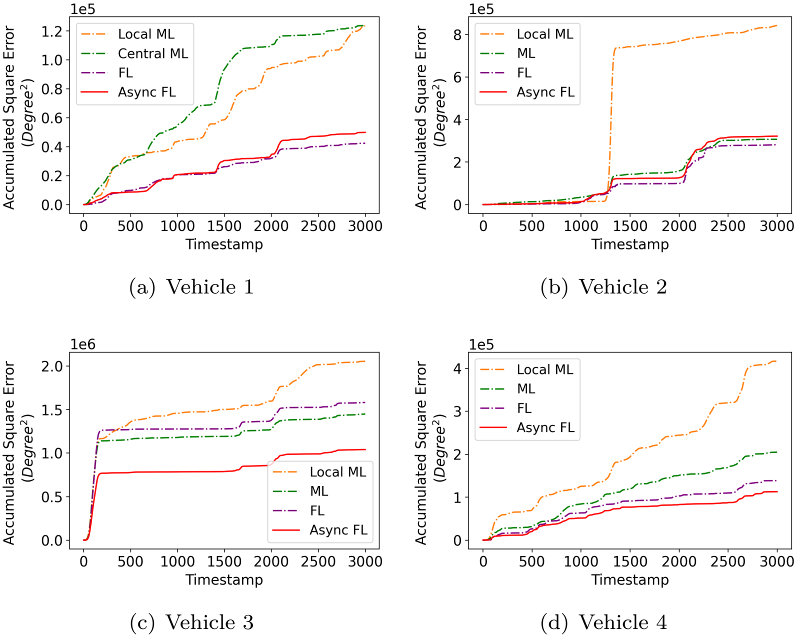
Figure 7.7: The comparison of angle prediction performance on four local vehicle test set with Federated Learning and three baseline models.

7.5 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 7.3. (The metrics are described in 8.2.) - (1) Angle prediction performance (2) Model Training Time (3) Bandwidth cost. The results are compared with other three 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 7.7 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.

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 7.8. The results provide the same information as Figure 7.7. 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 7.2 displays detailed numerical results, including the regression error (RMSE) on each test dataset in each vehicle and the overall av-



(a) Vehicle 1

(b) Vehicle 2

(c) Vehicle 3

(d) Vehicle 4

Figure 7.8: Accumulated error on test dataset in 4 edge vehicles with asynchronous Federated Learning and other baseline models.

erage accuracy among the test datasets of all participating edge vehicles.

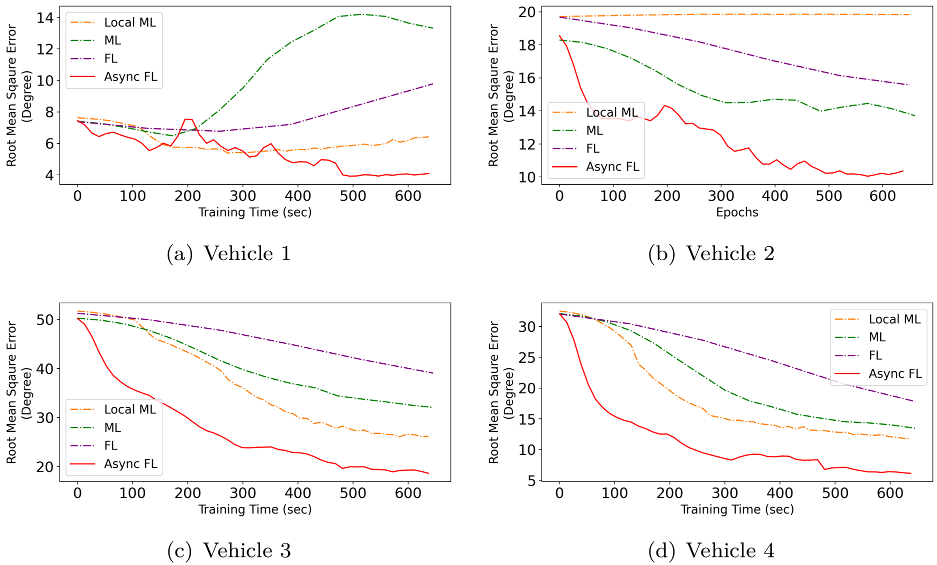
Table 7.2: Steering wheel angle regression error (RMSE) on test set of each edge vehicle (4 vehicles in total)

	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 7.9 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 7.3. For all the models, the total number of training epochs is



(a) Vehicle 1

(b) Vehicle 2

(c) Vehicle 3

(d) Vehicle 4

Figure 7.9: 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 7.3: 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	-

50. With async FL, FL, and Local ML learning approaches, model training is accelerated by Nvidia Tesla 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.

7.6 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.

7.7 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 8

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 [33] in 2016. The concept was originally used to solve the problem of local model training and updating in Android mobile devices [72]. 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 [175].

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 [176], 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 [71]. 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"* [149], 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 [66].

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 8.1, we introduce the background of this study. Section 8.2 details our research method, including the simulation testbed, the utilized machine learning method and the evaluation metrics. Section 8.3 presents the Asynchronous Federated Deep Neural Decision Forests approach proposed in this paper. Sections 8.4 includes evaluation of the proposed method to data sets that are relevant to industrial applications. Section 8.5 outlines the discussion on our observed results. Finally, Section 8.6 presents conclusions and future work.

8.1 Background

Deep Neural Decision Forests

Deep neural decision forests were firstly were first introduced by Kontschieder et al. [177] 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 [178].

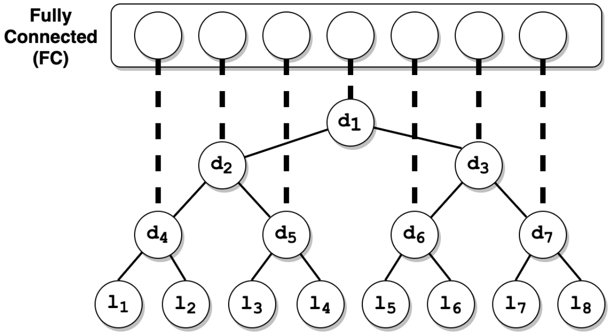


Figure 8.1: Network Structure of the Deep Neural Decision Forests

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 8.1 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.

$$d_n(x;\theta) = \sigma(f_n(x;\theta)) \tag{8.1}$$

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:

$$\mu_l(x|\theta) = \prod_{n \in N} d_n(x;\theta)^{\swarrow} \bar{d}_n(x;\theta)^{\searrow} \tag{8.2}$$

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:

$$\mu_{l_4} = d_1(x)\bar{d}_2(x)\bar{d}_5(x) \tag{8.3}$$

The probability of classifying input x as y is:

$$P[y|x,\theta,\pi] = \sum_{l \in L} \pi_{l_y} \mu_l(x|\theta) \tag{8.4}$$

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

$$P_F[y|x] = \frac{1}{k} \sum_{h=1}^{k} P_{T_h}[y|x] \tag{8.5}$$

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

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 [71]. 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 [42].

With the concept of cloud computing and decentralized data storage, AI engineering [149] 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 [33][73] 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 8.6 in particular:

$$\min_{w} f(w), \;\; where \;\; f(w) := \sum_{i=1}^{n} \lambda_i f_i(w) \tag{8.6}$$

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 [72], there have been several Federated Learning architectures, frameworks and solutions proposed to solve real-world issues [31]. 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 [158], 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 [179], 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.

8.2 Method

To produce a quantitative assessment and comparison between Federated Learning and centralized learning techniques, the empirical method and learning procedure provided in [100] 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.

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 [180]. 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 8.2 demonstrate the example samples in the data set.

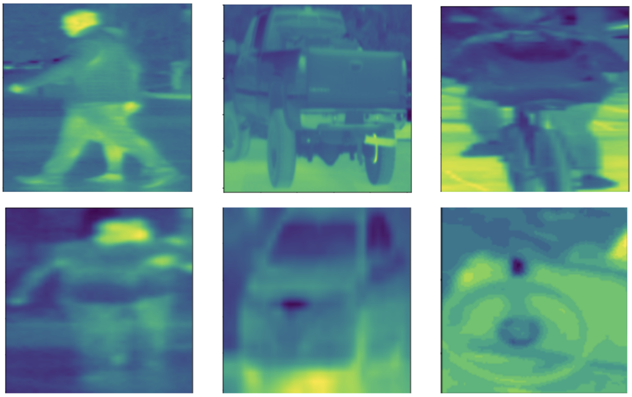


Figure 8.2: FLIR Thermal Sensing Advanced driver-assistance system Dataset

The second data set applied is BDD 100K [181], which was re-

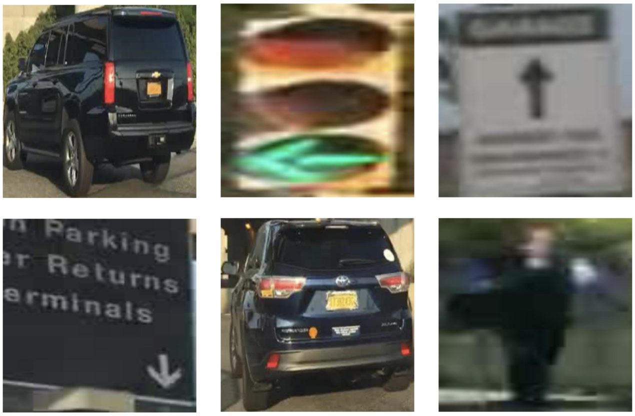


Figure 8.3: BDD100K: A Large-scale Diverse Driving Video Dataset

leased 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 8.3 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

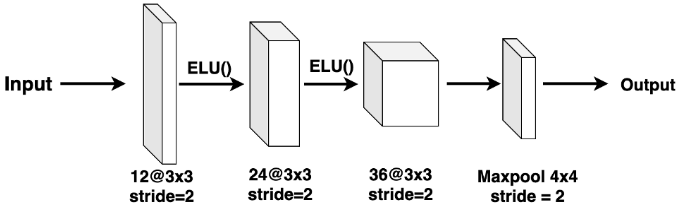


Figure 8.4: Convolutional neural network layer description

to 512 × 512 and normalized to $[-1, 1]$.

Hardware information for all of the servers is provided in Table 8.1. In order to simulate aggregation and edge operations, one server was designated as the aggregation server, while the others were designated as edge vehicles.

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 T4 GPU

Machine Learning Method

In order to find the optimal network, the random search [182] 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 8.4 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 are activated with the ELU function [183]. 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:

$$L(\theta, \pi; x, y) = -log(P_T[y|x, \theta, \pi]) \tag{8.7}$$

For the hyper-parameter of decision forests, we list all the settings in the following table 8.2:

Table 8.2: Hyper-parameter settings for Decision Forests layer

NUMBER_EPOCHS	20
TREE_DEPTH	8
NUMBER_TREE	12
FEATURE_RATE	0.75
DROPOUT_RATE	0.05

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 machine learning methods in an industrial context, namely model performance, model training efficiency and communication cost for data transfer between the edge nodes and

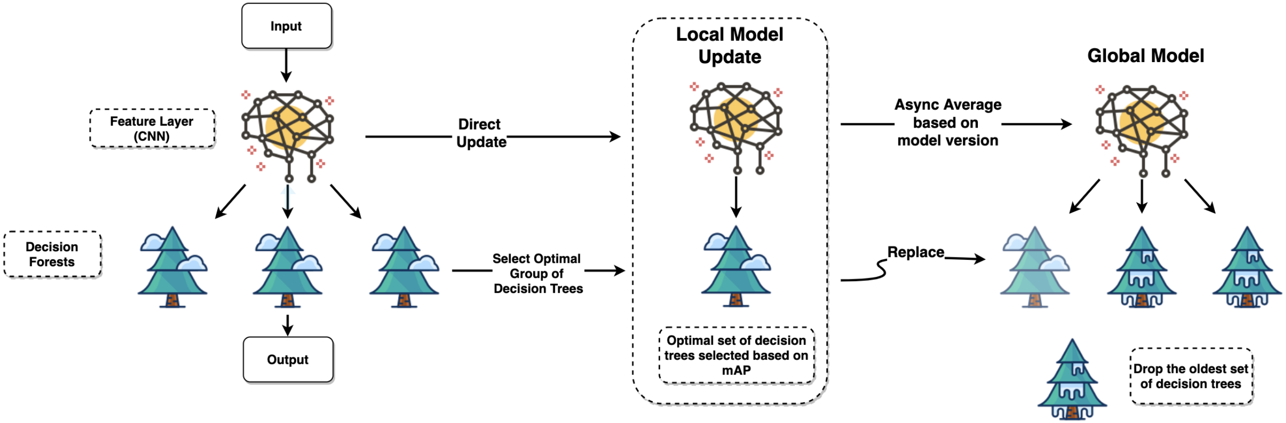

Input
Feature Layer (CNN)
Decision Forests
Output
Direct Update
Select Optimal Group of Decision Trees
Local Model Update
Optimal set of decision trees selected based on mAP
Async Average based on model version
Replace
Global Model
Drop the oldest set of decision trees

Figure 8.5: Local updating and aggregation of the Asynchronous Federated Deep Neural Decision Forests

the central server. [3].

- 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.

$$AUCC = \frac{T}{T + F} \tag{8.8}$$

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 [184][185].

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{8.9}$$

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 8.2.

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 [158], 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.

8.3 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 8.6 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 8.5 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.

$$mAP = \frac{\sum_{k=1}^{K} AP(k)}{K} \tag{8.10}$$

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.

In this paper, as we described in Section 8.2, 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 8.6 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

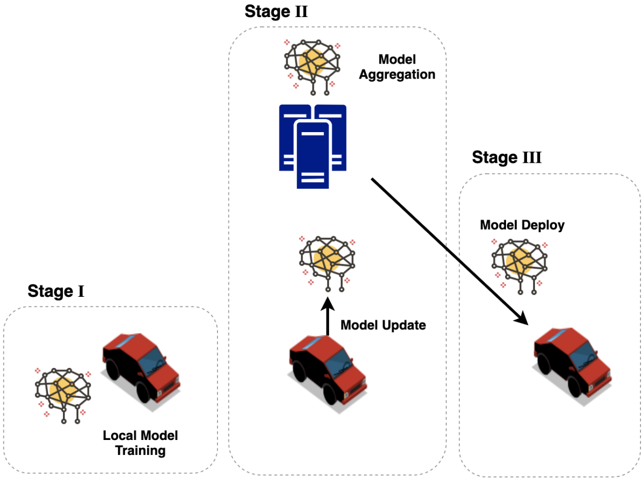


Figure 8.6: Three stages of the Asynchronous Federated Deep Neural Decision Forests

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 sub-

mitted 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:

$$w_{t+1} \longleftarrow (1-\alpha) \times w_t + \alpha \times w_{t+1}^l \tag{8.11}$$

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

$$\alpha = \frac{1}{ver_g - ver_l} \tag{8.12}$$

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.

8.4 Results

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 8.2 while the CNN network is demonstrated in Figure 8.4.) The results of the FLIR data set are listed in Table 8.3.

Table 8.3: 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.

The same conclusion can be obtained with BDD 100K data set. Table 8.4 gives the classifier performance among three different machine learning methods. The classification accuracy can increase around 18% compared with RF and about 5% compared

Table 8.4: 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 8.5: 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

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 8.5 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.

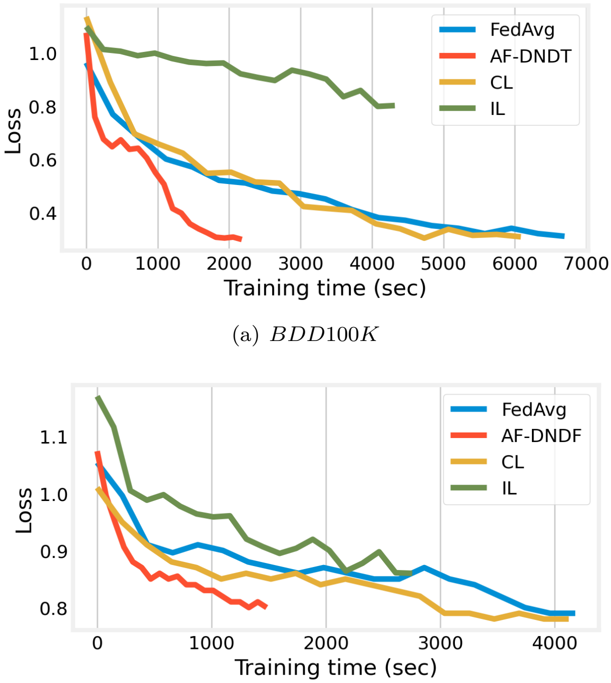
Model Training Efficiency

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 8.7 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.

Bandwidth Utilization

During the experiment, we also recorded bandwidth utilization by applying each learning approach. Figure 8.8 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.

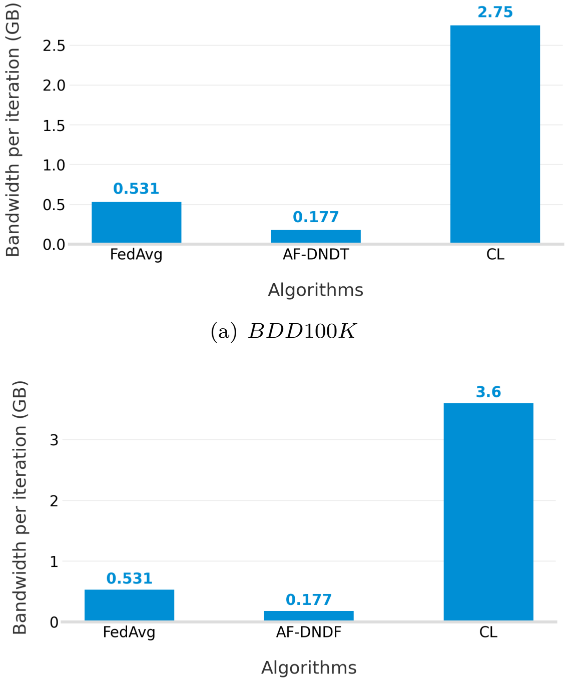
In summary, AF-DNDF reduces training time by around 60% and saves bandwidth by about 80% when compared to the centralized learning approach, which is cost-effective and suitable to real-world systems.



(a) $BDD100K$

(b) $FLIR$

Figure 8.7: The comparison between model training loss and the model training time with AF-DNDF and three baseline models



(a) $BDD100K$

(b) $FLIR$

Figure 8.8: Bandwidth consumption of different learning algorithm with two data sets

8.5 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.

8.6 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 9

Concluding Remarks and Future Work

To sum up, in this thesis, we present research 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. The empirical results show that due to privacy and data collection issues, more and more industrial cases are trying to solve their problem by using the Federated Learning framework. When implementing the Federated Learning method into a real-world context, we have proposed an asynchronous aggregation way to communicate between the edge and server to avoid synchronization problems when dealing with large scale systems. Synchronization and communication efficiency are the top priorities for a company when implementing model training and sharing components into heterogeneous hardware settings. We also combined Federated Learning with advanced machine learning methods such as neural decision forests and validate our solution with two important use cases, including steer wheel angle prediction and image recognition. Our results show the effectiveness of applying the Federated Learning method and our improved strategies if compared with widely used centralized learning methods.

Furthermore, we also 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. The research questions framed in Chapter 3 are addressed as follows:

RQ1. What are the main challenges with existing Machine Learning workflows and how can Federated Learning help in addressing and solving these challenges?

In order to answer this question, we conducted a literature review and interview-based case study to identify the benefits of applying Federated Learning to embedded systems. We figure out that there is three major concern associated with the typical machine learning workflow in the embedded system domain, including privacy, efficiency and storage concern. The Federated Learning method has the distinct advantage of retaining privacy, as there is no need to transfer the original data, which is always kept local to the device. The new model enables the device to download the model in real-time and iterate over it, allowing the new model to respond to user behaviour as quickly as possible, resulting in a real-time model update. The method is suitable for distributed systems where data cannot or is hard to be collected to a central location in the typical machine learning workflow. Furthermore, Federated Learning can reduce network latency, transmission usage, and make the most use of local computer capacity. Despite the impending introduction of 5G, internet speeds are not guaranteed in all circumstances and locales. In a sluggish network speed scenario, if all of the user's data is uploaded to the cloud and the service itself is fed from the cloud, network latency will dramatically damage the user experience. This is not true for Federated Learning-enabled services because the service is obtained locally. The studies help us gain the basic knowledge and motivation to help companies implement Federated Learning components.

RQ2. What are the primary constraints and limitations of current Federated Learning systems?

To answer this topic, we draw on current literature as well as the experience of industrial engineers. We identified the challenges that industries are attempting to address when adopting and transitioning their machine learning components to Federated Learning, such as component failures, inefficient communication, unstable model performance, large-scaled end customers, and insufficient system security. Furthermore, we provided six open research topics that will drive future studies on the implementation of Federated Learning in real-world contexts.

RQ3. What are the solutions that can help companies in the embedded systems domain build Federated Learning in practice?

We proposed the major components and five critical criteria for constructing and operating a dependable industrial Federated Learning system based on prior knowledge acquired from books and industrial engineers. Furthermore, in order to address the challenges associated with the transition to Federated Learning, we offered and examined various architecture alternatives, as well as suggested viable approaches for certain industrial scenarios. We devised an asynchronous aggregation protocol for Federated Learning to analyze and learn data in real-time in order to tackle the heterogeneous hardware difficulties. We also coupled Federated Learning with neural decision forests and optimized the model sharing mechanism to improve model learning performance while reducing communication overhead. Our results show that using Federated Learning and our upgraded approaches, the system can achieve the same or even greater prediction accuracy than commonly used centralized learning methods while lowering training time and bandwidth costs. The reduction would be more evident if the number of participating devices were increased, proving to be cost-effective and applicable to real-world systems.

9.1 Key Contributions

Based on our research and collaboration with companies in the software-intensive embedded systems domain at Software Center, we first identify the difficulties associated with traditional machine learning approaches and the challenges that companies encountered (data collection, model training, model distribution, etc.) while attempting to use machine learning to improve service quality in this thesis. We investigate why industries believe Federated Learning is a viable solution to the issues connected with the deployment of machine learning components in the domain of embedded systems, and we demonstrate the benefits that companies get after making the switch to Federated Learning. In order to further investigate Federated Learning and the ways to improve the method, we identified the key challenges and limitations of existing Federated Learning systems, as well as the challenges that industries in the embedded system domain are attempting to solve when adopting and transitioning to Federated Learning. We review the services and requirements needed for a dependable Federated Learning system and offer six open research problems for future Federated Learning research. We introduce four architecture alternatives that have been or can be applied to a Federated Learning system based on the challenges we identified and discussed with our industrial collaborators. The benefits and drawbacks of several architecture solutions for Federated Learning systems are discussed. In order to overcome the communication and training efficiency problem, we present a unique real-time end-to-end Federated Learning technique for asynchronously training Machine Learning models in a distributed context and consuming real-time streaming data for Federated Learning. The asynchronous aggregation protocol is also combined with the concept of Deep Neural Decision Forests. The proposed methods are validated with key automotive use cases, steering wheel angle prediction and road objective recognition in the field of autonomous driving by using empirical datasets. In summary, the main objectives of contribution are as follows:

- Objective 1: To identify the limitations of current machine learning workflow

- Objective 2: To identify how Federated Learning can influence the business and improve the service quality in the embedded systems

- Objective 3: To identify what are the main challenges of Federated Learning systems in a real-world context.

- Objective 4: To propose the solution which can help companies to implement Federated Learning components.

9.2 Future Work

In the future, we plan to further analyze our algorithm with different combinations of hyper-parameters and aggregation protocols, such as quality-based aggregation algorithms. As the parameter settings become more important with the increase of the number of participating learning vehicles, 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.

Another direction for our future work is to explore autonomously improving algorithms, such as reinforcement learning, etc. Reinforcement learning has re-emerged as a topic of interest in academia and industry. When implemented in a real-world environment, however, reinforcement learning is typically shown to be unstable and unable to generalize to varied contexts. To our knowledge, the majority of reinforcement learning research either places the algorithm in a game context or assumes a fixed running space. However, because the real world is a complex dynamic system rather than a static environment, the practicality of the reinforcement learning method must also be addressed in real-world applications. Because most reinforcement learning models are generated in a static environment, methods that enable the industry to apply the methods in real-world embedded systems are worth investigating. Reinforcement learning (RL) is a learning approach that is concerned with how software agents respond in an environment in order to maximize cumulative rewards. Because of this characteristic, the agent can learn from past experiences and iterate to adapt to the environment. In the next stage, we'd like to look at the viability of combining Federated Learning and reinforcement learning to allow edge devices to evolve themselves without the need for human intervention. We'd like to conduct research into how to merge those two approaches and the way to efficiently train a reinforcement learning model in a Federated Learning system. We would like to work with our industrial partners to test our techniques in various application domains.

References

[1] P. Harrington, *Machine learning in action.* Simon and Schuster, 2012.

[2] 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.

[3] P. Larrañaga, D. Atienza, J. Diaz-Rozo, A. Ogbechie, C. Puerto-Santana, and C. Bielza, *Industrial applications of machine learning.* CRC press, 2018.

[4] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, Elsevier, 1992, pp. 65–93.

[5] C. W. Hanson III and B. E. Marshall, "Artificial intelligence applications in the intensive care unit," *Critical care medicine*, vol. 29, no. 2, pp. 427–435, 2001.

[6] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, "Systematic review of research on artificial intelligence applications in higher education - where are the educators?" *International Journal of Educational Technology in Higher Education*, vol. 16, no. 1, pp. 1–27, 2019.

[7] A. Bahrammirzaee, "A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems," *Neural Computing and Applications*, vol. 19, no. 8, pp. 1165–1195, 2010.

[8] H. Mannila, "Data mining: Machine learning, statistics, and databases," in *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*, IEEE, 1996, pp. 2–9.

[9] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[10] T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin, and R. Shanmugam, "Ai for testing today and tomorrow: Industry perspectives," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2019, pp. 81–88.

[11] P. Niewiadomski, A. Stachowiak, and N. Pawlak, "Knowledge on it tools based on ai maturity–industry 4.0 perspective," *Procedia Manufacturing*, vol. 39, pp. 574–582, 2019.

[12] B. S. Sergi, E. G. Popkova, A. V. Bogoviz, and T. N. Litvinova, *Understanding industry 4.0: AI, the internet of things, and the future of work*. Emerald Group Publishing, 2019.

[13] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Guéhéneuc, "Studying software engineering patterns for designing machine learning systems," in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, IEEE, 2019, pp. 49–495.

[14] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine learning on big data," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, IEEE, 2013, pp. 1242–1244.

[15] S. Amershi, A. Begel, C. Bird, *et al.*, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, 2019, pp. 291–300.

[16] 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.

[17] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[18] 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.

[19] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: A survey of case studies," *arXiv preprint arXiv:2011.09926*, 2020.

[20] L. Baier, F. Jöhren, and S. Seebacher, "Challenges in the deployment and operation of machine learning in practice.," in *ECIS*, 2019.

[21] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Information and software technology*, vol. 127, p. 106 368, 2020.

[22] A. Shabtai, Y. Elovici, and L. Rokach, *A survey of data leakage detection and prevention solutions.* Springer Science & Business Media, 2012.

[23] A. Yapo and J. Weiss, "Ethical implications of bias in machine learning," 2018.

[24] M. Coeckelbergh, "Artificial intelligence: Some ethical issues and regulatory challenges," *Technology and regulation*, vol. 2019, pp. 31–34, 2019.

[25] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[26] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[27] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–36, 2021.

[28] 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.

[29] A. Holzinger, P. Kieseberg, E. Weippl, and A. M. Tjoa, "Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable ai," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2018, pp. 1–8.

[30] D. S. Char, M. D. Abràmoff, and C. Feudtner, "Identifying ethical considerations for machine learning healthcare applications," *The American Journal of Bioethics*, vol. 20, no. 11, pp. 7–17, 2020.

[31] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[32] 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.

[33] 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.

[34] T. Hiessl, D. Schall, J. Kemnitz, and S. Schulte, "Industrial federated learning–requirements and system design," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2020, pp. 42–53.

[35] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[36] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng, "Machine learning basics," *Deep learning*, pp. 98–164, 2016.

[37] I. El Naqa and M. J. Murphy, "What is machine learning?" In *machine learning in radiation oncology*, Springer, 2015, pp. 3–11.

[38] T. O. Ayodele, "Types of machine learning algorithms," *New advances in machine learning*, vol. 3, pp. 19–48, 2010.

[39] E. Alpaydin, *Introduction to machine learning.* MIT press, 2020.

[40] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia*, Springer, 2008, pp. 21–49.

[41] T. Hastie, R. Tibshirani, and J. Friedman, “Overview of supervised learning,” in *The elements of statistical learning*, Springer, 2009, pp. 9–41.

[42] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[43] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[44] H. B. Barlow, “Unsupervised learning,” *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[45] M. E. Celebi and K. Aydin, *Unsupervised learning algorithms.* Springer, 2016.

[46] J. J. Oliver, R. A. Baxter, and C. S. Wallace, “Unsupervised learning using mml,” in *ICML*, Citeseer, 1996, pp. 364–372.

[47] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[48] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[49] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, “An overview of machine learning,” *Machine learning*, pp. 3–23, 1983.

[50] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[51] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[52] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning.* MIT press Cambridge, MA, USA, 2017, vol. 1.

[53] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[54] K. Tanaka, *Embedded Systems: High Performance Systems, Applications and Projects.* BoD–Books on Demand, 2012.

[55] A. Berger, *Embedded systems design: an introduction to processes, tools, and techniques*. CRC Press, 2001.

[56] L. P. Kaelbling, *Learning in embedded systems*. MIT press, 1993.

[57] 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.

[58] S. Branco, A. G. Ferreira, and J. Cabral, "Machine learning in resource-scarce embedded systems, fpgas, and end-devices: A survey," *Electronics*, vol. 8, no. 11, p. 1289, 2019.

[59] J. Lee, M. Stanley, A. Spanias, and C. Tepedelenlioglu, "Integrating machine learning in embedded sensor systems for internet-of-things applications," in *2016 IEEE international symposium on signal processing and information technology (ISSPIT)*, IEEE, 2016, pp. 290–294.

[60] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge ai application," *IEEE Access*, vol. 8, pp. 9102–9114, 2020.

[61] S. Akter, K. Michael, M. R. Uddin, G. McCarthy, and M. Rahman, "Transforming business using digital innovations: The application of ai, blockchain, cloud and data analytics," *Annals of Operations Research*, pp. 1–33, 2020.

[62] J. Welser, J. Pitera, and C. Goldberg, "Future computing hardware for ai," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 1–3.

[63] Y.-L. Lee, P.-K. Tsung, and M. Wu, "Techology trend of edge ai," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, IEEE, 2018, pp. 1–2.

[64] R. J. Solomonoff, "Machine learning-past and future," *Dartmouth, NH, July*, 2006.

[65] M. F. Deering, "Architectures for ai: Hardware and software for efficient processing," *Byte*, vol. 10, no. 4, pp. 193–206, 1985.

[66] 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.

[67] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2017, pp. 1–8.

[68] A. Kumar, M. Boehm, and J. Yang, "Data management in machine learning: Challenges, techniques, and systems," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1717–1722.

[69] S. Reddy, S. Allan, S. Coghlan, and P. Cooper, "A governance model for the application of ai in health care," *Journal of the American Medical Informatics Association*, vol. 27, no. 3, pp. 491–497, 2020.

[70] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge ai: Algorithms and systems," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2167–2191, 2020.

[71] 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.

[72] T. Yang, G. Andrew, H. Eichner, *et al.*, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[73] 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.

[74] D. T. Campbell and J. C. Stanley, *Experimental and quasi-experimental designs for research.* Ravenio books, 2015.

[75] 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.

[76] G. Heitink, *Practical theology: History, theory, action domains: Manual for practical theology.* Wm. B. Eerdmans Publishing, 1999.

[77] R. Malhotra, *Empirical research in software engineering: concepts, analysis, and applications.* CRC press, 2016.

[78] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in software engineering," in *Empirical methods and studies in software engineering*, Springer, 2003, pp. 7–23.

[79] A. Dresch, D. P. Lacerda, and J. A. V. Antunes, "Design science research," in *Design science research*, Springer, 2015, pp. 67–102.

[80] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.

[81] P. J. Denning, "A new social contract for research," *Communications of the ACM*, vol. 40, no. 2, pp. 132–134, 1997.

[82] P. Y. Papalambros, "Design science: Why, what and how," *Design Science*, vol. 1, 2015.

[83] 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.

[84] 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.

[85] A. Strauss and J. Corbin, *Basics of qualitative research.* Sage publications, 1990.

[86] A. Fink, *Conducting research literature reviews: From the internet to paper.* Sage publications, 2019.

[87] 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.

[88] 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.

[89] 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.

[90] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[91] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons, 2012.

[92] 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.

[93] S. Baskarada, “Qualitative case study guidelines,” *Baškarada, S.(2014). Qualitative case studies guidelines. The Qualitative Report*, vol. 19, no. 40, pp. 1–25, 2014.

[94] C. Robson and K. McCartan, *Real world research: a resource for users of social research methods in applied settings.* Wiley, 2016.

[95] R. E. Stake, “Qualitative case studies.,” 2008.

[96] 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.

[97] O. Kempthorne, “The design and analysis of experiments.,” 1952.

[98] 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.

[99] D. C. Montgomery, *Design and analysis of experiments.* John wiley & sons, 2017.

[100] D. Zhang and J. J. Tsai, “Machine learning and software engineering,” *Software Quality Journal*, vol. 11, no. 2, pp. 87–119, 2003.

[101] “Collaborating partners, software center.” (), [Online]. Available: `https://www.software-center.se/` (visited on 03/15/2022).

[102] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[103] L. Bickman, *Validity and social experimentation.* Sage, 2000, vol. 1.

[104] 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.

[105] L. Bickman and D. J. Rog, *The SAGE handbook of applied social research methods.* Sage publications, 2008.

[106] 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.

[107] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[108] R. Feldt. "Isi se journals (ranked)." (), [Online]. Available: `http://www.robertfeldt.net/advice/se%5C_venues/`.

[109] 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.

[110] 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.

[111] 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.

[112] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, 2020.

[113] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[114] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[115] 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.

[116] 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.

[117] 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.

[118] 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.

[119] 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.

[120] 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.

[121] 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.

[122] 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.

[123] 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.

[124] 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.

[125] J. Mills, J. Hu, and G. Min, “Communication-efficient federated learning for wireless edge intelligence in iot,” *IEEE Internet of Things Journal*, 2019.

[126] 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.

[127] 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.

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] D. Verma, S. Julier, and G. Cirincione, “Federated ai for building ai solutions across multiple agencies,” *arXiv preprint arXiv:1809.10036*, 2018.

[133] 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.

[134] 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.

[135] 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.

[136] D. Shultz, *When your voice betrays you*, 2015.

[137] 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.

[138] 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.

[139] 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.

[140] G. Walsham, "Interpretive case studies in is research: Nature and method," *European Journal of information systems*, vol. 4, no. 2, pp. 74–81, 1995.

[141] R. K. Yin, *Case study research and applications: Design and methods.* Sage publications, 2017.

[142] J. A. Maxwell, *Qualitative research design: An interactive approach.* Sage publications, 2012, vol. 41.

[143] L. Sgier, "Qualitative data analysis," *An Initiat. Gebert Ruf Stift*, vol. 19, pp. 19–21, 2012.

[144] C. Rivas, "Coding and analysing qualitative data," *Researching society and culture*, vol. 3, no. 2012, pp. 367–392, 2012.

[145] 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.

[146] 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.

[147] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[148] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[149] J. Bosch, I. Crnkovic, and H. H. Olsson, *Engineering ai systems: A research agenda*, 2020.

[150] 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.

[151] 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.

[152] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[153] 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.

[154] 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.

[155] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[156] 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.

[157] 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.

[158] 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.

[159] 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.

[160] D. Pomerleau, "An autonomous land vehicle in a neural network," *Advances in neural information processing systems'(Morgan Kaufmann Publishers Inc., 1989)*, vol. 1, 1998.

[161] 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.

[162] S. Du, H. Guo, and A. Simpson, "Self-driving car steering angle prediction based on image recognition," *arXiv preprint arXiv:1912.05440*, 2019.

[163] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[164] 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.

[165] 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.

[166] 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.

[167] N. Fernandez, "Two-stream convolutional networks for end-to-end learning of self-driving cars," *arXiv preprint arXiv:1811.05785*, 2018.

[168] 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.

[169] 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.

[170] S. Doomra, N. Kohli, and S. Athavale, "Turn signal prediction: A federated learning case study," *arXiv preprint arXiv:2012.12401*, 2020.

[171] SullyChen. "Collection of labeled car driving datasets." (2018), [Online]. Available: `https://github.com/SullyChen/driving-datasets`.

[172] 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.

[173] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*, Springer, 2003, pp. 363–370.

[174] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[175] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[176] 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.

[177] 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.

[178] L. Rokach, "Decision forest: Twenty years of research," *Information Fusion*, vol. 27, pp. 111–125, 2016.

[179] H. Zhang, J. Bosch, and H. H. Olsson, *Real-time end-to-end federated learning: An automotive case study*, 2021.

[180] FLIR. “Flir thermal dataset for algorithm training.” (2018), [Online]. Available: `https://www.flir.in/oem/adas/adas-dataset-form/`.

[181] 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.

[182] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization.,” *Journal of machine learning research*, vol. 13, no. 2, 2012.

[183] 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.

[184] 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.

[185] Scikit-learn. “Scikit-learn: Average precision.” (2007), [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html`.