New Paradigms of Distributed AI for Improving 5G-Based Network Systems Performance

by

Khaled Bedda
B.Sc. in Biomedical Engineering and Systems, Cairo University, 2021

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL INTELLIGENCE)

Supervisory Committee

Dr. Zubair Fadlullah
Supervisor
(*Associate Professor, Department of Computer Science, University of Western Ontario, London, ON, Canada.*)

Dr. Mostafa Fouda
Co-Supervisor
(*Associate Professor, Department of Electrical and Computer Engineering, Idaho State University, Pocatello, ID, USA.*)

Dr. Muhammad Asaduzzaman
Internal Examiner
(*Assistant Professor, Department of Computer Science, Lakehead University, Thunder Bay, ON, Canada.*)

Dr. Al-Sakib Khan Pathan
External Examiner
(*Professor, Department of Computer Science and Engineering, United International University (UIU), Bangladesh.*)

ABSTRACT

With the advent of 5G technology, there is an increasing need for efficient and effective machine learning techniques to support a wide range of applications, from smart cities to autonomous vehicles. The research question is whether distributed machine learning can provide a solution to the challenges of large-scale data processing, resource allocation, and privacy concerns in 5G networks. The thesis examines two main approaches to distributed machine learning: split learning and federated learning. Split learning enables the separation of model training and data storage between multiple devices, while federated learning allows for the training of a global model using decentralized data sources. The thesis investigates the performance of these approaches in terms of accuracy, communication overhead, and privacy preservation. The findings suggest that distributed machine learning can provide a viable solution to the challenges of 5G networks, with split learning and federated learning techniques showing promising results for spectral efficiency, resource allocation, and privacy preservation. The thesis concludes with a discussion of future research directions and potential applications of distributed machine learning in 5G networks.

In this thesis, we investigate four case studies of both 5G network systems and LTE and Wifi (legacy parts). In chapter3, we implement an asynchronous federated learning model to predict the RSSI in robot localization indoor and outdoor environments. The proposed framework provides a good performance in terms of convergence, accuracy, and overhead reduction. In chapter4, we transfer the deployment of the asynchronous federated learning framework from the Wifi use case to a part of 5G networks (Network slicing), where we use the framework to predict the slice type for rapid and automated intelligent resource allocation. The results demonstrated the same findings in chapter3. In chapter5, we introduce a novel split learning framework to fill the gap of some drawbacks of the federated learning framework. We use it to predict mmWave 5G throughput in a traffic analysis of wireless networks. The results demonstrate better performance compared to the vanilla split learning approach. Finally, In chapter6, we set a comparison of performance between the two proposed frameworks. We deploy the frameworks to predict the channel quality indicator in terms of SNR. The results yield that the proposed split learning approach performs better than the federated one.

Overall, We propose two methodologies; PC-SSL peer-coordinated sequential slit learning and AFD Asynchronous federated learning. The proposed PC-SSL minimizes the data transmitted between the client BSs and a server by processing data locally on the clients. This results in low latency and computation overhead in making handoff decisions and other network operations. At the same time, the Federated learning model permits a reasonably accurate decision for the resource allocation for different 5G users without violating their privacy or introducing additional load to the network. Experimental results demonstrate the efficiency of the asynchronous weight updating federated learning in contrast with the conventional FedAvg (Federated averaging) strategy and the traditional centralized learning model. In particular, our proposed technique achieves network overhead reduction with a consistent and significantly high prediction accuracy that validates its low latency and efficiency advantages.

ACKNOWLEDGEMENTS

In the name of Allah, the most gracious and the most merciful.

All praises belong to Allah and his blessing for completing this thesis work in fulfillment of my degree.

Aside from my efforts, the success of my thesis is heavily reliant on the support and guidance of many people. I would like to use this opportunity to show my gratitude to everyone who helped me finish my thesis work.

I am thankful to each of the following funding sources for their support of my research:

- Lakehead University Faculty of Graduate Studies;
- Lakehead University Faculty of Science and Environmental Studies;
- Dr. Zubair Fadlullah (Faculty Research Award)

I want to express my sincere gratitude and thanks to my supervisor, Dr. Zubair Fadlullah, for advising, mentoring, supporting, and guidance throughout my research work and academic advising. Dr. Fadlullah's organization, attitude, analyst skills, and new research directions inspired me and helped me to enter a new research area more easily which helped me achieve the interdisciplinary research work goals. I will be eternally thankful to him for his assistance in both my academic career and personal life during the journey of fulfilling the master's degree.

I express my gratitude to our research collaborators and co-authors for their participation and valuable contributions to our research. I would like to express my gratitude and appreciation to Dr. Mostafa Fouda, Assistant Professor at Idaho State University, for his invaluable insights and opinions that greatly contributed to my research in various stages. I am also grateful to all my lab members from the ACCESS Lab for the engaging research discussions. Additionally, I would like to thank the examiners, Dr. Al-Sakib Khan Pathan and Dr. Muhammad Asaduzzaman, for their valuable comments and constructive recommendations.

Lastly, I express my heartfelt appreciation and eternal gratitude to my dear family members, including my fiancée (special thanks for non-stop warm support and motivation), parents, and siblings, for their unwavering support, and continuous prayers throughout my academic and research journey, as well as in life in general. I am also thankful to my colleagues, friends, and everyone who has helped me or encouraged me to pursue my master's degree at Lakehead University.

PUBLICATIONS

Parts of this thesis have been submitted for peer review or published or to be submitted:

- **K. Bedda, Z. M. Fadlullah and M. M. Fouda, "Efficient Wireless Network Slicing in 5G Networks: An Asynchronous Federated Learning Approach."** is **published** in 2022 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), BALI, Indonesia, 2022, doi: 10.1109/IoTaIS56727.2022.9976007. (Chapter 4)
- **K. Bedda, M. M. Fouda and Z. M. Fadlullah , "PC-SSL: Peer-Coordinated Sequential Split Learning for Intelligent Traffic Analysis in mmWave 5G Networks"** is **Submitted** to IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. (Chapter 5)
- **K. Bedda, M. M. Fouda, and Z. M. Fadlullah, "Distributed learning Paradigms for channel quality estimation in LTE Networks" to be Submitted** to IEEE Global Communications Conference, December 2023. (Chapter 6)
- **K. Bedda, Z. M. Fadlullah and M. M. Fouda, "New Paradigms of Distributed AI for Improving 5G-Based Network Systems Performance" to be Submitted** to as a journal paper to IEEE Access. (All chapters)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

5G networks offer a substantial improvement in wireless communication, providing higher speeds and more stable connectivity than previous generations. As a result, they have emerged as a main driver of improved communication performance, driving more workloads to the edge and supporting low-latency data-driven use cases and applications [1]. 5G networks, with rates of up to 10 Gbps, enable hitherto hard-to-deploy applications and areas in smart communications, such as remote surgery, real-time virtual reality, and smart city infrastructure. High-speed and low-latency connectivity provided by 5G networks can transform various industries, including healthcare, transportation, and more. The advent of 5G technology has marked a significant milestone in communication network evolution, unlocking unprecedented opportunities for innovation. With unparalleled speed, minimal latency, and the ability to connect massive numbers of devices, 5G can revolutionize industries by enabling autonomous vehicles, facilitating the creation of smart cities, and powering the growth of the Internet of Things (IoT).

Aside from enabling new applications, 5G networks have been designed to support massive machine-type communication (mMTC) and ultra-reliable low-latency communication (URLLC), both of which are critical to the advancement of emerging technologies like autonomous vehicles and industrial automation. The mMTC functionality enables 5G networks to support a large number of connected devices, including IoT devices and sensors while consuming little energy and transmitting data at high speeds. This capacity is critical for the development of smart cities and other applications that gather and send data across a huge network of interconnected devices. Conversely, URLLC alludes to the capacity of 5G networks to furnish ultra-dependable, low-latency connectivity for applications necessitating immediate response times, such as autonomous vehicles and industrial automation [2]. By utilizing 5G networks, these applications can be operated in real-time from remote locations, hence fostering greater flexibility and efficiency in transportation systems and industrial processes.

Reducing operational expenses and achieving returns on network investments are the primary objectives of service providers who leverage AI. The growing intricacy of multi-layer and multi-band combined 5G networks has led to a surge in demand for ML-powered automation to help service providers maintain or decrease their operational costs [3].

Another important aspect of 5G networks is network slicing, a technique for creating virtual network slices with specific QoS requirements for different types of applications. Network slicing enables the efficient allocation of network resources, improves network performance, and enhances user experience (Zhang et al., 2021) [4]. With network slicing, service providers can offer tailored network services to different customers based on their specific requirements, such as low latency or high throughput. Network slicing also allows for flexible and dynamic allocation of network resources, essential for efficiently using network resources and supporting diverse applications [5].

In 5G networks, network slicing constitutes a critical technology that enables the creation of several virtual network slices across a singular physical infrastructure. Each slice is individually optimized for a specific application, granting customized connectivity with specific quality of service (QoS) requirements. The notion of network slicing was first introduced in 5G networks as a solution to address various use cases with varying QoS requirements, such as low latency for autonomous vehicles or high bandwidth for video streaming services. Network slicing facilitates the efficient allocation of network resources and enhances overall network performance by adapting to changes in network traffic and user demands in real-time. Network slicing is anticipated to be essential in developing 5G networks and implementing novel applications and services.

5G networks facilitate the integration of diverse connected nodes, including smart devices and sensors, to cater to a broad spectrum of applications. In order to ensure the seamless operation of low-latency applications and maintain the confidentiality of data, there has been a significant shift in mobile computing and 5G from centralized mobile cloud computing to mobile edge computing (MEC). The requirement to extract meaningful insights from vast amounts of decentralized data while maintaining data privacy has prompted the emergence of distributed machine learning on mobile edge. In recent years, considerable efforts have been invested in academia and industry to advance these technologies [3].

While 5G systems seek to provide high throughput and ultra-low latency communication services to improve users' QoE [6]. Enabling intelligent capabilities in 5G systems using deep learning is a costly affair due to the need for powerful hardware and software to support complex training and inference. However, there are several emerging tools that make deep learning feasible in mobile networks. These include advanced parallel computing, distributed machine learning systems, specialized deep learning libraries, fast optimization algorithms, and fog computing.

For the integration of machine learning in 5G network and MEC applications, there are two main approaches to consider: (i) decomposing the model to train or make inferences with its individual components, or (ii) scaling and parallelizing the training process to update the model at distributed locations linked to data containers. This emphasizes the significance of utilizing distributed machine learning methods [7].

Split learning has proven to be especially useful in applications such as throughput prediction, which is critical in optimizing network resource allocation and improving user Quality of Service (QoS). Throughput prediction involves estimating the data rate that a user can achieve at a specific time and location. The network can then dynamically allocate resources to provide optimal QoS while minimizing waste. Real-time accurate throughput prediction is crucial for supporting high-bandwidth applications like video streaming and online gaming that require consistent and reliable network performance. This privacy-preserving machine learning technique is effective in various network scenarios, such as edge computing, 5G networks, and Internet of Things (IoT) systems. Research in the literature on split learning for IoT networks found it to be superior to traditional federated learning methods while maintaining user privacy. Another study on split learning for 5G networks revealed that it improved the prediction accuracy of mmWave throughput while reducing computation costs compared to traditional machine learning approaches. The success of split learning in these network scenarios demonstrates its potential as a practical solution for privacy-preserving machine learning in networks.

Since we can conclude that 5G networks and their legacy systems are highly dynamic, the need for performance improvement in terms of spectral efficiency (.e.g., throughput, delay, signal strength, bandwidth). While a ton of research is done on machine learning deployment in 5G network systems, there is still a research gap in distributed learning considerations. Our main two objectives are 1) Improve the performance of 5G network-based systems by designing intelligent solutions built above the distributed machine learning framework to improve the spectral efficiency of the network in different topologies, particularly the traffic analysis scenarios. 2) Preserve the privacy of the network users while fulfilling the first objective. Through our work, we assume that all of the clients have similar resources or computational powers, and the complexity of the deep learning model is optimized towards the least participating client resources. The latter problem is discussed in chapter5, where the split learning model is designed with the motivation of this limitation.

1.0.1 Contributions

Our main contributions are:

- An asynchronous federated learning framework used in RSSI (Radio signal strength indicator) prediction, 5G network slicing, and Channel quality prediction use cases.

- A novel Peer coordinated sequential split learning framework used in mmWave 5G throughput prediction and Channel quality prediction use cases.

Chapter 2

Background

2.1 Preliminaries: 5G Network Fundementals

2.1.1 What Is New In 5G?

Radio waves have a limit to the amount of information they can carry based on the frequency bands they operate on. When this limit is reached, the allocation may become Pareto-optimal, which can negatively impact other users in the network (e.g., in 4G networks). However, 5G networks have increased bandwidth, which allows for a greater number of user devices and faster data transmission speeds, thereby improving network capacity. In addition to providing more capacity for existing tasks, 5G also opens up opportunities for innovative use cases, such as securely streaming high-quality video from a connected ambulance to a hospital. This technology enables a range of new types of smart devices and applications for digitization in various industries. To scale these applications, AI solutions are expected to play a critical role in managing the coordination of devices, radio, and compute resources.

The 5G network is engineered to link devices beyond just smartphones, with multiple types of connections tailored to the requirements of specific devices and applications. These connections include an energy-conserving option, ideal for low-power devices like smartwatches, as well as a highly stable and speedy option for industrial robots. Furthermore, network slicing can be implemented with 5G, allowing for the creation of distinct network slices tailored for a particular use case. Each slice can optimize its resources to meet the requirements of a specific service without unnecessary waste.

The 5G network offers innovative technologies that differentiate it from previous generations, with a particular focus on Mobile Edge Computing (MEC). One such technology is Software-Defined Networking (SDN), which simplifies network management and service deployment by dividing the control and data planes. The control plane is responsible for policies on the cloud, while the data plane evaluates whether to forward traffic based on the control plane policies [8]. Additionally, Network Virtualization Function (NVF) enables the execution of network functions on virtualization software located on servers, making it more flexible, automated, and scalable and avoiding overburdening the cloud with network functions [9]. Furthermore, massive Multiple-Input Multiple-Output (MIMO) enhances the signal-to-noise ratio without increasing transmission power, allowing for increased network capacity through the offloading of tasks to an edge server from a UE [10].

The next-generation wireless access technology, 5G NR [11], is an important enabler of MEC, which allows connectivity from different networks and devices, resulting in better scalability and lower latency. 5G also enables device-to-device communication through adhoc links, enabling communication between UE in close proximity without the need for the signal to go through the base station, which can help in reducing traffic congestion and improving network throughput [12].

To make the most of these new features of 5G, effective coordination and optimization of data transmission and compute distribution across the network are essential. Distributed AI offers a promising solution that utilizes both centralized and distributed resources, preserves data privacy, and optimizes network performance while addressing the added complexity associated with a range of use cases and applications.

2.2 Mobile Edge Computing

MEC plays a crucial role in implementing 5G networks and IoT and is widely regarded as the most effective way to deliver computation and communication resources to mobile devices [13]. The concept behind MEC is to run applications and processing tasks closer to the end user, at the edge of the network. MEC technology offers a flexible and agile approach to deploying applications while providing a distributed computing environment for hosting applications and services. The ability to store and process content in close proximity to cellular subscribers is especially important for real-time applications that require low latency.

Moreover, real-time radio access network information can be shared with applications, opening up new possibilities for future 5G applications. The combination of edge computing and 5G presents significant opportunities across various industries. It enables computation and data storage to be located near data sources, facilitating better data control, lower costs, quicker insights and actions, and uninterrupted operations. It is projected that by 2025, 75% of enterprise data will be processed at the edge, a significant increase from the current 10% [14].

However, The integration of MEC and 5G technology introduces various challenges. One such challenge is handling delay-sensitive data generated by UEs in real-time use cases. Edge servers can be utilized as intermediaries to efficiently transmit data and prevent significant delays from directly accessing the cloud. Accurately predicting network demand is another challenge, especially in real-time scenarios where the split between cloud offloading and local processing on edge cloud varies. Resource management is also challenging, as edge nodes lack the full computing capabilities of the cloud, and optimally allocating resources based on constantly changing application requirements is complex. To improve QoE and QoS, service providers must consider the different constraints and demands of users and use cases in different geographical locations. Device and communication heterogeneity is another challenge that requires complex, distributed systems to cater to the different entities. Despite reduced communication between UE and the cloud, privacy and security remain significant challenges due to the dynamic network requirements and the increasing number of heterogeneous devices. Comprehensive security management systems are necessary but may introduce overhead. These challenges may hinder the adoption of MEC in 5G and future networks. However, they provide a foundation for designing systems that can perform optimally by addressing the different requirements and constraints of various use cases [15].

2.3 5G Network Spectral Efficiency

5G spectral efficiency, also known as spectral efficiency or bandwidth efficiency, refers to the ability of 5G networks to transmit more data over a given amount of radio spectrum compared and is equivalent to the maximum number of data bits that can be transmitted to a specified number of users per second while maintaining a good quality of service. This is achieved through a combination of several technological advancements, including advanced antenna systems, higher-order modulation techniques, and more efficient coding schemes.

The spectral efficiency of 5G networks depends on several key parameters and components, including:

- Modulation: Modulation is the process through which information in a wireless transmission is encoded. All data sent through a wireless transmission is modulated in some fashion. A wider bandwidth is achievable with more complex modulation. One of the often employed modulation methodologies in 4G and 5G cellular systems is QAM, or Quadrature Amplitude Modulation, a high-bandwidth modulation that encodes data by adjusting the amplitude and phase of the signal. 5G networks use higher-order modulation schemes, such as 256-QAM, which allow more bits to be transmitted, increasing the amount of data that can be transmitted over a given spectrum [16].
- Coding: 5G networks use more efficient coding schemes, such as low-density parity-check (LDPC) codes, which enable more reliable transmission of data over the airwaves, even in challenging wireless environments.
- Multiple antennas: 5G networks use advanced antenna systems, such as massive multiple-input, multiple-output (MIMO), which enable more efficient use of spectrum by allowing multiple data streams to be transmitted and received simultaneously using employing hundreds or even thousands of antennas connected to a base station [17].
- Spectrum allocation: 5G networks use spectrum more efficiently by allocating it dynamically to different users and applications based on their individual needs, using techniques such as carrier aggregation and dynamic spectrum sharing.
- Network architecture: 5G networks are designed with a more distributed and flexible architecture, with small cells and network function virtualization (NFV), enabling more efficient spectrum use by reducing interference and optimizing network resources.

2.4 Preliminaries: 5G Network Slicing

Before delving into the details of 5G network slicing, we will introduce some key enabling technologies that the network slicing concept is built on.

- **Virtualization**: The first technology is virtualization, which enables slice creation of physical resources such as servers or network devices. Virtualization is an essential technology that allows multiple applications or services to run on the same physical infrastructure, improving resource utilization and reducing costs.
- **SDN**: The second technology is software-defined networking (SDN). Software-defined networking introduces network programmability by separating the control plane from

the data plane, and logically centralized network intelligence, which can be leveraged to create new network services by dynamically chaining various functions based on user requirements and network conditions [18] which enables more flexible and efficient network management. SDN allows a shared infrastructure to support multiple client instances efficiently. This is achieved by providing a complete abstract set of resources isolated by the SDN controller's virtualization, making it a natural fit for supporting network slicing. Using programmable interfaces and corresponding client-server contexts, the SDN controller can dynamically orchestrate resources for network slices belonging to the same context. Additionally, the recursive nature of SDN architecture allows for end-to-end federated or recursive network slices across different domains.

These technologies underpin network slicing, which enables the construction of virtualized, software-defined, and functionally optimized network segments. Network slices can be tailored to match the demands of specific applications or users, resulting in differentiated services and improved network performance overall.

To create a network slice, a network operator must define a set of characteristics that describe the slice, such as the quality of service (QoS) requirements, security policies, and network functions required. The operator can then allocate network resources such as bandwidth, processing power, and storage to the slice based on the characteristics defined.

Once a network slice has been created, it can be managed separately from other network slices, allowing different policies and configurations to be applied. Network slices can be created dynamically in response to changing network conditions or user demands and scaled up or down as needed.

Network slicing provides a more reliable and consistent user experience for users, as network resources are optimized for specific applications or services. It also provides greater flexibility, allowing users to choose the services and applications that best meet their needs.

However, network slicing also presents some challenges, particularly related to security and privacy. Because network slices are functionally separated, there is a risk that malicious actors could exploit vulnerabilities in one slice to gain access to other slices or the underlying network infrastructure. Network operators must therefore ensure that appropriate security measures are in place to protect against these risks.

Another challenge is the complexity of managing multiple network slices. Network operators must have the tools and processes in place to manage and monitor multiple slices, ensuring that they are performing as expected and that resources are being allocated appropriately.

5G network slicing enables more efficient and flexible use of network resources. While it presents some challenges, it has the potential to transform the way that mobile networks are managed and operated, providing greater flexibility, reliability, and differentiated services for both operators and users.

2.5 Preliminaries: 5G intelligence

This sub-section depicts the fundamental ideas of diverse AI-enabled methods exploited in the remaining chapters of the thesis for tackling challenges in different smart health use-cases. Firstly, we discuss the traditional machine learning-based techniques, and then we shed light on the neural network and deep learning techniques adopted in the thesis.

2.5.1 Fundamentals of Deep Learning Models

This sub-section illustrates some fundamental ideas behind Neural Networks (NN) and Deep Learning (DL). NN is a sub-category of machine learning, and the deep learning sub-field mainly originated from the NN as well. Neural networks and deep learning methods can be categorized into three categories such as supervised, semi-supervised, and unsupervised learning. Additionally, Reinforcement Learning (RL) is also considered another type of DL/NN approach. The NN-based deep learning approach is also called universal learning because we can apply it to almost any application domain [19]. The DL can be considered as a viable approach when we have a lot of data in hand to analyze. Therefore, these are appropriate techniques to solve diverse problems in the computing plane of IoT that generates a tremendous amount of data continuously. Among various available variations of the DL algorithms, in the following chapters, we have mainly utilized diverse custom Artificial Neural Networks (ANNs), and Convolutional Neural Networks (CNNs) in order to propose solutions to the selected problems in the mentioned 5G network use cases.

2.5.1.1 Dense Neural Networks

Artificial Neural Networks (ANN) are mathematical models that can be used to solve a variety of problems, such as pattern recognition, autonomous control, and smart healthcare, by leveraging a large amount of historical data to learn and improve over time. These models have been widely adopted by various organizations to tackle problems across different domains, with the economic sector being a key application area falling under the domain of operations research [20]. Like the biological neural network, the fundamental building block of an ANN is the artificial neuron or node, making it comparable to the biological nervous system in terms of its function as an information processing model. In summary, ANNs are a powerful tool for solving complex problems and can be thought of as a digital model of the human brain's information processing capabilities.

Dense neural networks, also known as fully connected neural networks, are a type of artificial neural network that consists of multiple layers of interconnected neurons. In a dense neural network, each neuron in a given layer is connected to every neuron in the previous layer, and every neuron in the subsequent layer is connected to every neuron in the current layer. This creates a dense or fully connected structure that allows for complex, nonlinear transformations of input data.

A dense neural network can be mathematically represented as a function that maps an input vector x to an output vector y. Let X be a matrix of size n x m, where n is the number of samples and m is the number of input features, and let Y be a matrix of size n x k, where k is the number of output classes. The output of a dense neural network can be computed as follows:

$$Y = f\left(W_2 \cdot f\left(W_1 \cdot X + b_1\right) + b_2\right) \tag{1}$$

where W1 and W2 are weight matrices of size m x p and p x k, respectively, b1 and b2 are bias vectors of size p and k, respectively, and f is an activation function that introduces nonlinearity into the network.

Another representation of the DNN assuming the output of the dense neural network is represented as a vector y of size m, and the weights and biases of the network are represented as matrices $W^{(1)}, W^{(2)}, \ldots, W^{(L)}$ and vectors $b^{(1)}, b^{(2)}, \ldots, b^{(L)}$, where L is the number of layers in the network, the equation for the output can be written as:

$$y = \sigma\left(W^{(L)}\sigma\left(W^{(L-1)}\sigma\left(\ldots\sigma\left(W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}\right)\ldots\right) + b^{(L-1)}\right) + b^{(L)}\right) \tag{2}$$

where x is the input vector of size n and σ is the activation function applied element-wise to the output of each layer.

The first layer of a dense neural network is the input layer, which receives the input data and passes it through to the first hidden layer. The number of neurons in the input layer is equal to the number of input features, while the number of neurons in the subsequent hidden layers can be adjusted based on the complexity of the problem being solved. The final layer of the network is the output layer, which produces the network's prediction for a given input.

The weights and biases in a dense neural network are learned through a process called backpropagation, which involves computing the gradient of a loss function with respect to the network's parameters and updating them accordingly. The loss function measures the difference between the network's output and the true labels, and the goal of training is to minimize this difference.

Fig. 2.1 illustrates a DNN model's high-level architecture. Dense neural networks (DNNs) are made up of three category layers: input, hidden, and output. The input layer usually has up to n nodes, where n denotes the number of data features. Between the input and output layers, more hidden layers can be inserted. For classification tasks, the output layer has k nodes, where k is the number of classes and typically just one node for regression or prediction tasks unless it's a multi-regression task, then the number of the nodes is the number of the predicted variables. The number of nodes in the hidden layers is random and

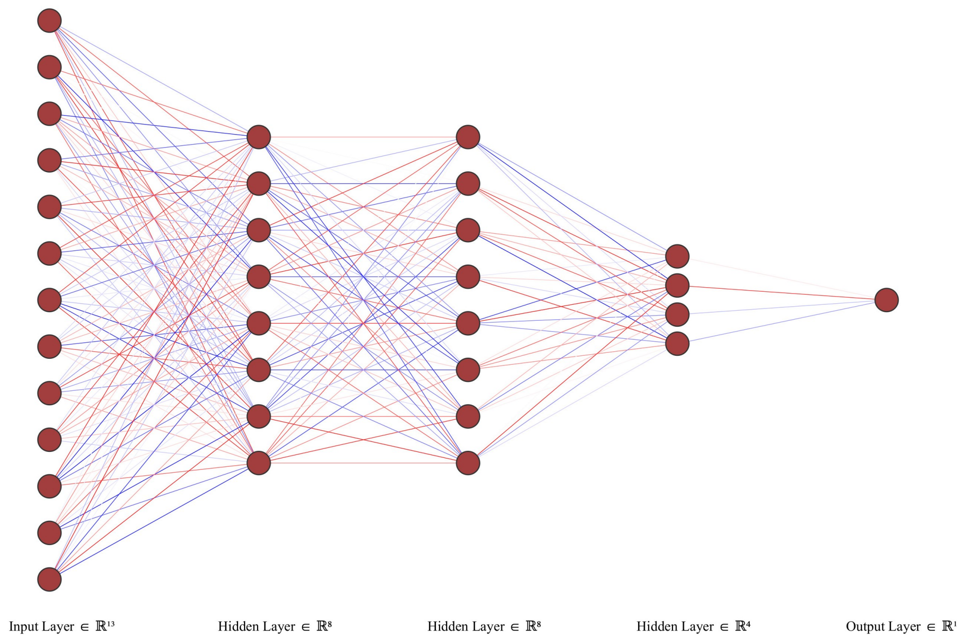

Figure 2.1: Architecture of a typical DNN model.

unrelated. The inputs are first weighted by the node's connection weights and then summed to determine a node's output. A bias component is added to this total, and the result is sent via an activation function, which brings non-linearity into the model. Various optimizers based on gradient descent are used to learn model parameters or weights. Backpropagation is used to change the weights until the learning phase is complete.

2.5.1.2 Convolutional Neural Network

Recently, one of the most popular categories of deep neural networks is the Convolutional Neural Network (CNN). CNN is a typical NN that is extended across space via sharing weights. Popular variations of the CNN are 1-D and 2-D CNN. The 1-D CNNs are generally customized to analyze tabular and time-series data, whereas the 2-D CNNs are widely adopted for pattern detection from image data. As 1-D CNN requires significantly less computational complexity than 2-D CNN, it is more suitable for lightweight real-time applications with limited resources, such as resource-hungry IoT sensors. A standard CNN has multiple layers: convolutional, pooling, regularization, and fully connected/dense. The dense layers and convolutional layers have parameters for training/learning; however, regularization layers and pooling do not have learned parameters. Various studies have concluded that CNN performs excellently in various AI tasks [21].

The typical architecture of CNNs consists of two fundamental parts: the feature extractor and the classification part. In the feature extraction layers, the previous layer's outcome is used as the input of a particular layer, and the result of that layer is then passed to the next layer. Usually, the lower-level or shallow layers are responsible for extracting higher-level features from the input data. With the propagation to the higher-level or deep layers, the dimensions of the features are usually decreased depending on the kernel size of the convolution. The max-pooling layers are also crucial for downsampling the features' dimensions, reducing the computation. As the sub-sampling layer, the max and average pooling are typically adopted to take the maximum and average values, respectively. After several convolutional and sub-sampling layers, fully connected/dense layers are usually used to conduct high-level reasoning. Like typical NNs, each neuron is connected to every other neuron in the dense layers to generate global semantic information. However, an atomic-dimensioned (i.e., 1x1 Conv) convolution layer can replace the fully-connected layer. The last layer of CNNs is the output layer. The softmax activation function is commonly employed for classification tasks, and for regression/prediction tasks, the linear and sigmoid activation function is applied chiefly [22].

In order to design a robust CNN model, we need to determine optimal values for different hyperparameters. Typically, in different layers of the CNN architecture, we need to define a number of hyperparameters before the learning phase begins. Firstly, we have the number of layers used in the CNN model, which determines how deep or shallow a particular model is. In the convolution layers, typically, the kernel size and the number of filters are the hyperparameters. The hyperparameter of the sub-sampling layer is usually the pool size for reducing the size of features. The regularization layer (i.e., dropout layer) is often used in the CNN architecture to avoid overfitting. The dropout rate is the general hyperparameter in the dropout layer. Other than these hyperparameters, we also have a few other vital hyperparameters needed in the CNN model, such as the activation function, optimizer, learning rate, batch size, and the number of epochs [23]. By finding the appropriate values for the diverse set of hyperparameters of the CNN model, we can control the trade-off between the performance (i.e., accuracy) and computation burden (i.e., memory and time requirements) of the resource-constrained IoT devices in order to analyze health data. Hence, considering this trade-off, in the following chapter of the thesis, we have emphasized the hyperparameter optimization for constructing suitable custom CNN models to analyze medical data and efficient health data transmission.

2.5.2 Distributed Deep learning approaches on edge

The use of distributed machine learning (ML) on edge is being extensively researched as a more efficient approach for training deep learning models [24]. The main motivation behind this is the potential for an enhanced model that incorporates data from various sources while ensuring security, and minimizing communication, and computation overhead. There are two primary distributed ML methods: Federated Learning (FL) and Split Learning (SL).

2.5.2.1 Federated Learning (FL) architecture

Federated learning is a distributed machine-learning technique that allows multiple devices to collaborate in the training of a machine-learning model, without exchanging raw data. In the deployment of federated learning, a central server first selects a machine learning model and distributes it to multiple devices, such as smartphones or IoT devices. Each device then trains the model on its own data, using privacy-preserving computation techniques to ensure that the data remains private. The updated model parameters are then transmitted back to the central server, aggregating them to update the global model. This process is repeated for multiple rounds, to improve the accuracy of the model over time.

Federated learning enables machine learning models to be trained on distributed data, without compromising the privacy of the data. While it presents some technical challenges, it has the potential to transform the way that machine learning is performed, particularly in scenarios where data privacy is a concern. As federated learning continues to evolve, it is likely to become an increasingly important technique for enabling distributed machine learning applications in a range of domains, including healthcare, finance, and IoT.

Algorithm 1: FedAvg Algorithm

```
  Input  : $\mathsf{C}, \mathsf{K}, \mathsf{E}, \mathsf{B}$
  Output: $w_{fed}$
1 Function FedAvg($\mathsf{C}, \mathsf{K}, \mathsf{E}, \mathsf{B}$):
2     $w_{fed} \leftarrow$ initialize model weights; for $round = 1$ to $\mathsf{E}$ do
3         m $\leftarrow$ min($\mathsf{K}, |\mathsf{C}|$); $\mathsf{B}_t \leftarrow$ random m clients from $\mathsf{C}$; S$t \leftarrow$ empty list; for $i = 1$
           to $|\mathsf{B}(t)|$ do
4             w$i \leftarrow$ get current weights of client $i$ from server;
               g$i \leftarrow$ compute gradient using local data on client $i$ with weights w$i$;
               S$t \leftarrow$ append (w$i$, g$i$) to S$t$;
5         end
6         g $\leftarrow$ average of gradients in S$t$; $w_{fed} \leftarrow w_{fed} - \frac{\mathsf{B}}{|\mathsf{C}|}$g;
7     end
8     return $w_{fed}$;
```

Algorithm 1 shows the Federated Averaging algorithm as follows: Initialize the model weights to some initial values. For a certain number of communication rounds, do the following: a. Select a random subset of clients (i.e., devices) from the set of all clients. The size of this subset is the minimum between the total number of clients and a predefined number. b. For each client in the selected subset, do the following: i. Get the current model weights from the server. ii. Compute the gradient of the model using the local data on the client with the current weights. iii. Add the pair of model weights and gradients to a list. c. Compute the average of all the gradients in the list. d. Update the model weights by subtracting a fraction of the average gradient from the current model weights. The fraction is determined by the size of the subset of clients relative to the total number of clients. Return the final model weights.

2.5.2.2 Split Learning architecture

Split learning is a distributed machine learning algorithm that enables models to be trained on data distributed across multiple devices where the model is split between a client and a server. The algorithm involves splitting the model into two parts: a front-end and a back-end. The front end is distributed to each device, where it processes the data locally and generates intermediate representations. These intermediate representations are then sent to the back end, which combines them to generate the final output.

The split learning algorithm consists of several steps. The first step is data partitioning, in which the data is divided into smaller subsets, and each subset is assigned to a specific device. This ensures that the data remains on the device where it was generated, and reduces the amount of data that needs to be transmitted between devices.

The second step is model splitting, in which the machine learning model is divided into two parts: a front-end and a back-end. The front end is distributed to each device, where it processes the data locally and generates intermediate representations. The back end is located on a central server and receives the intermediate representations from all devices to generate the final output.

The third step is training, in which the model is trained iteratively using a gradient descent algorithm. During each iteration, the front end processes the data on each device to generate intermediate representations, which are then transmitted to the back end. The back end combines the intermediate representations and calculates the gradient of the loss function. This gradient is then transmitted back to the front end, where it is used to update the model parameters.

The fourth step is synchronization, in which the model parameters are synchronized between the front end and the back end to ensure that they are consistent with each other. This involves transmitting the updated model parameters from the front end to the back end and then distributing the updated parameters back to each device.

The split learning algorithm is particularly useful in scenarios where data privacy is a concern, as it enables data to be kept on the device where it was generated rather than being transmitted to a central server. Additionally, split learning can be used to reduce the computational and communication costs associated with training machine learning models, as it distributes the computation and communication load across multiple devices.

The split learning algorithm can be broken down into the following steps as shown in algorithm 2:

- Partition of the model: The first step is to partition the neural network model into the front-end and back-end. The front end is part of the model that processes the data on the device, while the back end is part of the model that combines the intermediate representations from all the devices.

- Distribute the front-end: The front-end is distributed to the devices that hold the data. Each device processes the data locally using the front end, generating intermediate data representations.

- Transmit intermediate representations: The intermediate representations are transmitted from the devices to the back end. The transmission can be done in a compressed form to reduce the bandwidth required.

- Combine intermediate representations: The back end combines the intermediate representations from all the devices to generate the final output. This is done using the back-end part of the neural network model.

- Update the model: The model is updated using the back-propagation algorithm. The gradients are computed on the back end and transmitted to the devices, which are used to update the front-end part of the model.

Algorithm 2: Split Learning Algorithm

```
   Input   : $X, Y, \mathsf{theta}, \mathsf{alpha}, \mathsf{eta}$
   Output: $\mathsf{theta}$
1  Function SplitLearning($X, Y, \mathsf{theta}, \mathsf{alpha}, \mathsf{eta}$):
2      $\mathsf{theta}_1 \leftarrow$ front-end of $\mathsf{theta}$; $\mathsf{theta}_2 \leftarrow$ back-end of $\mathsf{theta}$; for $i = 1$ to $N$ do
3          $\mathsf{theta}_1^{(i)} \leftarrow$ initialize front-end on device $i$;
4      end
5      for $t = 1$ to $T$ do
6          for $i = 1$ to $N$ do
7              $X_i \leftarrow$ data on device $i$; $Y_i \leftarrow$ labels on device $i$; $h_i \leftarrow \mathsf{theta}_1^{(i)}(X_i)$;
               $z_i \leftarrow \mathsf{theta}_2(h_i)$; $g_i \leftarrow \nabla_{z_i} L(Y_i, z_i)$; $\nabla_{\mathsf{theta}_2} L(Y_i, z_i) \leftarrow g_i$;
               $\nabla_{h_i} L(Y_i, z_i) \leftarrow \mathsf{theta}_2^T g_i$; $\mathsf{theta}_1^{(i)} \leftarrow \mathsf{theta}_1^{(i)} - \mathsf{eta} \nabla_{\mathsf{theta}_1^{(i)}} L(Y_i, h_i)$;
8          end
9          $\mathsf{theta}_1 \leftarrow$ average front-ends on all devices;
           $\mathsf{theta}_2 \leftarrow \mathsf{theta}_2 - \mathsf{alpha} \nabla_{\mathsf{theta}_2} L(Y_i, z_i)$;
10     end
11     return $\mathsf{theta}$;
```

Chapter 3

Decentralized robot wireless connectivity estimation in indoor and outdoor environments using Asynchronous Federated learning

3.1 Introduction

This chapter investigates the RSSI prediction for rapid and automated wireless networking in indoor and outdoor environments. The performance of sensor nodes in a wireless network is determined by the quality of the network's essential links. Traditionally, essential connections are detected via time-consuming on-field RSSI measurements. We investigate a distributed ML technique for an approximation but quicker and communication-wise more efficient RSSI prediction in this research.

In typical models, an empirical or deterministic technique is employed to forecast environment-related route loss [25]. Empirical models often comprise a set of equations generated from extensive field data. In contrast, site-specific deterministic models employ physical principles of radio propagation to forecast signal intensity or path loss at a given place. An example of an empirical model is a path loss log distance model, which estimates the received power at each receiver location based on the distance between the transmitter and the receiver. However, these measures do not hold the assumptions in practice and produce disappointing findings [26]. It is always dependent on physical disturbances and unseen environmental factors.

In this context, we are exploring the possibility of using Machine Learning (ML) concepts to predict Received Signal Strength Indication (RSSI) and estimate the path loss exponent in wireless network deployments. Specifically, we are concentrating on using ML for RSSI prediction by treating it as a supervised learning problem. To achieve this, we employed a dataset that includes distance-based input features and measured RSSI values as the output. We utilized two ML approaches - a simple linear regression model and an Artificial Neural Network (ANN) model.

It has been observed that Machine Learning (ML) techniques have achieved remarkable success in wireless systems, especially in terms of the accuracy of the models. Our approach is to do more performance improvement by focusing on providing a distributed learning framework to have lower latency and maintain high reliability and high prediction accuracy.

3.2 Problem Description and Motivation

We need to develop a model for wireless connectivity prediction since it's vital in many applications like the teleoperation of robots. There are several deterministic models for indoor robot positioning; the most popular one is the path-loss exponent model. To consider the non-deterministic part of the model, we use the Shannon-Hamilton theorem [27], which indicates that the RSS can be modeled as in eq1

$$RSS = RSS_{d_0} - \underbrace{10\eta \log_{10}(\frac{d}{d_0})}_{deterministic} - \underbrace{\Psi(d) - \Omega(d,\ t)}_{stochastic}; \tag{1}$$

Where RSS_{do} is the RSS at a reference distance (d0),η is the propagation constant of the environment, d is the distance of the receiver from the radio source, Ψ is a stochastic (gaussian) variable representing (spatial) shadowing effects caused by the objects in the environment, and Ω is another stochastic variable in the RSS representing (spatial and temporal) multipath fading effects and dynamics in the environment [28]. On the other hand, the complexity of the environment and path factors in the outdoor environment makes it challenging to formulate a mathematical model for it. We are motivated by this problem and by the idea that indoor modeling still suffers from high variation in parameters with any change in the setup environment and has a stochastic part that has a complex pattern to model. All of these challenges could be addressed by using data-driven deep learning models, which are capable of capturing complex patterns. An additional challenge is that changing the environment will change its deterministic model of it. So we need a distributed solution to learn from different environments and different setups at the same time while perserving privacy and reducing the overhead in network communication. So we propose an asynchronous federated learning framework.

3.3 Methodology

In this section, we present our proposed methodology based on the system model assumption of RSS for indoor and outdoor wifi environments. In the remainder of this section, we elucidate the dataset, the deep learning model leveraged for both local and global model training, and our proposed federated learning framework with asynchronous weight updating capability.

3.3.1 Data

We use the dataset in [29]. This dataset provides the RSSI (Radio Signal Strength Indicator) data collected with a mobile robot in two environments: indoor (KTH) and outdoor (Dortmund). RSSI metric was used to collect the RSS data in terms of dBm. The mobile robot's location was recorded using its odometry (dead reckoning). The indoor data was collected at Stockholm. The outdoor data was collected at an abandoned steel factory in Dortmund, Germany. The RSS was collected using the RSSI metric in dBm. The mobile robot data such as position and orientation was collected with the help of the Robot Operating System (ROS) drivers of the respective robot.

3.3.2 Selection of Centralized Deep Learning Model for Network Slice Prediction

Here, we describe our selection of the centralized deep learning model for RSSI prediction given the considered system model. Among a number of candidate machine and deep learning models, we select the Convolutional Neural Network (CNN) for their recent success in communication networks. A CNN is a feed-forward neural network architecture with convolutional layers, pooling layers, and fully-connected layers. Due to the format of the RSSI data prepared (as mentioned in section 6.3.1), we use a one-dimensional Convolutional

Neural Network (1D CNN) [30]. The detailed motivation behind choosing the 1D CNN for proactively predicting network traffic and accordingly the corresponding network slices can be found in [31, 32].

3.3.3 Proposed Federated Learning Model with Asynchronously Weight Update Capability

Federated learning was first introduced in [33] where local models are trained on different distributed points. The centralized model is the aggregation of these local models by averaging them. However, it lacks efficiency due to stale local nodes. Then, the asynchronously weight updating federated learning approach was introduced in earlier work to address this problem [34, 35]. Motivated by the performance of our earlier work, we propose our asynchronously weight updating federated learning framework for privacy-preserving, low-latency network slicing. Our proposed framework resides in two separated operations, namely training on the clients' side and training on the central node (e.g., a server or a cloud), respectively.
The way the asynchronous federated update works is by sending only the shallow layers parameters and every certain number of time rounds it sends all the network parameters which is called deep parameter update. The ratio of the shallow layers out of the total neural network and the number of time rounds are both considered hyperparameters.

Algorithm 11 demonstrates the steps of the update and training in our proposed framework. First, an update on the set of all available clients is received by the central node. Then, the central node returns the pre-trained central model, denoted by M_c. Step 1 of the algorithm initializes the parameters and hyperparameters of our considered 1-D CNN model. In the following steps, the set of all clients is traversed, and we aggregate all of the local parameters to form the centralized model and feedback to the clients. Then, we assess the model's performance and compare it with the minimum loss threshold. The training process at the centralized model continues running until the loss is less than or equal to the threshold.

The second part of the algorithm entails the procedure on each client's side. This procedure works with the inputs u, δ, T, and λ. λ denotes the shallow parameter update ratio. The parameter is $timestep_u$ represents the iteration information at the deep model parameter exchange step with the central node. First, the hyperparameters of the local model (M_u) are initialized. The algorithm iteration starting afterward indicates that the overall training process at each local node (i.e., every client) occurs from iterations $t = 1$ to $t = T$, where the parameter T denotes the number of time steps during which the entire process for each client runs. The local model for each client is then trained. After δ time-rounds, the ratio of the locally learned parameter, according to the shallow update rate and time-round values, is sent to the central node when this condition remains valid.

Algorithm 3: Proposed asynchronously weight updating federated learning algorithm ofr network slice allocation.

```
Input: Distributed clients at subnetwork points, deep parameter rate $\lambda$
       $(0 < \lambda < 1)$, minloss , $\delta$ (time round), T (total number of iterations)
Output: $M_c$: Centrally trained model
Procedure at the central:
$M_c \leftarrow \theta$;
while True do
    for $u = 1$ to len(Clients) do
        Update $M_c$ by aggregating the recieved parameters ($W_u$) from Clients
    end
    $Current_{acc} \leftarrow$ current performance of global model $M_c$;
    if $(loss_{curr} < loss_{min})$ then
        break
    end
end
return $M_c$
Procedure at the client:
Initialize: Hyperparameters
for (t=1 to T) do
    $M_u \leftarrow$ update the model of $Client_u$ by adopting the model hyperparameters
    if $(t \bmod \delta) = 0$ then
        t is assigned to $timestep_u$
        $W_u \leftarrow$ extract local weights of $\lambda$ from $M_u$ transmit $W_u$, and $timestep_u$ to
          the cloud.
    else
        $W_u \leftarrow$ extract local weights of (1-$\lambda$) from $M_u$ transmit $W_u$, and $timestep_u$
          to the cloud.
    end
end
```

3.4 Performance Evaluation

To evaluate the performance of our proposed asynchronous weight updating federated learning framework, we first compare it with the synchronous weight updating (the vanilla federated learning) model performance with varying numbers of clients. In Algorithm 11, we removed the condition of (t mod δ) at the clients' side procedure to derive the synchronous weight updating condition. Second, we evaluated the deep learning model performance by monitoring the loss against various clients over several time rounds. In addition, we compared the standalone, centralized model's performance with that of our proposal, i.e., the client's model aggregation. Third, we evaluated the performance of our asynchronous weight updating strategy by monitoring the deep learning model loss with different time rounds. Finally, we assessed the overhead model reduction, time-round values, and deep

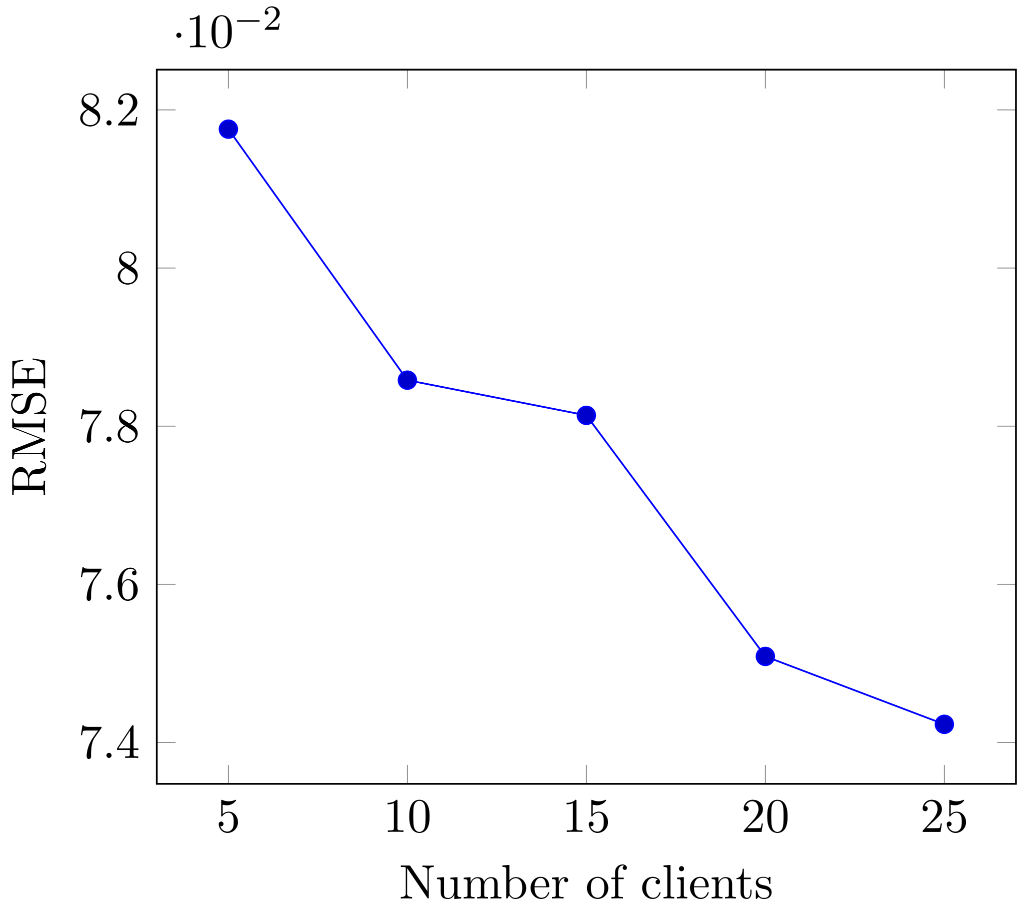

Figure 3.1: Loss performance (convergence) for varying numbers of clients (UEs) in case of the proposed asynchronous weight updating method (Indoor environment).

parameter rates.

First, in Figs. 3.1 and 3.3, the performance of our proposed asynchronously weight updating federated learning technique in terms of loss with an increasing number of clients for indoor and outdoor data respectively. On the other hand, Figs. 3.2 and 3.4 show the loss performance of the traditional method (synchronously weight updating mechanism) for various UEs. As evident from these results, our proposed asynchronous weight updating method approximates the performance of the conventional method with an increasing number of users. In summary, the performance of the proposal steadily improves with the number of clients.

Next, Figs . 3.5 and 3.6 demonstrate the performance of our proposal in terms of loss over varying time-rounds. The figure shows that the model performance increases with the number of time rounds. A time-round value determines after how many iterations the model will make a deep update instead of a shallow layer update of the local deep learning model. For example, if the time-round value is five, a deep layer parameter update occurs

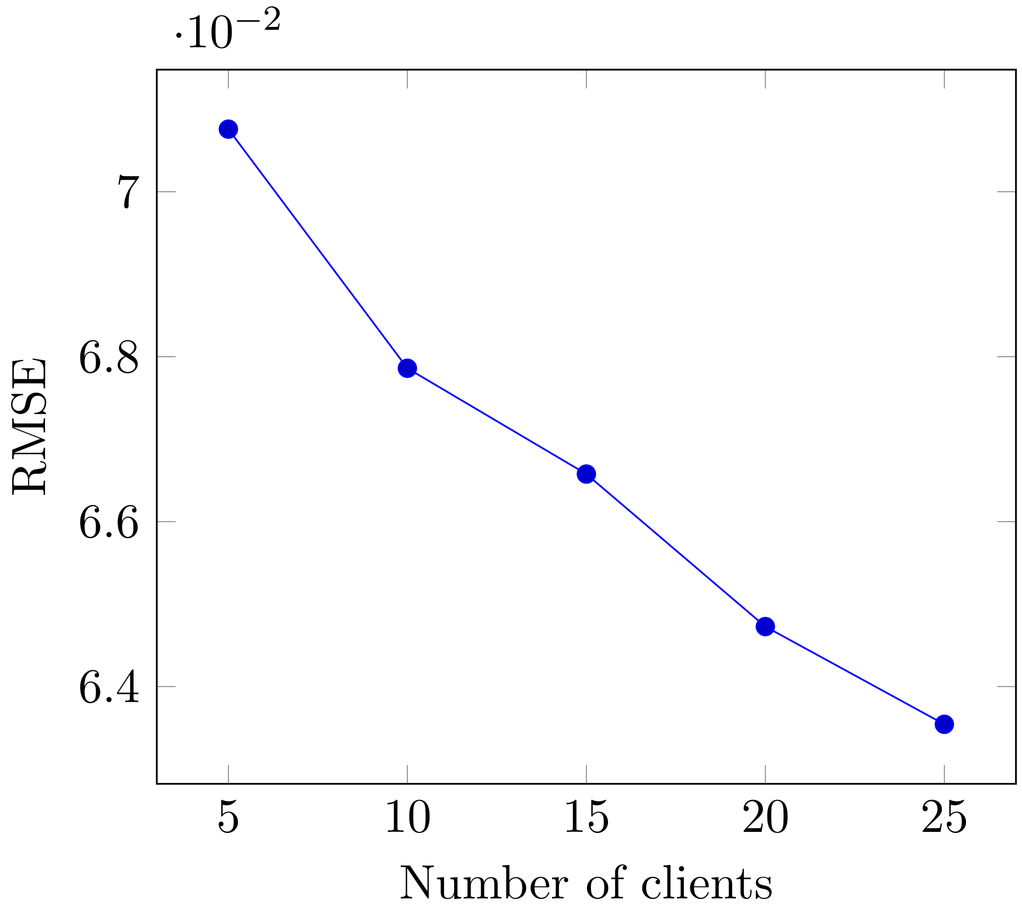

Figure 3.2: Loss performance (convergence) for varying numbers of clients (UEs) in case of the traditional synchronous weight updating method (Indoor environment).

in place every five iterations and the shallow update occurs in the remaining iterations. That means a higher value of the time-rounds provides a larger focus on the shallow layer updates. This result shows that the model performs better with more shallow updates, focusing on the general features learned on the clients' side more than the model-specific features. Furthermore, experimental results demonstrate that the aggregated model training performance over the epochs is comparable with the centralized model, thereby indicating the reliability of our proposal in proactive slice-type allocation.

Finally, Figs. 3.7 and 3.8 demonstrate the overhead reduction (with the network bandwidth usage percentage as a performance indicator) for varying deep parameter update rates and different time-round values for each parameter. The result illustrates that the increase in the deep learning parameters significantly improves the overhead reduction for the considered time-rounds of 5, 10, 15, and 20.

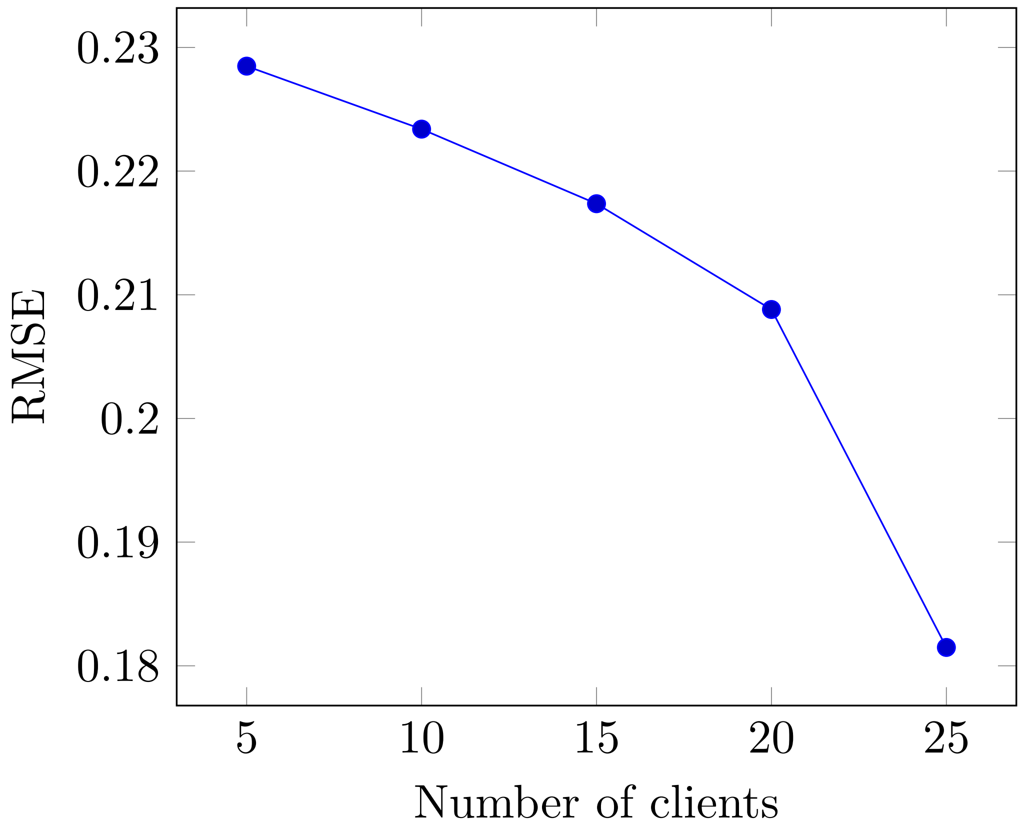

Figure 3.3: Loss performance (convergence) for varying numbers of clients (UEs) in case of the proposed asynchronous weight updating method (Outdoor environment).

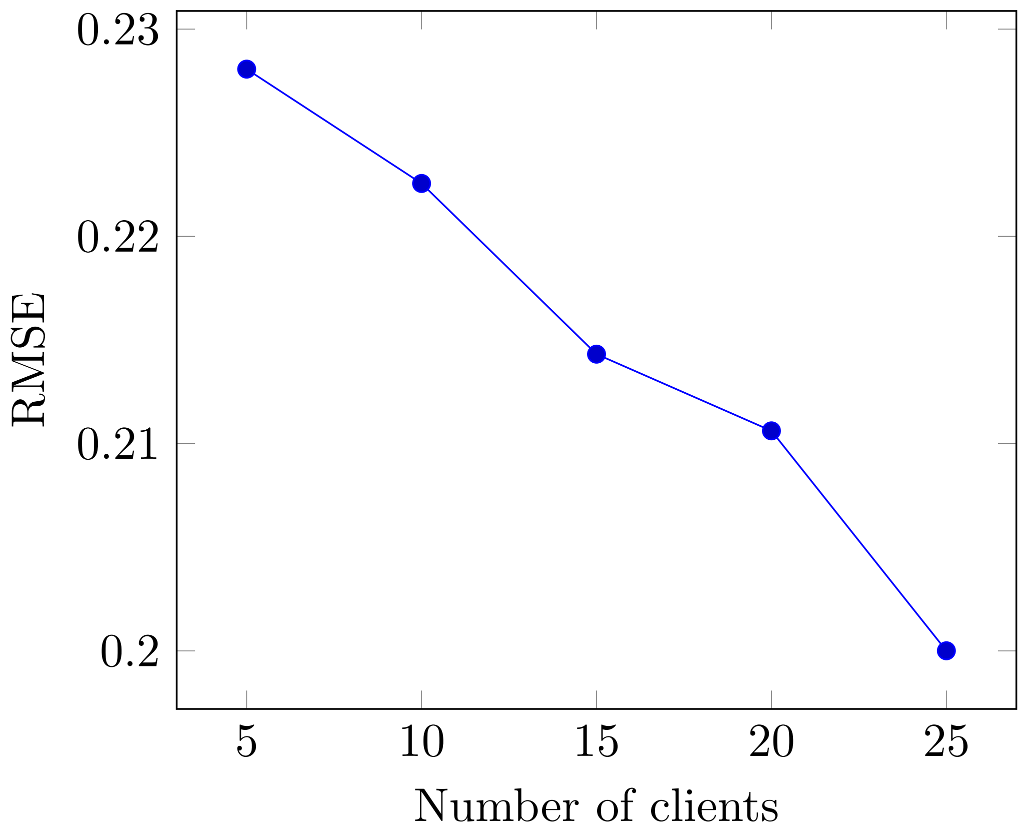

Figure 3.4: Loss performance (convergence) for varying numbers of clients (UEs) in case of the traditional synchronous weight updating method (Outdoor environment).

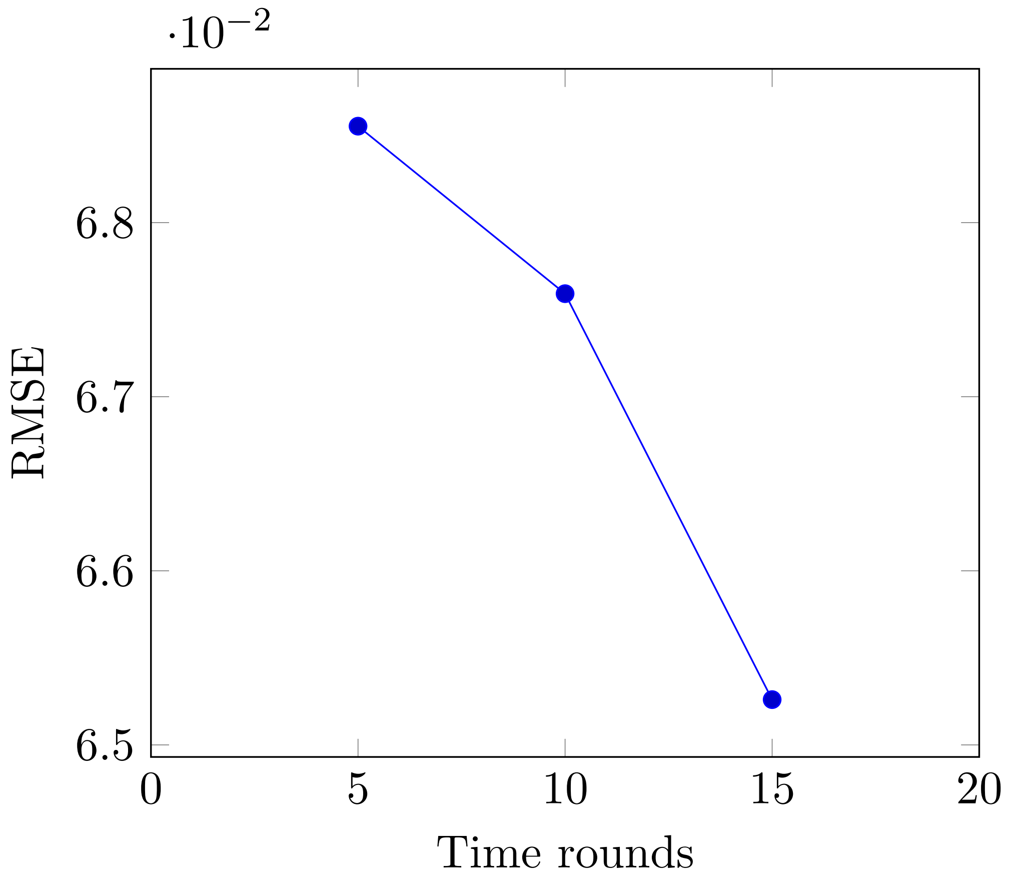

Figure 3.5: Loss vs number of time-rounds in the proposed asynchronously weight updating federated learning framework (Indoor environment).

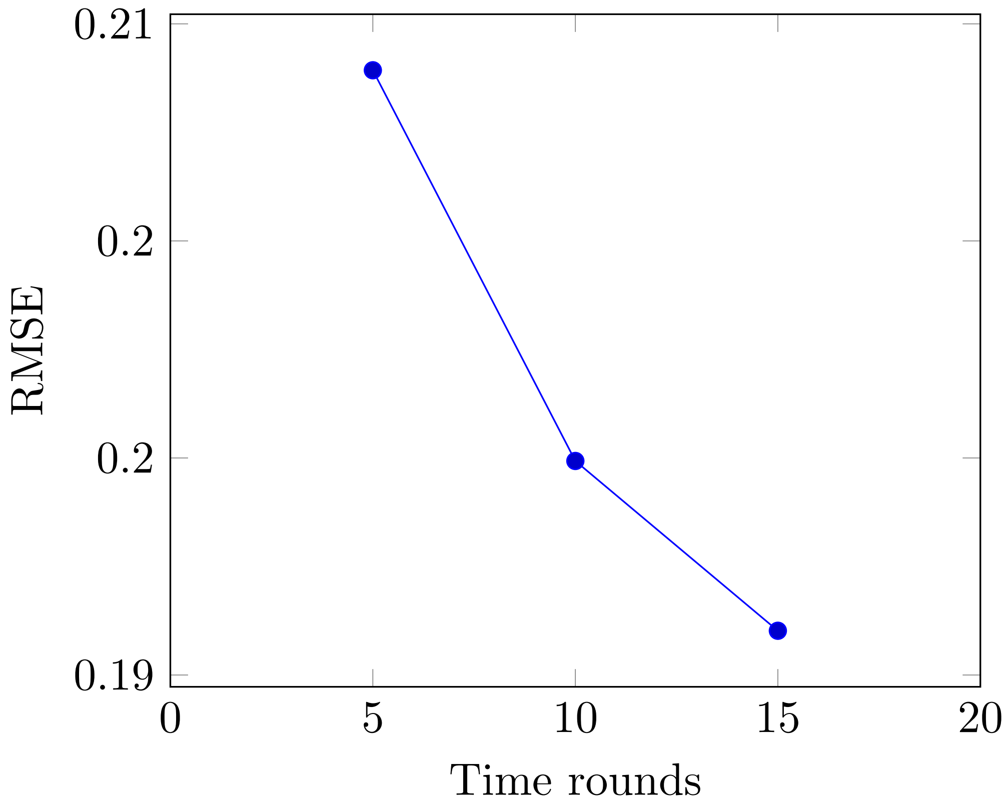

Figure 3.6: Loss vs number of time-rounds in the proposed asynchronously weight updating federated learning framework (Outdoor environment).

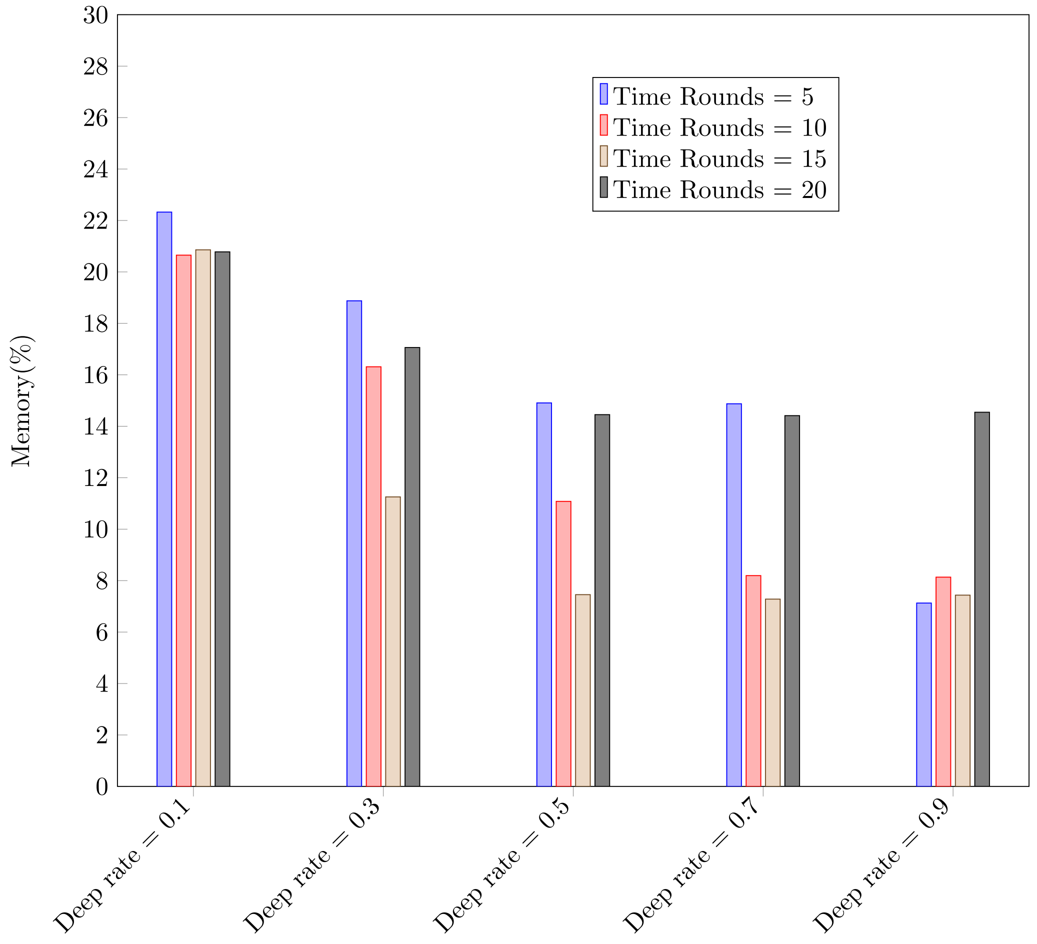

Figure 3.7: Overhead reduction ratio on memory used for different deep parameters rates each along with different time rounds values (Indoor environment).

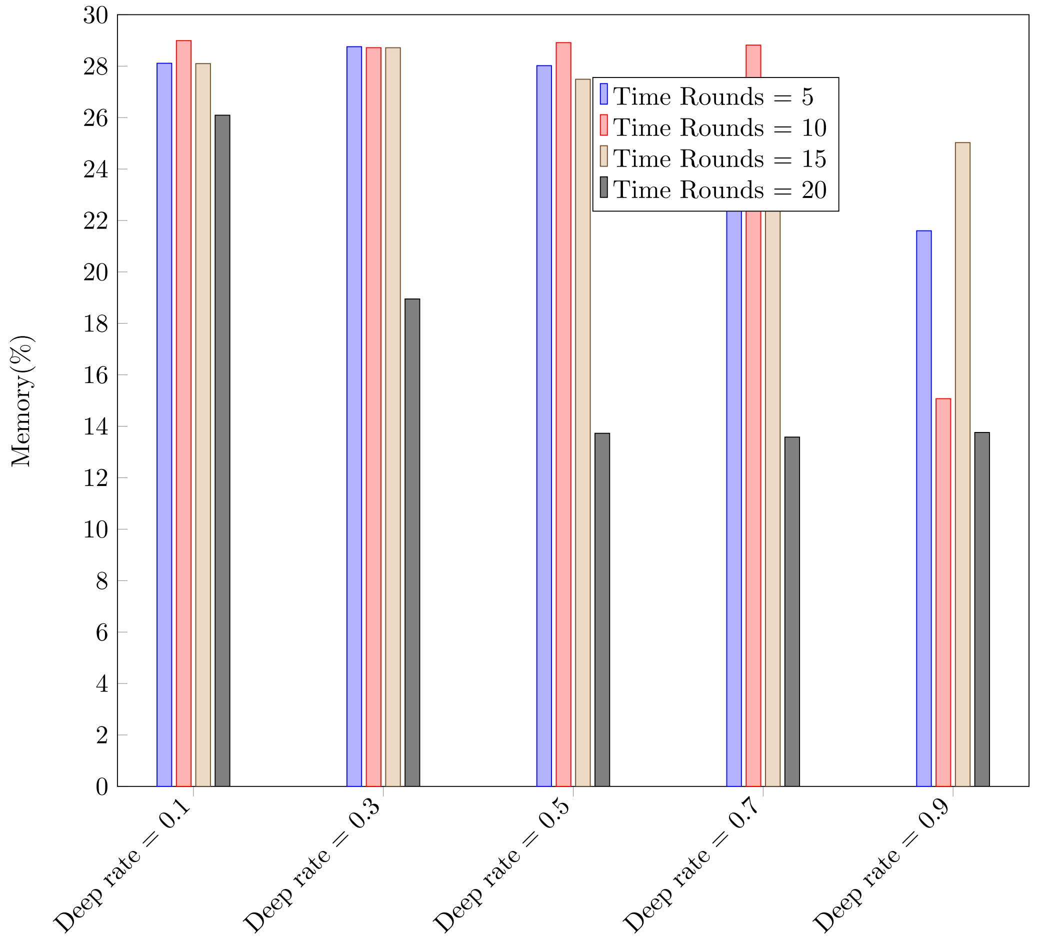

Figure 3.8: Overhead reduction ratio on the memory used for different deep parameters rates each along with different time rounds values (Outdoor environment).

Chapter 4

Efficient Wireless Network Slicing in 5G Networks: An Asynchronous Federated Learning Approach

While researchers continue to incorporate intelligent algorithms in Fifth Generation (5G) and beyond networks to achieve high-accuracy decisions with ultra-low latency and significantly high throughput, the issue of privacy-preservation became a critical research area. This is because mobile service providers not only need to satisfy the Quality of Service (QoS) of users in terms of ultra-fast user connectivity but also ensure reliable, automated solutions that will enable them to design a vast multi-tenant system on the same physical infrastructure while preserving the user privacy. With the adoption of data-driven machine learning models for providing smart network slicing in 5G and beyond networks and Internet of Things (IoT) systems, the issue of privacy-preservation integration is yet to be considered. We address this issue in this chapter, and design an asynchronous weight updating federated learning framework that is efficient, reliable, and preserves the privacy as well as achieve the required low latency and low network overhead. Thus, our proposal permits a reasonably accurate decision for the resource allocation for different 5G users without violating their privacy or introducing additional load to the network. Experimental results demonstrate the efficiency of the asynchronous weight updating federated learning in contrast with the conventional FedAvg (Federated averaging) strategy and the traditional centralized learning model. In particular, our proposed technique achieves network overhead reduction with a consistent and significantly high prediction accuracy that validates its low-latency and efficiency advantages.

4.1 Introduction

Given the performance of the federated learning framework that we proposed in the previous chapter 3, in this chapter we deploy the framework on an essential part of the 5G networks to improve the efficiency of network slicing in 5G networks.
Recently, the Fifth Generation (5G) and beyond networks have contributed to an exponential growth in mobile traffic that has played a dominant role in realizing the much anticipated ubiquitous connectivity toward a smart society. While 5G and beyond networks have been associated with a significantly higher network capacity and throughput [36] in contrast with the earlier generations of communication networks, the required level of intelligence to automate resource allocation decisions is a desired (yet still not available) feature in such systems. However, service providers frequently confront management challenges to optimally and timely satisfy the load demands of the massive number of users (referred to as user equipment or UEs) on multi-tenant networks. While the 5G and beyond networks have proliferated mobile broadband supporting high data rates and reduced delay, thereby promoting real-time services with high Quality of Service (QoS) performance and innovative applications with stringent throughput and latency requirements [37]. However, since 5G networks are envisioned as multi-service, multi-tenant networks, a larger eco-system with intelligent (i.e., proactive) decision-making needs to be designed. For instance, in order to offer a better Quality of Experience (QoE) to mobile UEs along with high speed and reliability, it is important to take into consideration the network traffic requirements and develop a new mechanism, i,e., a smart framework, to make proactive and accurate decisions pertaining to wireless resource allocation in 5G networks.

As for the 5G resource allocation, network slicing is an important technique that enables mobile operators to effectively manage a plethora of network instances across a single infrastructure to provide a variety of services to wide and multiple UE-types with a high level of QoS [38]. In this vein, an appropriate network slice selection for each UE device is required. Formulating the network slice selection is typically an optimization problem, which is classically solved by a centralized entity in the mobile network system. However, finding an optimal solution to such a problem is often not possible, and often heuristics are employed to obtain faster solutions at the expense of sacrificing the quality of the solution. To remedy the solution, machine, and deep learning algorithms can be leveraged to develop data-driven models to enable mobile operators reasonably accurate network slice predictions at real-time, thereby resolving the trade off between the solution quality and the time required to obtain the solution. In other words, the machine/deep learning paradigm may assist in the network slice prediction, based on which the resources required by the UEs can be accurately assigned.

However, with the increase of the number of UEs with time and also the variety of services used by the UEs within the 5G network ecosystem, we need the deep learning model to be up-to-date. This poses two unique challenges. First, the privacy of the UEs needs to be preserved while we collect data from all users to train the deep learning model. Second, such a consistently changing model with a growing size of data on a large scale will cause more load on the network itself, thereby significantly affecting the model decision time and in turn, impacting the overall network latency.

In this chapter, we aim to address the aforementioned challenges by proposing a federated learning framework whereby the local parameters are updated in an asynchronous manner. Thus, we facilitate a decentralized deep learning model that efficiently predicts the network slice type with high accuracy, high speed, and reliability and maintains the UE privacy simultaneously.

The remainder of this chapter is structured as follows. In section 5.2, we introduce the existing research work in two categories, namely, network slicing in general and network slicing with federated learning. Section 5.3 presents the problem formulation and the motivation behind our proposal. Section 6.3 presents our proposed asynchronously federated learning algorithm. The performance of our proposal is evaluated in section 6.4. In section **??**, we discuss the conclusion of our work contribution and future work.

4.2 Related Work

Several works are done in the literature on network slicing to ascertain the assignment of devices to connections properly. In [39], researchers addressed the resource allocation in a sliced multi-tenant network by determining how the capacity of the Mobile Virtual Network Operator (MVNO) system is affected by the transmit power, allocated bandwidth, and the number of users. Researchers in [40] suggested a methodology for managing network traffic priorities for smart cities using the software-defined network (SDN) architecture, where services that require priority are placed in virtualized networks, and the technique was demonstrated through a priority management layer in the SDN architecture. The work in [41] introduced a new management architecture for 5G service based on SDN/NFV (network function virtualization) to provide a distributed deployment of network slicing that meet the network functions demands. On the other hand, the research work in [42] considered a Radio Access Network (RAN) slicing deep learning architecture in order to apply application-specific radio spectrum scheduling.

While the aforementioned research work considered network slice management in a classical manner, other researchers dedicated a substantial effort toward intelligently assigning network slices using deep learning techniques. For instance, by considering deep learning for network slicing, researchers in [43] designed a model called DeepSlice to achieve network load efficiency and network function availability via accurately predicting the network slice type to assign user-devices to appropriate links. However, the DeepSlice model is considered a centralized learning environment. On the other hand, in applying a distributed learning framework for network slice allocation, particularly via federated learning framework, the researchers of [44] introduced a deep federated Q-learning (DFQL) technique to manage and allocate resources for differentiated QoS services in networks. Furthermore, in [44], a federated deep reinforcement learning (FDRL) framework was proposed to train bandwidth allocation models among Mobile Virtual Network Operators (MVNOs) in RANs. In another early work of applying the distributed learning approach to 5G network slicing, the research work conducted in [45] employed federated learning to predict the slices' service-oriented Key Performance Indicators (KPIs) by forecasting the response time of the Mobility Management Entity (MME) as one of the key service-oriented KPI of a VNF running inside a network slice. Next, researchers in [46] exploited a statistical federated learning (SFL) framework to SFL enforce a slice-based service level agreement that was demonstrated to result in further network overhead reduction in contrast with both the state-of-the-art FedAvg and centralized constrained deep learning approaches. To provide 5G slicing services in RAN, researchers in [47] envisioned a framework, referred to as O-RANFed, that was shown to improve the performance of federated learning by a joint mathematical optimization model of local learners selection and resource allocation. The O-RANFed system employed the Successive Convex Approximation (SCA) method to improve the solution to the original resource allocation problem.

4.3 Problem Description and Motivation

From the related research work in section 5.2, we understand that there is a research gap in terms of distributed learning paradigm for network slice allocation to UEs in a 5G network system. In particular, the existing research work did not consider UE privacy incorporation while designing classical optimization models. On the other hand, other researchers considered distributed models in a federated learning framework that may attain UE-privacy, however, did not investigate how to further improve the network overheads while training a large-scale, localized model for network slice allocation. These two problems consist of the formal problem description in our work. Based on this problem, we consider an asynchronous weight updating federated learning framework for network slicing that we adopt as a model aggregation model.

4.4 Methodology

In this section, we present our proposed methodology based on the system model assumption of network slices for contemporary mobile networks. For details of the considered communication system model, readers are referred to the work in [43]. In the remainder of this section, we elucidate the dataset preparation, the deep learning model leveraged for both local and global model training, and our proposed federated learning framework with asynchronous weight updating capability.

4.4.1 Dataset Preparation

Before delving into the details of the deep learning technique for network slice allocation for UEs, we first describe the steps pertaining to dataset preparation. We employ the DeepSlice dataset, utilized in [43], that consists of KPIs from both the network and UEs. The KPIs include the type of UEs that request the connection, QoS Class Identifier (QCI), packet delay budget, maximum packet loss, time instant, and the day of the week. The dataset was collected from a variety of devices requesting access to the network operator that include smartphones, general Internet of Things (IoT) devices, Augmented/Virtual Reality (A/VR) devices, Industry 4.0 traffic, e911 or public safety communication, healthcare, smart city or smart homes traffic, and so forth. In addition, unknown devices requesting access to one or multiple services were included in the dataset.

Table 4.1 lists the features of the dataset according to [43]. The average amount of time that each incoming request spends in the system is displayed in the second column. The network slice categories include enhanced Mobile Broad Band (eMBB), Ultra Reliable Low Latency, Communication (URLLC), massive Machine Type Communication (mMTC), and the Master slice. The latter refers to a slice comprising network functions belonging to each of the other slices [43], and serves as a default slice allocation for those UEs that are not rendered a dedicated slice type.

Table 4.1: Partial results summary.

Input Type	**Duration (sec)**	**Packet Loss Rate**	**Packet Delay Budget (ms)**	**Predicted Slice**
Smartphone	300	0.01/0.001 /0.000001	60/75/100/ 150/300	eMBB
IoT Device	60	0.01	50/300	mMTC
Smart Transportation	60	0.000001	10	URLLC
Industry 4.0	180	0.001/0.000001	10/50	mMTC/URLLC
AR/ VR/ Gaming	600	0.001	10/50	eMBB
Health care	180	0.000001	10	URLLC
Public Saftey / E911	300	0.000001	10	URLLC
Smart City / Home	120	0.01	50/300	mMTC
Unknown Device Type	60/120 /180/300	0.01/0.001 /0.000001	10/50/60/75/ 100/150/300	eMBB/mMTC/URLLC

4.4.2 Centralized Deep Learning Model for Network Slice Prediction Federated Learning Model

We use the same framework used in chapter 3

4.5 Performance Evaluation

To evaluate the performance of our proposed asynchronously weight updating federated learning framework, we first compare its performance with the synchronously weight updating (the classic federated learning) model performance with a varying number of client nodes (i.e., UEs). In Algorithm 11, at the clients' side procedure, we removed the condition of (t mod δ) to derive the synchronously weight updating condition. Second, we evaluated the deep learning model performance by monitoring the loss against a varying number of clients and over a different number of time-rounds. In addition, we compared the standalone, centralized model's performance with that of our proposal, i.e., the clients models aggregation. Third, we evaluated the performance of our asynchronously weight updating strategy by monitoring the deep learning model loss with different time rounds. Finally, we evaluated the overhead model reduction along with different time-round values and varying deep parameter rates.

First, in Fig. 4.1, the performance of our proposed asynchronously weight updating federated learning technique in terms of loss with an increasing number of clients (i.e.,

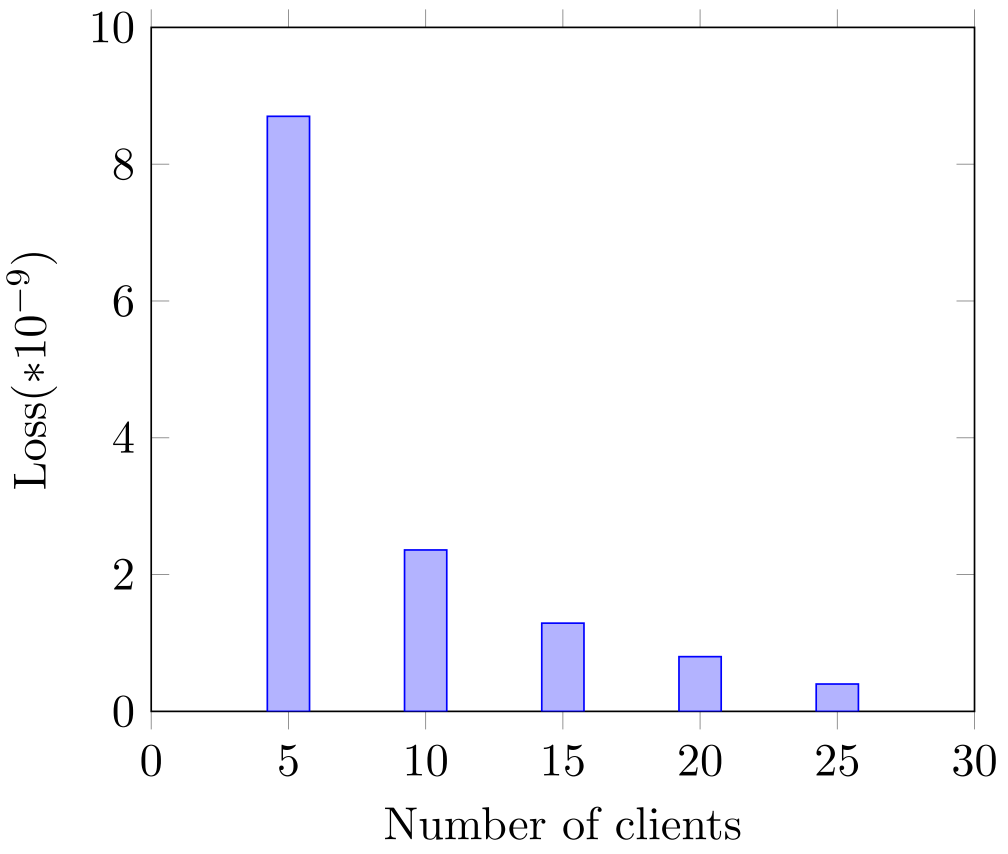

Figure 4.1: Loss performance for varying numbers of clients (UEs) in case of the proposed asynchronously weight updating method.

UEs). On the other hand, Fig. 4.2 shows the loss performance of the traditional method (synchronously weight updating mechanism) for various UEs. As evident from these results, our proposed asynchronously weight updating method approximates the performance of the traditional method with an increasing number of users. In summary, the performance of the proposal steadily improves with the number of clients.

Next, Fig. 4.3 demonstrates the performance of our proposal in terms of loss over varying time-rounds. The figure shows that the model performance increases with the number of time-rounds. This result shows that the model performs better with more shallow updates, focusing on the general features learned on the clients' side more than the model-specific features. Furthermore, experimental results demonstrate that the aggregated model training performance over the epochs is comparable with the centralized model, thereby indicating the reliability of our proposal in proactive slice-type allocation.

Finally, Fig. 4.4 demonstrates the overhead reduction (with the network bandwidth usage percentage as a performance indicator) for varying deep parameter update rates and different time-round values for each parameter. The result illustrates that the increase in the

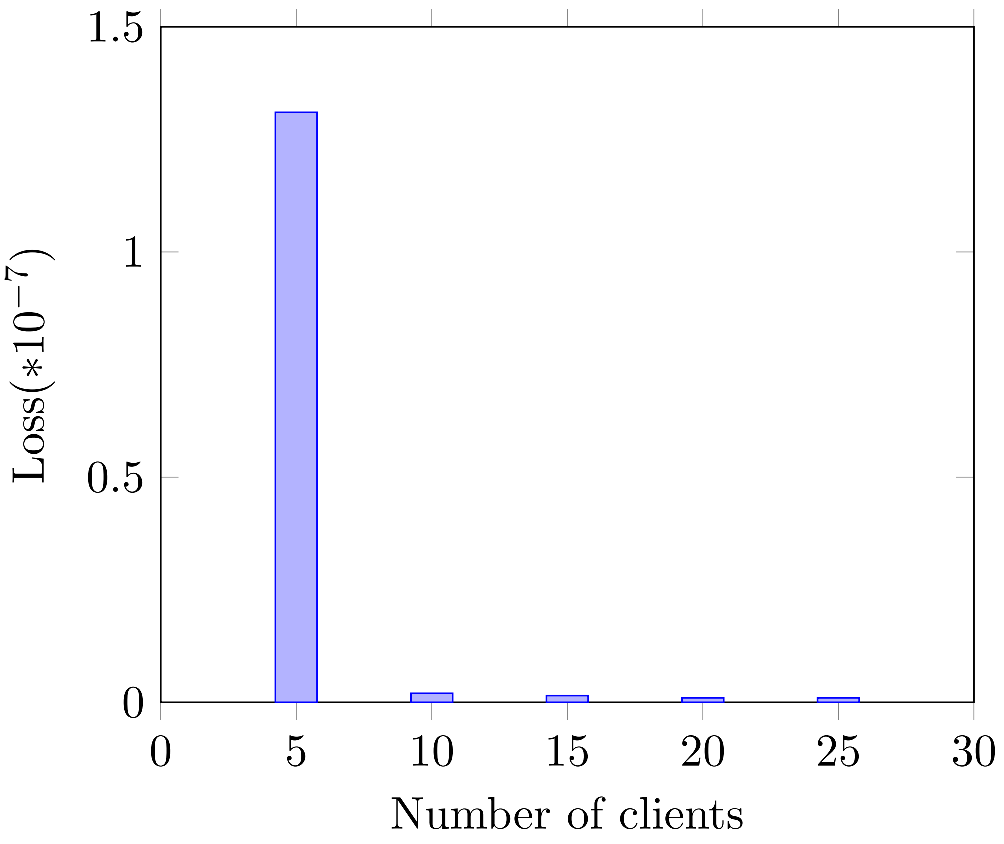

Figure 4.2: Loss performance for varying numbers of clients (UEs) in case of the traditional synchronously weight updating method.

deep learning parameters significantly improves the overhead reduction for the considered time-rounds of 5, 10, 15, and 20.

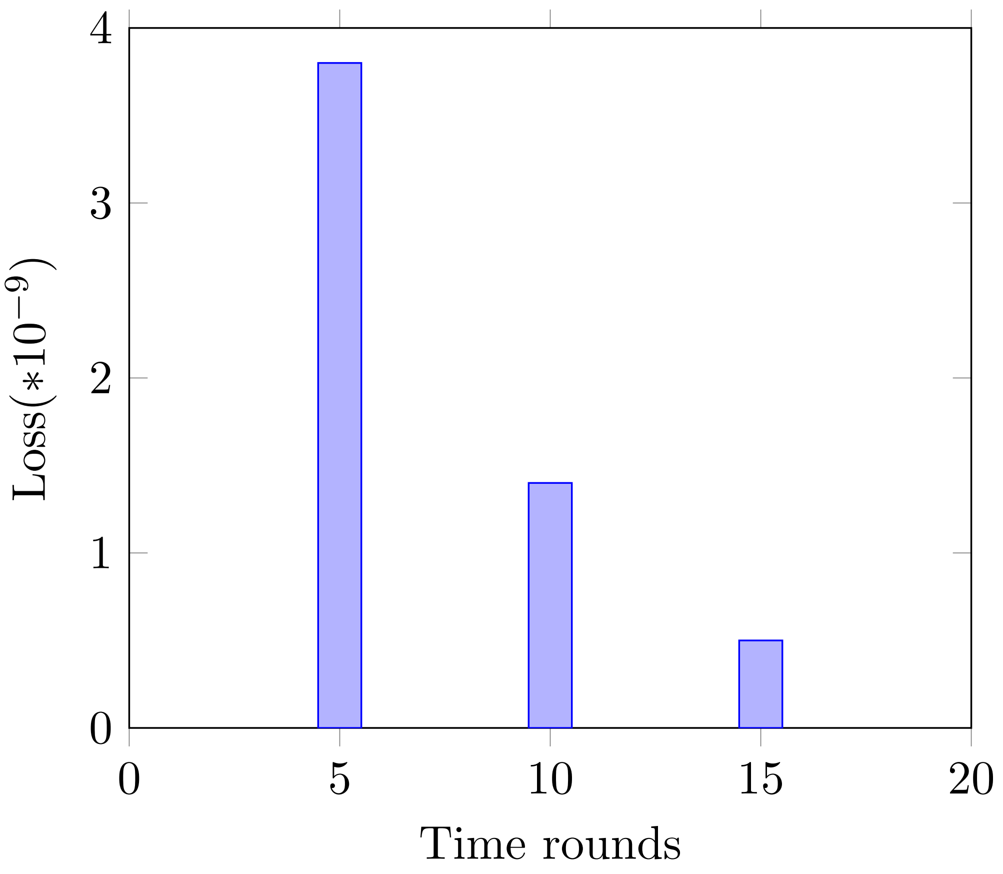

Figure 4.3: Loss vs number of time-rounds in the proposed asynchronously weight updating federated learning framework.

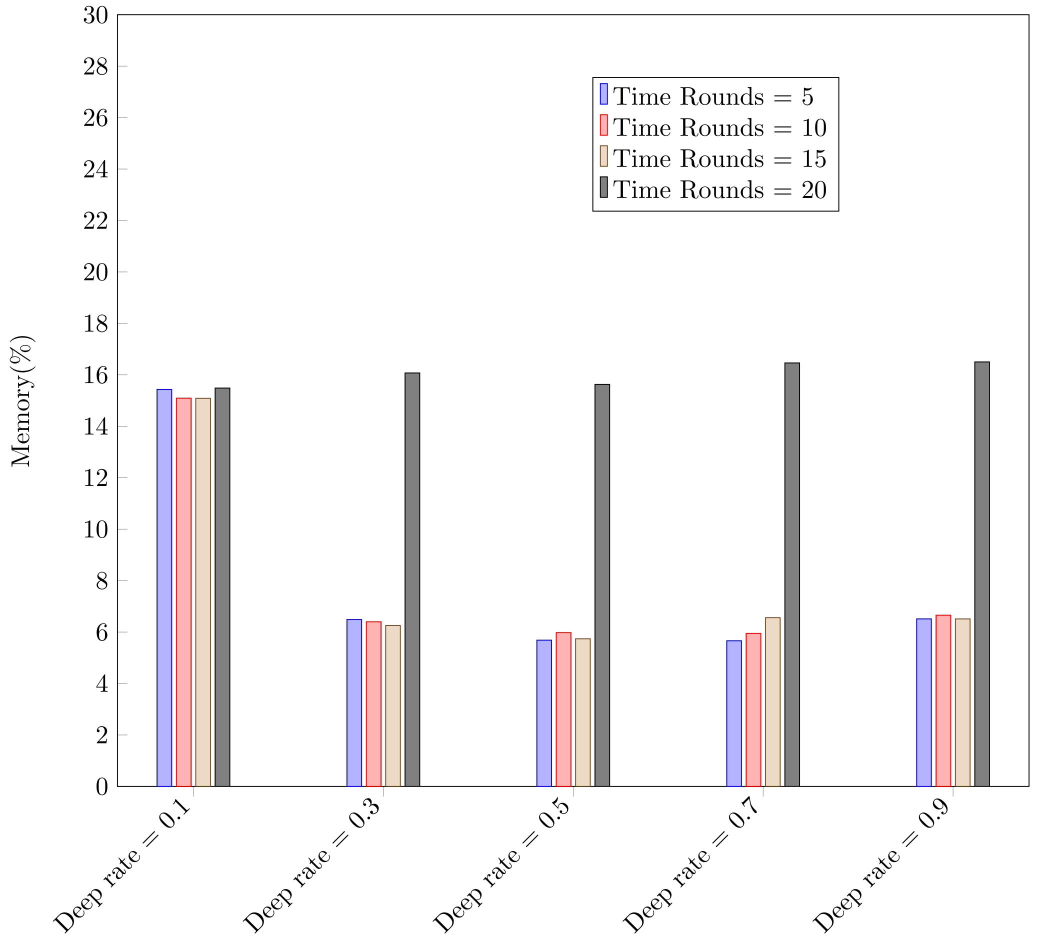

Figure 4.4: Overhead reduction ration on memory used for different deep parameters rates each along with different time rounds values.

Chapter 5

PC-SSL: Peer-Coordinated Sequential Split Learning for Traffic analysis of mmWave 5G Networks

Fifth Generation (5G) networks operating on mmWave frequency bands are anticipated to provide an ultra-high capacity with low latency to serve mobile users requiring high-end cellular services and emerging metaverse applications. Managing and coordinating the high data rate and throughput among the mmWave 5G Base Stations (BSs) is a challenging task, and it requires intelligent network traffic analysis. While BSs coordination has been traditionally treated as a centralized task, this involves higher latency that may adversely impact the user's Quality of Service (QoS). In this chapter, we address this issue by considering the need for distributed coordination among BSs to maximize spectral efficiency and improve the data rate provided to their users via embedded AI. We present Peer Coordinated Sequential split learning dubbed PC-SSL, which is a distributed learning approach whereby multiple 5G BSs collaborate to train and update deep learning models without disclosing their associated mobile users data, i.e., without privacy leakage. Our proposed PC-SSL minimizes the data transmitted between the client BSs and a server by processing data locally on the clients. This results in low latency and computation overhead in making handoff decisions and other network operations. Based on a real dataset, we evaluate the methodology in the mmWave 5G throughput prediction use-case. The results demonstrate that our methodology outperforms local models and has a comparable performance to vanilla centralized split learning.

5.1 Introduction

After we deployed the asynchronous federated learning framework in chapters 3 and 4, we find that there is still some room for performance improvement in terms of a distributed learning approach. Federated learning methodologies still have some limitations. First is the computational limitations which are limited by the client's capabilities, where the server and the client still share the same deep learning model, you cannot expand the model to the server's capabilities but to the clients'. Second, the entire model is shared and sent over the network to the clients, so every time we initialize or change the model it will put more load on the network. Since split learning doesn't have these limitations, we, in this chapter, introduce a new method of split learning and deploy it for mmWave 5G traffic analysis which is another important part of the 5G network where we predict the throughput to participate in improving the 5G networks spectral efficiency.
Recently, Fifth generation (5G) cellular networks emerged to support Ultra-Reliable Low-Latency Communication (URLLC), massive Machine Type Communication (mMTC), and enhanced Mobile Broadband (eMBB) services. The high bandwidth in 5G networks can be attributed to the new radio (NR) specifications, which encompass a wide range of frequencies, including low-band through mid-band to high-band, particularly mmWave frequency spectrum. These services open the way for a wide range of intriguing applications, including (Industrial) IoT, autonomous driving, Augmented/Virtual Reality (AR/VR), and ultra-high resolution and high-responsive metaverse applications. While the majority of commercial 5G services deployed globally in 2019 utilized mid-band and low-band frequencies for 5G systems, mmWave-enabled 5G BSs started to be commercially deployed [48].

5G mmWave technology offers higher data transfer rates and lower latency, opening up a plethora of new application fields and use cases. However, The signal intensity, signal-to-noise ratio (SNR), the distance between the device and the base station, and the number of

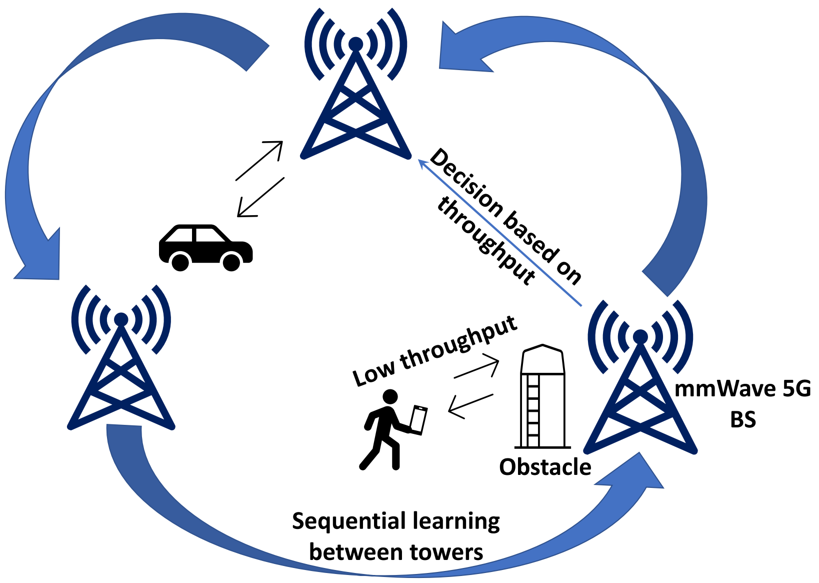
Decision based on throughput
Low throughput
mmWave 5G BS
Obstacle
Sequential learning between towers

Figure 5.1: A typical scenario of the netwrok traffic analysis problem based on user's mobility and location in mmwave 5G networks.

devices connected to the network are all essential variables that affect the throughput of 5G mmWave networks, making it challenging to estimate [49–51]. Conventional, centralized traffic analysis methods for forecasting 5G mmWave throughput include gathering and analyzing network data, which may be both time-consuming and privacy-violating.

The ultra-high bandwidth of mmWave 5G, which may theoretically reach up to 20 Gbps, opens up interesting new possibilities for supporting a wide range of current and future applications that require high bandwidth anticipated of the 5G eMBB service. However, many technical challenges face mmWave radios, including directionality, limited range, and high sensitivity to obstructions, making the design and management of 5G services based on mmWave radio difficult. Also, It might be challenging to establish and maintain a solid communication link with user equipment (UE), especially when the UE moves around as depicted in Fig. 5.1. Given these challenges, it is hard to carry out a real-time network traffic analysis and make proactive decisions regarding the bandwidth allocation, load balancing, user handoff, and so forth in a seamless manner. In other words, even though the development of mmWave 5G networks has transformed the telecommunications sector, adopting the mmWave frequencies has significant barriers, such as limited coverage area and increased path loss [51–53]. As a consequence, a key challenge in mmWave 5G network design is how to accurately predict the network throughput [54]. While conventional machine learning models have been employed to address this issue, they typically demand the exchange of vast amounts of data between devices and BSs, posing privacy and security concerns.

To address the 5G mmwave BSs distributed coordination, in this chapter, we present a split learning approach since this concept allows data to be incorporated into the deep learning models locally on BSs without violating privacy of the mobile users they serve. In particular, we explore the use of split learning for throughput prediction in mmWave 5G networks, and demonstrate its potential to improve network performance while maintaining user privacy. Split learning is a novel machine learning technique allowing multiple parties to train deep learning models to allow them to learn cooperatively without granting access to data [55]. By keeping the client data on the device and processing it locally, split learning can preserve the privacy of sensitive data while enabling efficient model training. This approach significantly reduces the amount of data that needs to be shared between the user device and the central server, making it ideal for applications in which data privacy is a concern. In the context of mmWave 5G networks, split learning can be used to predict network throughput while maintaining user privacy, as the data remains on the device and is not shared with the network operator or other users. By enabling network operators to predict throughput more accurately and allocate resources more efficiently, split learning can help optimize the performance of 5G mmWave networks and enable the deployment of new and innovative applications.

5.2 Related Work

Several works on throughput prediction in 5G are done in the literature using deep learning and machine learning methodologies [56]. Authors in [57] establish a technique for estimating the cellular link throughput for end-users and evaluating the efficacy of network slices. To achieve this, they conduct a measurement study to investigate real-world scenarios, including driving in urban, sub-urban, and rural areas, and experiments in crowded environments. Then, they construct machine learning models that utilize lower-level metrics that portray the radio environment to forecast the attainable throughput. In [58], authors propose a deep-learning-based TCP approach for a disaster 5G mmWave network. Their model learns about the node's mobility and signal strength and predicts the network is disconnected and reconnected, which helps adjust the TCP congestion window. Their work aims to provide network stability and higher network throughput. In [59], authors propose a deep learning-based framework to design and optimize a 5G air-to-ground network. They deploy a deep neural network to predict the user throughput and another DNN to optimize the throughput deployment parameters.

While conducting research on intelligent 5G throughput prediction using deep learning models, other scholars have also focused on the development of distributed 5G intelligent systems that prioritize privacy preservation. Several research was done using split and federated learning in 5G intelligent applications [60, 61]. In [62] authors propose a secure framework based on blockchain and federated learning (FL) that leverages smart contracts to prevent unreliable and malicious participants from participating in the FL process. The system automatically identifies such participants through the execution of smart contracts and thus mitigates the risk of poisoning attacks. Additionally, they incorporate local differential privacy techniques to safeguard against membership inference attacks. Authors of [63] introduce a novel hybrid threat detection approach using split-ML that leverages both machine learning and human intelligence to detect cyber threats. Their work focuses on analyzing Distributed Denial of Service (DDoS) attacks based on their temporal and threshold behavior across various network communication protocols.

5.3 Problem Description and System Model

In this section, we first present our considered system model, followed by a formal description of the research problem.

The motivation behind our work is to introduce a decentralized technique of split learning to make every client (i.e., 5G BS) capable of making accurate generalized decisions without the need for communication with a centralized server. Furthermore, we may conclude from the associated related research work in section 5.2 that there is a research gap in distributed learning systems in terms of throughput prediction in a 5G network system choosing a use-case based on traffic analysis of mmWave 5G networks. Moreover, previous studies did not take into account user-privacy while developing optimization models for traffic analysis in 5G networks. In Fig. 5.1, we depict a typical scenario of our use-case. The BSs perform the sequential split learning training process to be up-to-date in terms of deep learning model weights. The deep learning model at each BS analyzes every connected user's data based on the location and mobility and then predicts the throughput for that user. Based on the throughput, an intelligent decision could be made. For example, in the figure, the pedestrian is shown to suffer from low throughput due to obstacles, a scenario in which the user-device could change the BS it is connected to based on its location and mobility.

Based on the described scenario and problem setting, Fig. 5.2 illustrates the architecture of our peer-coordinated split learning system model. The training flows sequentially till the last user, and then the last user closes the circle by making the first user the next user. Every process between the user's client and its server is indicating a conventional local training process. And every process between the user's server and the next user's client represents a vanilla peer-to-peer split learning process. This process is described mathematically as follows. Assume that we have the neural network model denoted by $M(x : \theta)$, where x refers to the input and θ refers to the model parameters. In a peer-to-peer paradigm, the L layers of the model are split into L_s and L_d layers, denoting shallow and deep layers, respectively.

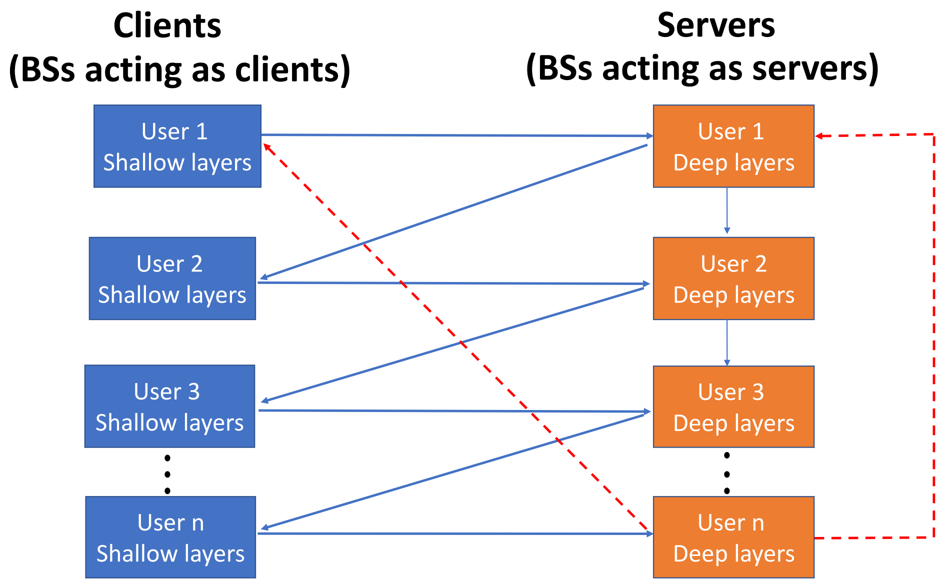

Figure 5.2: Peer-coordinted Split learning architecture. Note that during a time-round, a BS takes up the role of a server while other BSs serve as a client, and this is repeated for all the BSs in the considered 5G network. Also, note that the users refer to the participating BSs which aim to perform decentralized coordination for analyzing the network traffic with privacy-preservation. This simple yet powerful concept lays the foundation for a peer BS-coordinated split learning mechanism.

The client-end has the sub-model of L_s, and the server-side comprises the submodel $L - L_d$. Then, a process called data smashing takes place at the client-end, which transfers the input x into the feature representation $H = h(x)$, where H denotes the features vector represented by $A(F(x : \theta_s))$. Here, A refers to the activation function of the last layer in the L_s layers, and F represents the submodel of M. Then, the feature vector is sent to the server, where the output of the server is represented by $Y = A(F(H : \theta_d))$ such that Y in our aforementioned problem setting denotes the throughput of the considered 5G network system.

5.4 Methodology

In this section, we introduce our suggested decentralized technique, which is based on the network system model spatial parameters, along with user traffic and mobility, for commercial 5G mmWave networks. For further information on the communication setup and system under consideration, readers should refer to the work in [54]. We explain the dataset processing, the deep learning model used for training, and our proposed peer-coordinated sequential split learning architecture, referred to as PC-SSL, in the remainder of this section.

Table 5.1: Information about Areas.

Area	**Intersection**	**Airport**	**Loop**
Description	Outdoor 4-way traffic intersection	Indoor mall-area with shopping booths	Loop with railroad crossings, traffic signals, and open park restaurants
Trajectories	12	2	2
Trajectory Length	232 to 274 m	324 to 369 m	1300 m

5.4.1 Dataset Preparation

Before delving into the technicalities of the deep learning approach for networks, we introduce the data set adopted for our considered use-case. We considered the Lumos5G dataset, which is a collection of network performance data for over 100 commercial mmWave 5G BSs in the United States [54]. The dataset includes a range of features related to network performance, including signal strength, signal-to-noise ratio, channel quality, and network load. The data were collected using a custom data collection framework developed by Lumos Networks that includes a set of mobile measurement units to verify the network performance at various locations around each BS. Table 5.1 describes the locations from which data are collected. The dataset covers a range of different scenarios and use-cases, including indoor and outdoor environments, static and mobile devices, and different levels of network congestion. Overall, the Lumos5G dataset represents a valuable resource for researchers and practitioners interested in studying mmWave 5G network performance and developing new approaches for network performance prediction. Table 5.2 summarizes the data statistics while Table 5.3 presents the data attributes. In the data processing phase, we cleaned the data and applied preliminary processing (normalization and discretization) alongside feature selection. For our classification task, we selected the thresholds for the achievable throughput as follows; 0-150 Mbps as low throughput, 150-700 Mbps as medium throughput, and above 700 Mbps as high throughput.

5.4.2 Selection of Centralized Deep Learning Model for Throughput prediction

Here, we describe our selection of the centralized deep learning model for network for Throughput prediction given the considered system model. Among a number of candidate

Table 5.2: Summary of the data statistics used in the study.

Data Points	563,840 (per-sec. throughput w/ feature) samples
Mobility Modes	Walking (331 km), Driving (132 km), Stationary
Data	$38,632$ GBs of data downloaded over 5 G
Duration	6 months

Table 5.3: Data Fields and Descriptions.

Field	**Description**
Raw values	
Latitude & longitude	UE's spatial coordinates
User's mobility/activity	Indicates whether the user is walking, standing, or driving.
Moving speed	UE's moving speed reported by Android API
Compass direction	The horizontal direction of travel of the UE with respect to the North Pole (also referred to as azimuth bearing) and its accuracy
Post-processed values	
Throughput	Downlink throughput reported by iPerf 3.7
Radio type	Indicates whether the UE is connected to 5G or 4G
Cell ID	Identifies the tower the user is connected to
Signal strength	Signal strength of LTE (rsrp, rsrq, rssi) and 5G (ssrsrp, ssrsrq, ssrssi)
Horizontal handoff	UE switches from one 5G panel (cell ID) to another
Vertical handoff	UE switches between radio types (e.g., 4G to 5G)
UE-Panel distance	Distance between the UE and the panel it is connected to
Positional angle (θ_p)	Angle between UE's position relative to the line normal to the front-face of the 5G panel
Mobility angle (θ_m)	Angle between the line normal to the front-face of the 5G panel and UE's trajectory

machine and deep learning models, we select the DNN (Deep Neural Network) for its recent success in communication networks and being a lightweight DL model. A DNN is a feed-forward neural network architecture with Dense layers fully-connected and usually high in Depth. Due to the format of the network slice data prepared (as mentioned in section 5.4.1), we use a DNN.

5.4.3 Proposed Sequential Split Learning Model

Split learning was first introduced in [55]

Algorithm 11 demonstrates the steps of the update and training in our proposed framework. The algorithm works by splitting the model between the server and user devices, where the server holds the global model parameters and user devices hold their own local model parameters. The algorithm iterates over a fixed number of iterations T, and for each iteration, it randomly selects a subset of user devices to update their local model parameters. The split ratio r determines the proportion of user devices that will update their local model parameters during each iteration. If a user device is selected to update its local model parameters, it computes the local gradient of the loss function with respect to the model parameters based on its own data and sends the gradient to the server. The server updates the global model parameters using the received gradients and sends the updated parameters back to the user's device. If a user device is not selected to update its local model parameters, it sends a random batch of its data to the server to be used for updating the global model parameters. The algorithm combines the updated model parameters from all user devices using the average function to obtain the final global model parameters.

In algorithm 14, we introduce our algorithm and illustrate the main differences between it and the vanilla algorithm. Looking into 14 in every iteration, we loop on the number of towers, and every tower is considered a server and client. During the learning process, every tower updates its server from its client and then starts a peer-to-peer split learning process with the next tower client in the stack. After that, the server updates its weights and its client's weights then the next tower's server is updated and repeats the process. The learning process is done sequentially in a closed loop where every tower gets updates from the previous one and starts the peer-to-peer mechanism with the next one.

5.5 Performance Evaluation

To evaluate the performance of our proposed framework, we present two comparisons between every tower's model in the two approaches of standalone and sequential learning. First, we compare the accuracy progress of every tower in the two approaches as illustrated in figs., 5.3,5.4,5.5.

Algorithm 4: SplitNN Algorithm

```
Input  : $(x, y) \in D$, batch size $B$, number of epochs $E$, learning rates $\alpha_c$, $\alpha_s$
Output: Trained client and server models
1  Initialize client and server models $f_c$ and $f_s$ randomly;
2  for each epoch $e \in [1, E]$ do
3      Divide the data into batches $D_1, D_2, ..., D_{|D|/B}$;
4      for each batch $D_i \in D$ do
5          Client computes the gradients $\nabla_c L(f_c(x_i; \theta_c), y_i)$ and sends them to the
             server;
6          Server aggregates the gradients from all clients, updates the server model
             parameters, and sends the updated model parameters to the clients;
7          Client updates its own model parameters using the server's updated model
             parameters;
8      end
9      Server updates its own model parameters using the updated model parameters
         from the clients;
10 end
11 return Trained client and server models $f_c$ and $f_s$;
```

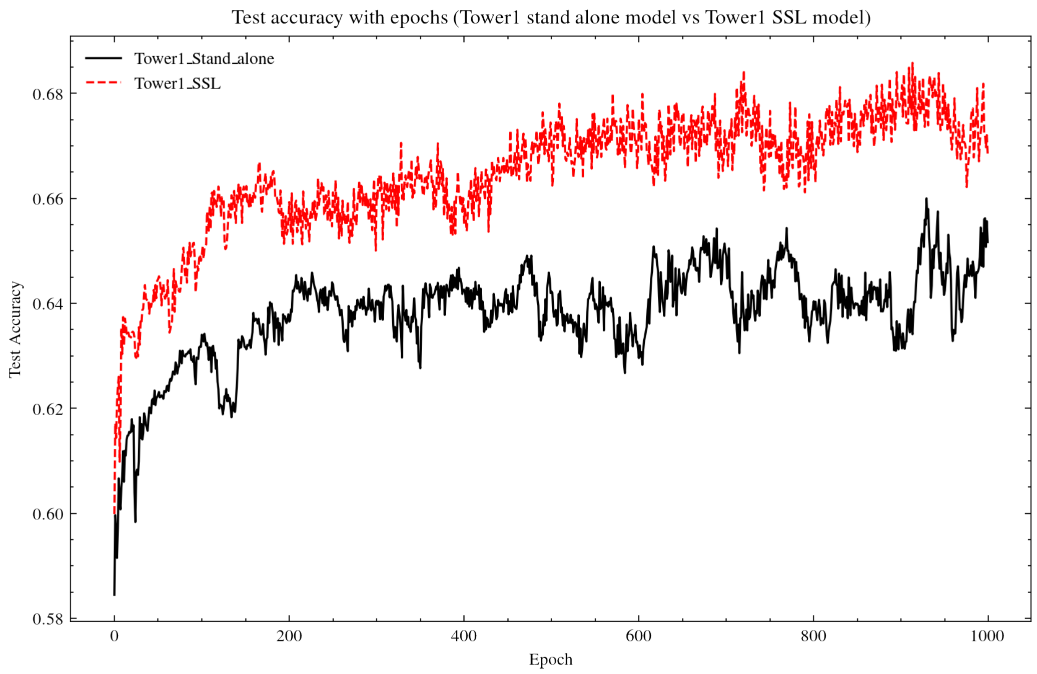

Figure 5.3: Comparison of first tower model in Standalone approach and Sequential learning approach

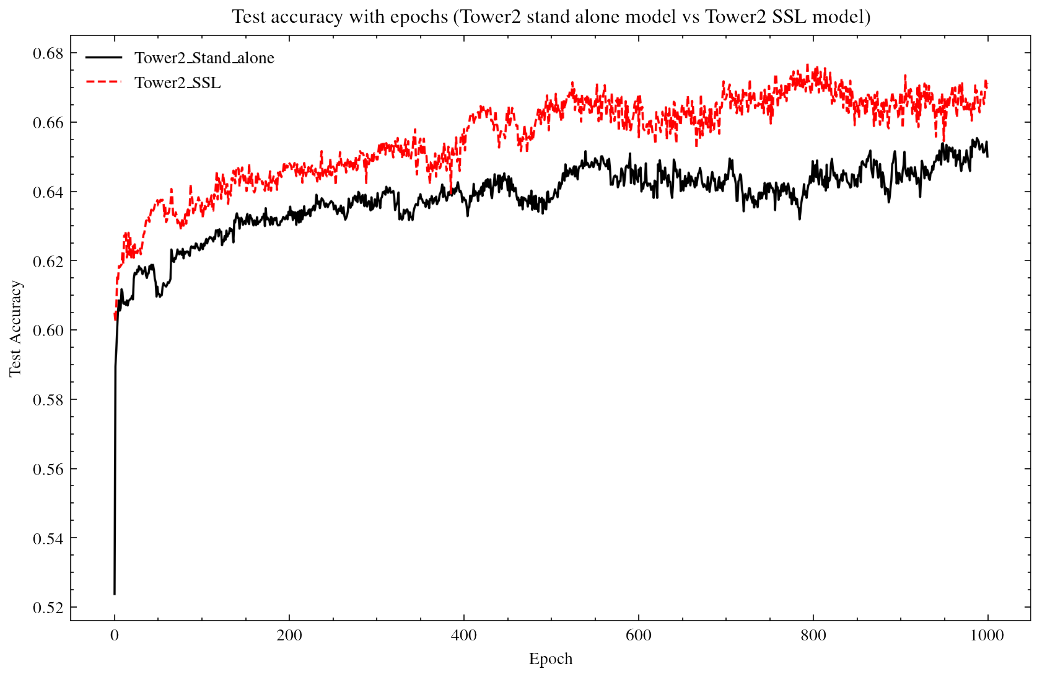

Figure 5.4: Comparison of Second tower model in Standalone approach and Sequential learning approach

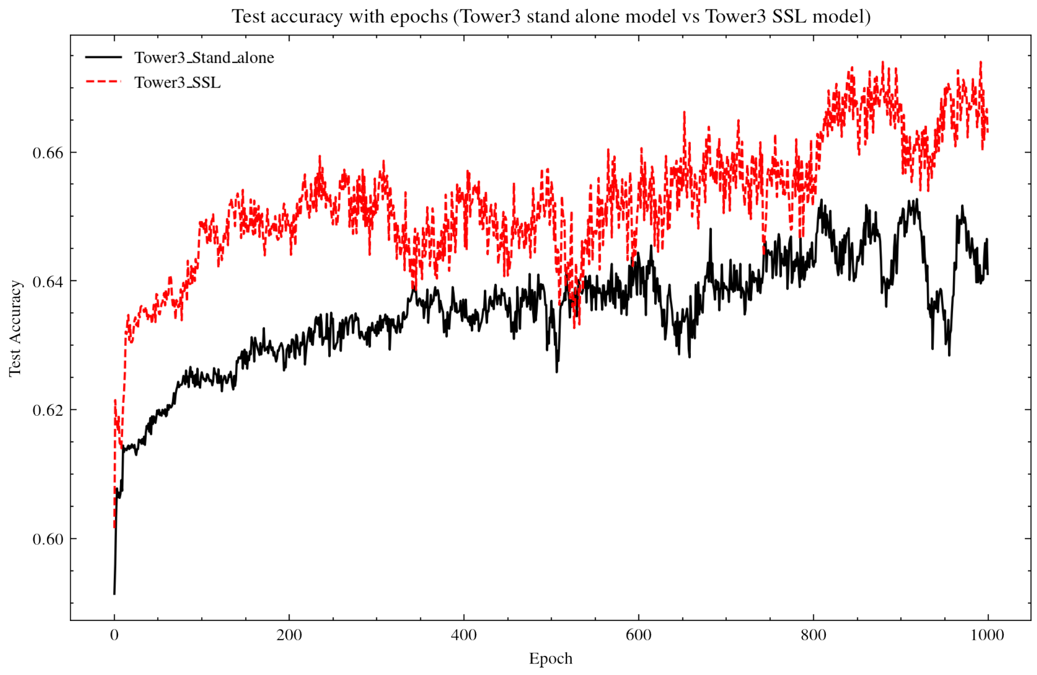

Figure 5.5: Comparison of Third tower model in Standalone approach and Sequential learning approach

```
Algorithm 5: PC-Sequential SplitNN Algorithm
Input  : $(x, y) \in D$, batch size $B$, number of epochs $E$, learning rates $\alpha_c$, $\alpha_s$ ,
         Stack of Towers T[Servers, Clients], number of Towers n
Output: Trained client and server models
1  Initialize client and server models $f_c$ and $f_s$ randomly;
2  for each epoch $e \in [1, E]$ do
3      Divide the data into batches $D_1, D_2, ..., D_{|D|/B}$;
4      for each batch $D_i \in D$ do
5          for each tower t in T[Servers, Clients] do
6              $Client_i$ computes the gradients $\nabla_c L(f_c(x_i; \theta_c), y_i)$ and sends them to the
                server;
7              $Server_i$ completes the forward pass and compute the gradients;
8              $Client_i$ updates its model parameters.;
9              $Server_{i+1}$ updates its model parameters.;
10             $Client_{i+1}$ updates its model parameters.;
11         end
12     end
13 end
14 return Trained stack of clients and servers models $f_c$ and $f_s$;
```

Then illustrate the accuracy performance progress of every tower in the sequential model and the server performance of the vanilla split learning model as illustrated in fig. 5.6.

We then evaluate each model of the towers in both the two scenarios of standalone and SSL alongside the central model with vanilla splitNN. We use accuracy, precision, and F1-score in evaluation. Table 5.4 shows the results of these metrics in a micro-analysis methodology for all three classes.

Table 5.4: Metrics results summary of all models.

Models		Metrics								
		Accuracy			Precision			F1-score		
SSL models	Tower1	**0.825**	0.749	0.756	**0.802**	0.5	0.624	**0.778**	0.462	0.674
	Tower2	**0.822**	0.778	0.753	**.795**	0.581	0.612	**0.776**	0.48	0.682
	Tower3	**0.823**	0.759	0.757	**0.787**	0.526	0.622	**0.779**	0.454	0.679
Standalone models	Tower1	**0.820**	0.733	0.747	**0.825**	0.467	0.614	**0.761**	0.468	0.66
	Tower2	**0.809**	0.747	0.743	**0.777**	0.493	0.60	**0.759**	0.415	0.668
	Tower3	**0.8**	0.742	0.738	**0.77**	0.483	0.60	**0.747**	0.448	0.649
	Central	**0.823**	0.77	0.78	**0.78**	0.549	0.666	**0.783**	0.506	0.699

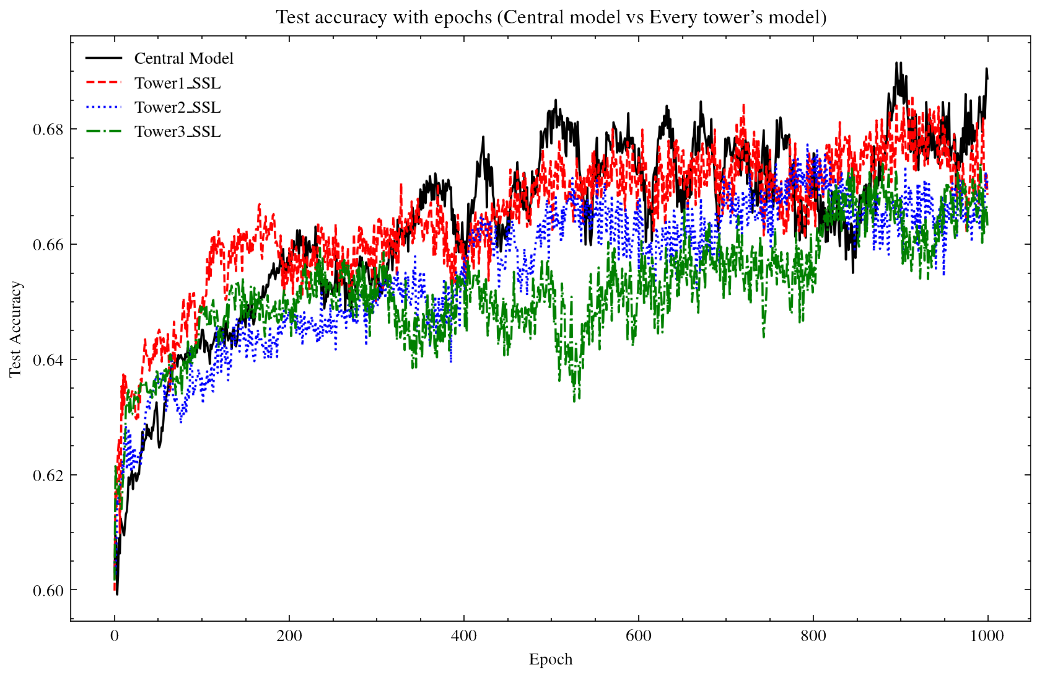

Figure 5.6: Comparison of Central model in Standalone approach and all towers in Sequential learning approach

Chapter 6

Distributed learning Paradigms for channel quality estimation in LTE Networks

6.1 Introduction

This chapter investigates channel quality indicator prediction in terms of SNR. Fifth Generation (5G) technology is intended to enable large data speeds on the downlink and uplink through more reliable and low latency systems. New Radio (NR) represents an advancement to 4G/LTE wireless systems in terms of Radio Access Networks (RANs). In 5G-based network systems, Device-to-Device (D2D) wireless communications are emerging rapidly in the 5G-enabled Internet of Things (IoT) and massive MIMO fields [64–67].

Estimation techniques for Signal-to-Noise Ratio (SNR) have been extensively studied and have contributed to network optimization in both past and current wireless communication technologies. Although SNR estimation techniques have been extensively researched and enabled network optimization, there are still opportunities to improve their accuracy and readiness. For example, better accuracy in SNR prediction can lead to improved transmission, such as modulation, power, and capacity. Similarly, shorter prediction turnaround time or latency can result in more efficient transmission, with less overhead and more data payload. Deep Learning (DL) is a new paradigm that has been successfully applied in various domains. DL has also been utilized in wireless communication.

Despite the expanding work in channel quality indicator prediction, particularly in terms of SNR, the deployment of a distributed learning paradigm is not popular yet, especially in D2D communication systems. In this chapter, we deploy our two distributed learning paradigms namely the asynchronous federated learning framework and the peer-coordinated sequential split learning.

6.2 Problem Description

Our system model's network topology can be considered as a D2D system composed of a device set of N transmitter-receiver $(\mathrm{Tx} - \mathrm{Rx})$ pairs. The configuration of $\mathrm{Tx} - \mathrm{Rx}$ connection can be represented as follows: 1) A source node S, acting as an access point or a service-providing end that communicates with a mobile user equipment (UE). 2) A D2D relay node R that sends data to the following node. 3) A D2D Rx node, also called destination node D (such as an access point or base station), that receives data from a D2D relay node. In order to improve the spectral efficiency through the channel quality q_C parameter, we need to design a model for SNR prediction as a channel quality indicator CQI. This channel quality indicator, i.e., SNR, can be estimated using the following equation:

$$\mathrm{SINR} = 10 \log_{10} \frac{\sigma_S}{(\sigma_I + \sigma_N)}$$

where σ_S, σ_I, and σ_N denote the desired signal power, interference signal power, and noise power, respectively.

6.3 Methodology

In this section, we present our proposed methodology based on the system model assumption of channel quality estimation based on SNR. In the remainder of this section, we elucidate the dataset, the deep learning model leveraged for both local and global model training, and our proposed distributed learning frameworks.

6.3.1 Data

We use the dataset in [68]. This data set includes measurements of the signal strength of a WSN link in an indoor scenario. It consists of more than 8000 data records between a Tx–Rx pair placed over a distance of 10 to 35 m, and it has different measurement metrics such as energy, throughput, delay, and packet loss. In our evaluation, we use two measurements of distance (the 10m distance as the closest and the 35m point as the furthest).

6.3.2 Selection of Centralized Deep Learning Model

We are using the same deep learning model mentioned in chapter3 (1D CNN), with more lightweight layers.

6.3.3 Proposed Distributed Learning Models.

In this chapter, we use both the asynchronous federated learning framework mentioned in chapter3 and the sequential split learning framework introduced in chapter 5.

6.4 Performance Evaluation

To evaluate the performance of our proposed distributed learning framework, we compare the two frameworks in terms of convergence, overhead reduction, and RMSE performance on test data with iterations. We first compare model convergence performance with varying numbers of clients for both sub-datasets as shown in figs 6.1 and 6.2. Then we compare the performance of the two methodologies in terms of overhead reduction in a communication network (Memory usage) with varying the number of clients as shown in figs 6.3 and 6.4.

Finally, we compare the convergence of RMSE during the training of both of the models with varying iterations as shown in figs 6.5 and 6.6.

We conclude from these results that the peer-coordinated split-learning approach outperforms the asynchronous federated learning in terms of convergence and overhead reductions.

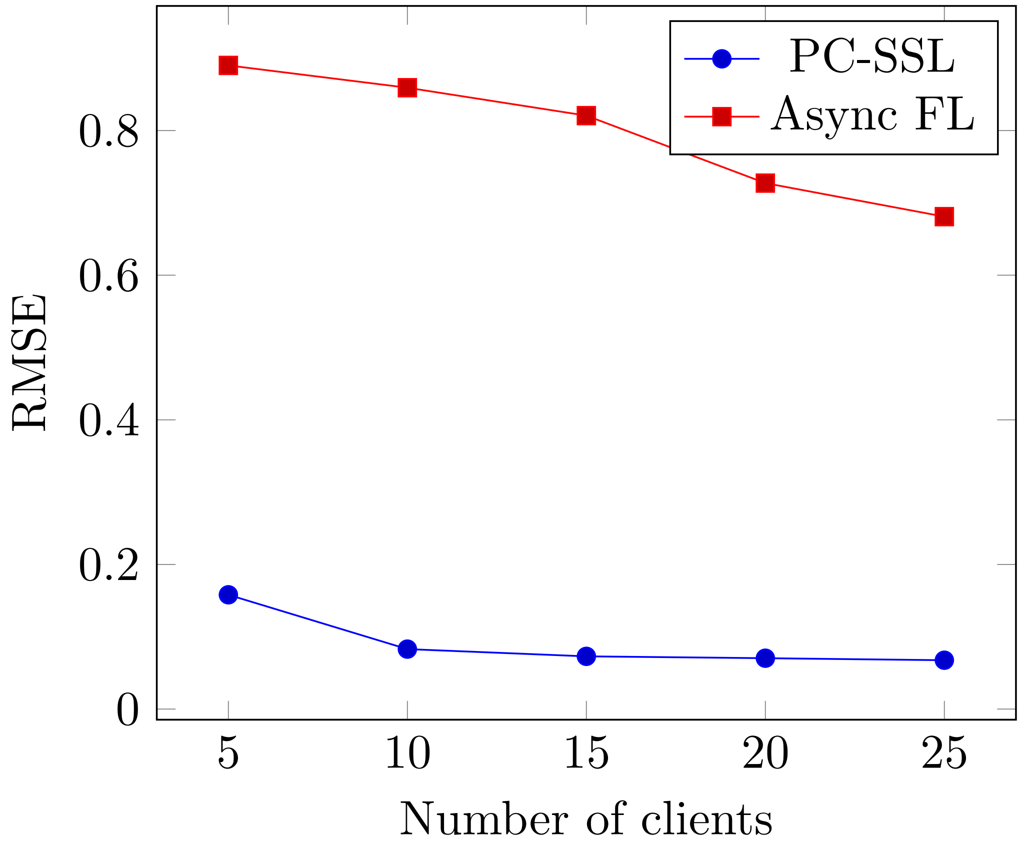

Figure 6.1: Convergence performance for varying numbers of clients (UEs) comparing Peer coordinated sequential split learning with Asynchronous federated learning.

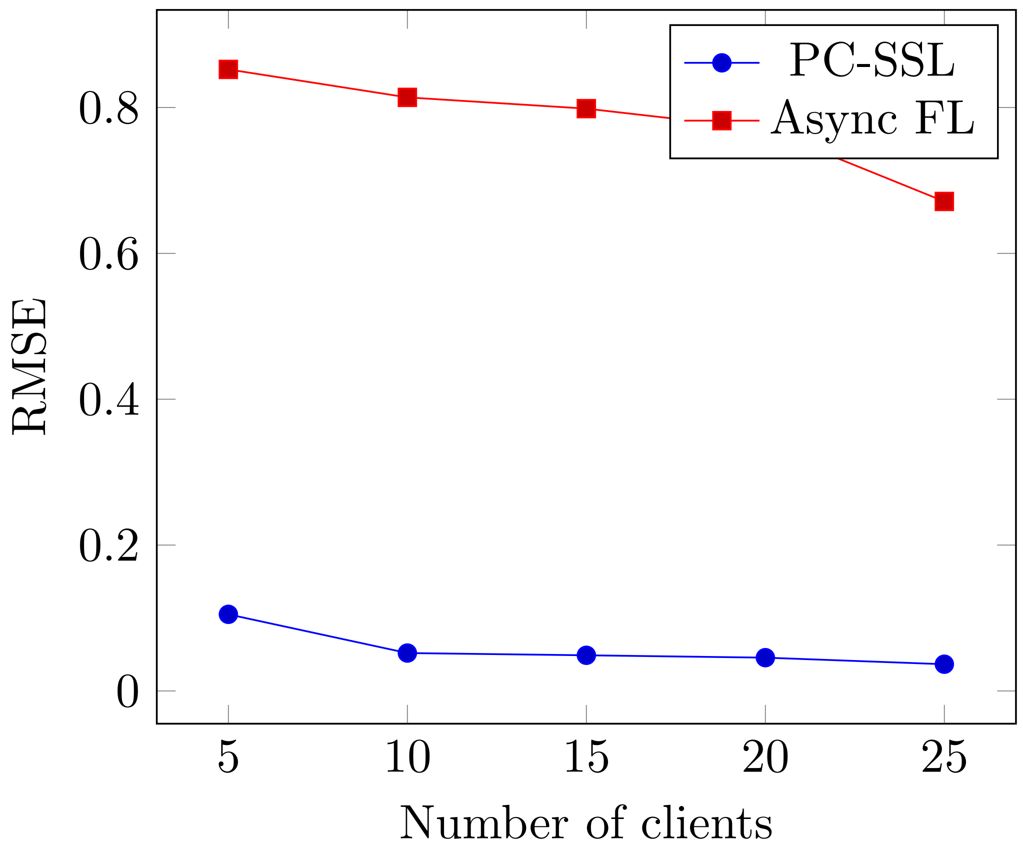

Figure 6.2: Convergence performance for varying numbers of clients (UEs) comparing Peer coordinated sequential split learning with Asynchronous federated learning.

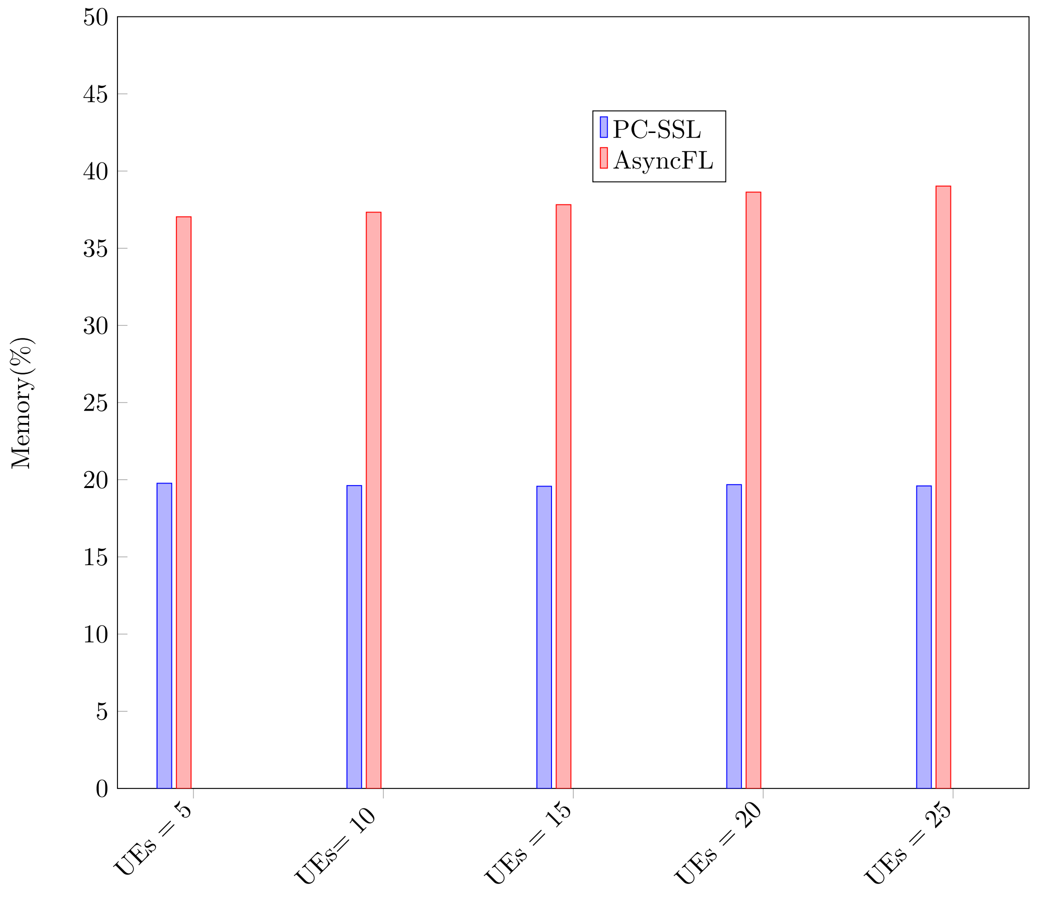

Figure 6.3: Overhead reduction ration on memory used for varying number of clients (UEs) deep comparing Peer coordinated sequential split learning with Asynchronous federated learning.

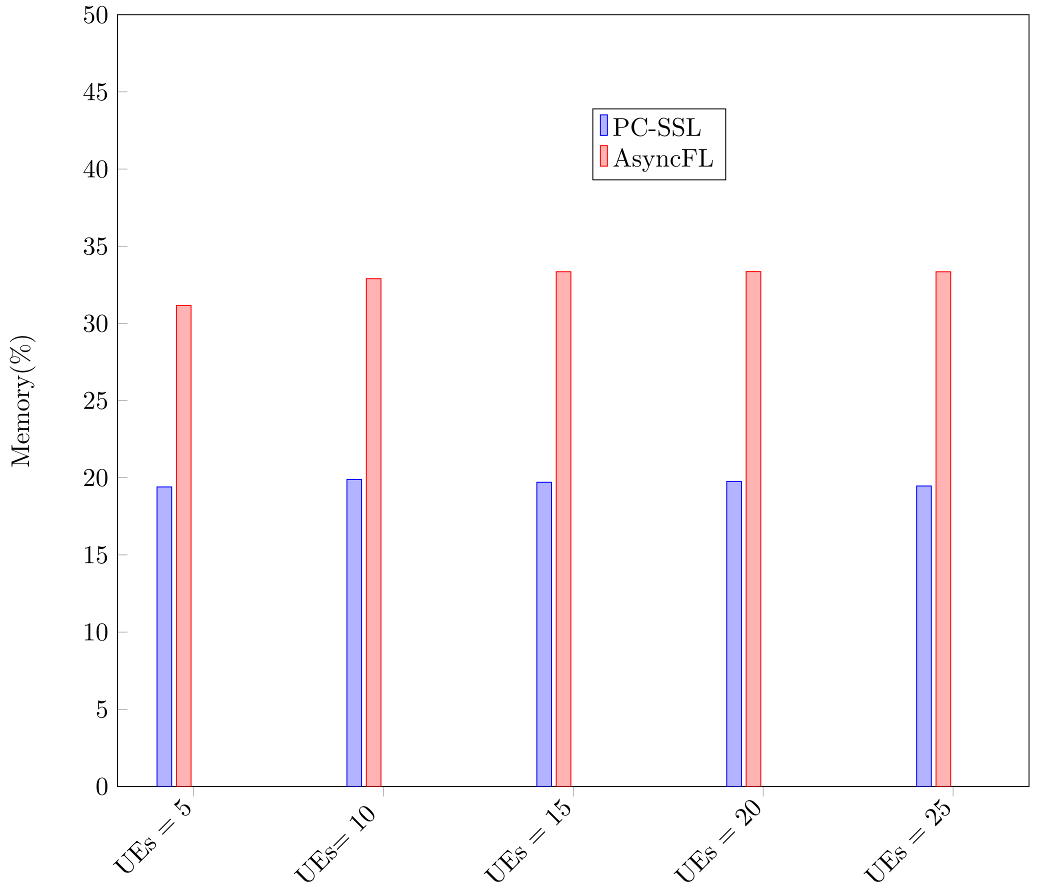

Figure 6.4: Overhead reduction ration on memory used for varying number of clients (UEs) deep comparing Peer coordinated sequential split learning with Asynchronous federated learning.

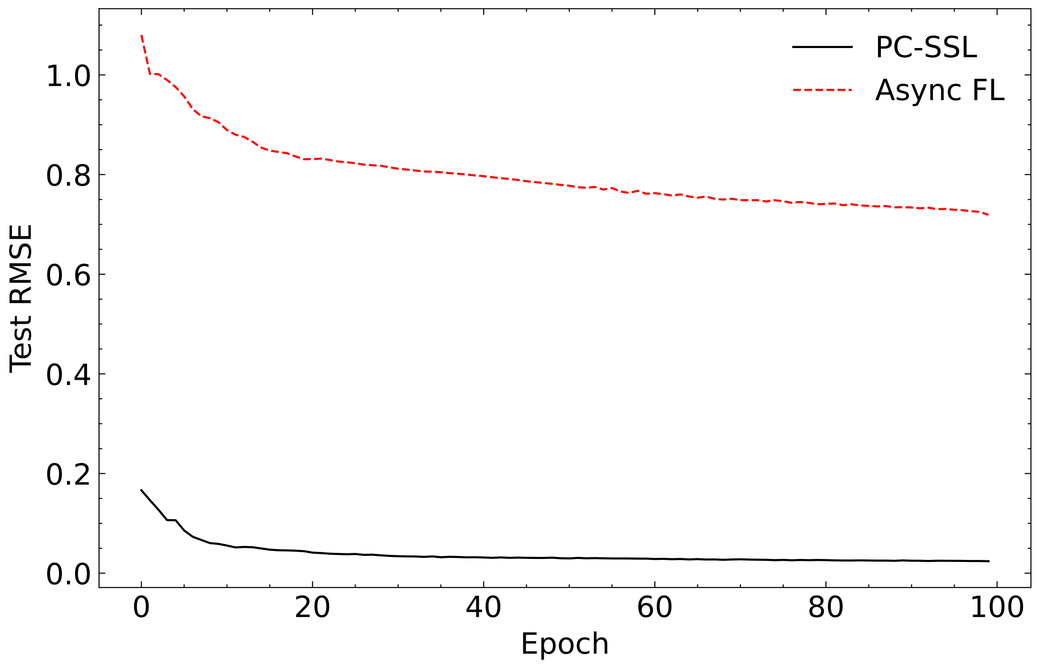

Figure 6.5: RMSE vs number of Epochs, A comparison between Peer coordinated sequential split learning and Asynchronous federated learning approach. Distance = 10m

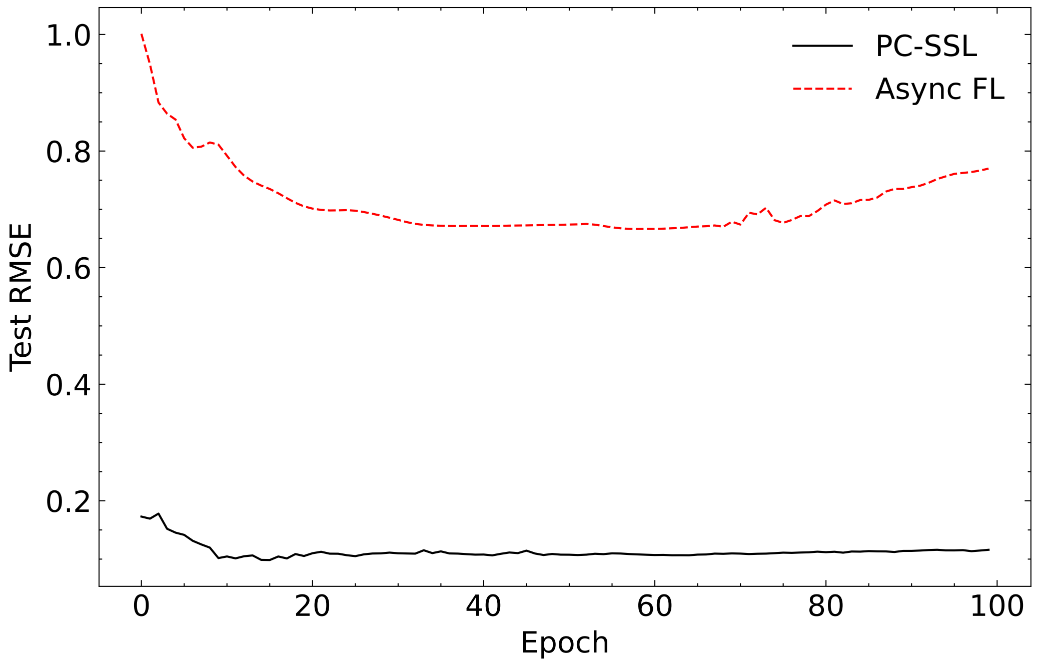

Figure 6.6: RMSE vs number of Epochs, A comparison between Peer coordinated sequential split learning and Asynchronous federated learning approach. Distance = 35m

Chapter 7

Conclusions and Future Works

This chapter summarizes the contributions of the dissertation work and manifests potential future research directions.

7.1 Contributions

This thesis presents two paradigms for privacy-preserving, distributed machine learning in 5G networks. The first approach is an asynchronous weight-updating federated learning framework for proactive network slicing. It allows for collecting more data for more accurate decisions without violating user privacy. This approach provides a reliable, low-latency method with reduced overhead, resulting in higher throughput. The proposed framework is capable of data-driven models for multi-tenant systems without the need for increased computational resources at a centralized node level.

The second approach is a decentralized split learning methodology for mmWave throughput prediction. The proposed sequential split learning framework enables each 5G base station to learn from a larger pool of user data (e.g., user location, mobility, traffic patterns, application types, and so forth), leading to more accurate and generalized decisions based on data-driven techniques with no need to establish communication with a central base station. This approach also ensures that user privacy is not leaked. This approach also ensures user privacy is not violated. The main objective of this paradigm is the development of a self-contained decision-making system that does not rely on a centralized server, allowing for intelligent decisions based on user mobility and traffic analysis. The results demonstrate the effectiveness of data-driven models in a spatial, wide-scale, multi-tenant system without requiring additional computational resources at a centralized level.

The proposed two paradigms' performance proves the feasibility and reliability of distributed AI frameworks in 5G-based network systems and their core parts and legacy parts, particularly regarding traffic analysis of communication.

7.2 Future Directions

Several future directions are possible.

- Using PC-SSL Peer-coordinated split learning alongside with expanding the dataset to include more device and signal types for broader IoT device applications and extending the work beyond traffic analysis to temporal analysis based on the user's history.
- Expand the network slicing dataset (.e.g., more device types, signal types) to serve a broader range of IoT devices. Further research work is needed also for generating labels for local users (e.g., using a semi-supervised approach) and for applying other model aggregation methods to serve a broader spectrum of service types.
- Expand the centralized models to different machine learning algorithms with distributed learning frameworks. Classical machine learning models could be considered, such as decision trees, random forests, and support vector machines, introduce a new collaborative learning methodology with classical models.
- Evaluate the impact of network topology on the performance of the distributed learning frameworks. We may study how different network topologies affect distributed learning algorithms' accuracy and convergence speed.
- Explore the use of transfer learning techniques to improve the performance of the proposed frameworks by investigating how pre-training models on related tasks or domains can help improve accuracy and reduce the data needed for training or affect heterogeneity.
- Find whether and how the proposed distributed learning models could be vulnerable to attacks.
- Usage of meta-learning techniques to improve the performance of asynchronous federated learning and PC-SSL. Meta-learning algorithms can be used to select the best machine-learning models and hyperparameters adaptively.

Bibliography

[1] V. V. Veeravalli and P. K. Varshney, "Distributed inference in wireless sensor networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1958, pp. 100–117, 2012.

[2] R. Dangi, P. Lalwani, G. Choudhary, I. You, and G. Pau, "Study and investigation on 5g technology: A systematic review," *Sensors*, vol. 22, no. 1, 2022.

[3] O. Nassef, W. Sun, H. Purmehdi, M. Tatipamula, and T. Mahmoodi, "A survey: Distributed machine learning for 5g and beyond," *Comput. Netw.*, vol. 207, no. C, apr 2022. [Online]. Available: https://doi.org/10.1016/j.comnet.2022.108820

[4] Y. Zhang, A. Wu, Z. Chen, D. Zheng, J. Cao, and X. Jiang, "Flexible and anonymous network slicing selection for c-ran enabled 5g service authentication," *Comput. Commun.*, vol. 166, pp. 165–173, 2020.

[5] E. S. Xavier, N. Agoulmine, and J. S. B. Martins, "On modeling network slicing communication resources with sarsa optimization," 2023. [Online]. Available: https://zenodo.org/record/7513695

[6] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[7] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[8] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[9] I. Sarrigiannis, E. Kartsakli, K. Ramantas, A. Antonopoulos, and C. Verikoukis, "Application and network vnf migration in a mec-enabled 5g architecture," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD).* IEEE, 2018, pp. 1–6.

[10] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *Ieee Access*, vol. 5, pp. 6757–6779, 2017.

[11] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The next generation wireless access technology.* Academic Press, 2020.

[12] Y. He, J. Ren, G. Yu, and Y. Cai, "D2d communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, 2019.

[13] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 38–67, 2019.

[14] R. van der Meulen *et al.*, "What edge computing means for infrastructure and operations leaders," *Smarter with Gartner*, 2018.

[15] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, vol. 7, pp. 127 276–127 289, 2019.

[16] J. Mao, M. A. Abdullahi, P. Xiao, and A. Cao, "A low complexity 256qam soft demapper for 5g mobile system," in *2016 European Conference on Networks and Communications (EuCNC)*, 2016, pp. 16–21.

[17] R. Chataut and R. Akl, "Massive mimo systems for 5g and beyond networks—overview, recent trends, challenges, and future research direction," *Sensors*, vol. 20, no. 10, p. 2753, 2020.

[18] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in nfv networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, 2017.

[19] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/292

[20] M. A. Boyacioglu, Y. Kara, and Ömer Kaan Baykan, "Predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: A comparative analysis in the sample of savings deposit insurance fund (SDIF) transferred banks in Turkey," *Expert Systems with*

Applications, vol. 36, no. 2, Part 2, pp. 3355–3366, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800078X

[21] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, Aug. 2017, pp. 1–6.

[22] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recogn.*, vol. 77, no. C, p. 354–377, may 2018. [Online]. Available: https://doi.org/10.1016/j.patcog.2017.10.013

[23] R. Z. Cabada, H. R. Rangel, M. L. B. Estrada, and H. M. C. Lopez, "Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems," *Soft Computing*, vol. 24, no. 10, pp. 7593–7602, 2020.

[24] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[25] E. Ostlin, H.-J. Zepernick, and H. Suzuki, "Macrocell path-loss prediction using artificial neural networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2735–2747, 2010.

[26] M. Ayadi, A. Ben Zineb, and S. Tabbane, "A uhf path loss model using learning machine for heterogeneous networks," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 7, pp. 3675–3683, 2017.

[27] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[28] W. Burgard, O. Brock, and C. Stachniss, *An Experimental Study of Exploiting Multipath Fading for Robot Communications*, 2008, pp. 289–296.

[29] R. Parasuraman, S. Caccamo, F. Baberg, and P. Ogren, "Crawdad kth/rss," 2022. [Online]. Available: https://dx.doi.org/10.15783/C7088F

[30] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional neural networks for patient-specific ECG classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 2608–2611.

[31] S. Sakib, T. Tazrin, M. M. Fouda, Z. M. Fadlullah, and N. Nasser, "A Deep Learning Method for Predictive Channel Assignment in Beyond 5G Networks," *IEEE Network*, vol. 35, no. 1, pp. 266–272, 2021, doi: 10.1109/MNET.011.2000301.

[32] ——, "An Efficient and Lightweight Predictive Channel Assignment Scheme for Multi-band B5G-Enabled Massive IoT: A Deep Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5285–5297, 2021, doi: 10.1109/JIOT.2020.3032516.

[33] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[34] Z. Md. Fadlullah and N. Kato, "HCP: Heterogeneous computing platform for federated learning based collaborative content caching towards 6g networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 112–123, 2022.

[35] Z. M. Fadlullah and N. Kato, "On smart IoT remote sensing over integrated terrestrial-aerial-space networks: An asynchronous federated learning approach," *IEEE Network*, vol. 35, no. 5, pp. 129–135, 2021.

[36] R. N. Clarke, "Expanding mobile wireless capacity: The challenges presented by technology and economics," *ERN: Models of Firms Capacity Choice & Growth (Topic)*, 2012.

[37] R. Dangi, P. Lalwani, G. Choudhary, I. You, and G. Pau, "Study and investigation on 5G technology: A systematic review," *Sensors (Basel, Switzerland)*, vol. 22, 2022.

[38] I. P. Chochliouros, A. S. Spiliopoulou, P. I. Lazaridis, A. Dardamanis, Z. D. Zaharis, and A. Kostopoulos, "Dynamic network slicing: Challenges and opportunities," *Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops*, vol. 585, pp. 47 – 60, 2020.

[39] S. O. Oladejo and O. E. Falowo, "5G network slicing: A multi-tenancy scenario," in *2017 Global Wireless Summit (GWS)*, 2017, pp. 88–92.

[40] R. Abhishek, S. Zhao, and D. Medhi, "SPArTaCuS: Service priority adaptiveness for emergency traffic in smart cities using software-defined networking," in *2016 IEEE International Smart Cities Conference (ISC2)*, 2016, pp. 1–4.

[41] L. Ma, X. Wen, L. Wang, Z. Lu, and R. Knopp, "An SDN/NFV based framework for management and deployment of service based 5G core network," *China Communications*, vol. 15, no. 10, pp. 86–98, 2018.

[42] P. Du and A. Nakao, "Deep learning-based application specific RAN slicing for mobile networks," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018, pp. 1–3.

[43] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019, pp. 0762–0767.

[44] S. Messaoud, A. Bradai, O. B. Ahmed, P. T. A. Quang, M. Atri, and M. S. Hossain, "Deep federated Q-learning-based network slicing for industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5572–5582, 2021.

[45] B. Brik and A. Ksentini, "On predicting service-oriented network slices performances in 5G: A federated learning approach," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 164–171.

[46] H. Chergui, L. Blanco, and C. Verikoukis, "Statistical federated learning for beyond 5G SLA-constrained RAN slicing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 2066–2076, 2022.

[47] A. K. Singh and K. Khoa Nguyen, "Joint selection of local trainers and resource allocation for federated learning in open RAN intelligent controllers," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1874–1879.

[48] R. Dangi, P. Lalwani, G. Choudhary, I. You, and G. Pau, "Study and investigation on 5g technology: A systematic review," *Sensors*, vol. 22, no. 1, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/1/26

[49] S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter-wave cellular wireless networks: Potentials and challenges," *Proceedings of the IEEE*, vol. 102, no. 3, pp. 366–385, 2014.

[50] M. Xiao, S. Mumtaz, Y. Huang, L. Dai, Y. Li, M. Matthaiou, G. K. Karagiannidis, E. Björnson, K. Yang, C.-L. I, and A. Ghosh, "Millimeter wave communications for future mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 9, pp. 1909–1935, 2017.

[51] X. Wang, L. Kong, F. Kong, F. Qiu, M. Xia, S. Arnon, and G. Chen, "Millimeter wave communication: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1616–1653, 2018.

[52] I. A. Hemadeh, K. Satyanarayana, M. El-Hajjar, and L. Hanzo, "Millimeter-wave communications: Physical channel models, design considerations, antenna constructions, and link-budget," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 870–913, 2018.

[53] G. R. MacCartney, J. Zhang, S. Nie, and T. S. Rappaport, "Path loss models for 5g millimeter wave propagation channels in urban microcells," in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 3948–3953.

[54] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. K. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li, F. Qian, and Z.-L. Zhang, "Lumos5g: Mapping and predicting commercial mmwave 5g throughput," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 176–193. [Online]. Available: https://doi.org/10.1145/3419394.3423629

[55] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *CoRR*, vol. abs/1812.00564, 2018. [Online]. Available: http://arxiv.org/abs/1812.00564

[56] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," 2018. [Online]. Available: https://arxiv.org/abs/1803.04311

[57] D. Minovski, N. Ogren, K. Mitra, and C. Ahlund, "Throughput prediction using machine learning in lte and 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 03, pp. 1825–1840, mar 2023.

[58] W. Na, B. Bae, S. Cho, and N. Kim, "Dl-tcp: Deep learning-based transmission control protocol for disaster 5g mmwave networks," *IEEE Access*, vol. 7, pp. 145 134–145 144, 2019.

[59] Y. Chen, X. Lin, T. Khan, M. Afshang, and M. Mozaffari, "5g air-to-ground network design and optimization: A deep learning approach," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–6.

[60] O. Nassef, W. Sun, H. Purmehdi, M. Tatipamula, and T. Mahmoodi, "A survey: Distributed machine learning for 5g and beyond," *Comput. Netw.*, vol. 207, no. C, apr 2022. [Online]. Available: https://doi.org/10.1016/j.comnet.2022.108820

[61] Q. Duan, S. Hu, R. Deng, and Z. Lu, "Combined federated and split learning in edge computing for ubiquitous intelligence in internet of things: State of the art and future directions," 2022. [Online]. Available: https://arxiv.org/abs/2207.09611

[62] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif, "A secure federated learning framework for 5g networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 24–31, 2020.

[63] B. S. Rawal, S. Patel, and M. Sathiyanarayanan, “Identifying ddos attack using split-machine learning system in 5g and beyond networks,” in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.

[64] A. Arfaoui, S. Cherkaoui, A. Kribeche, and S. M. Senouci, “Context-aware adaptive remote access for iot applications,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 786–799, 2020.

[65] S. M. A. Oteafy and H. S. Hassanein, “Leveraging tactile internet cognizance and operation via iot and edge technologies,” *Proceedings of the IEEE*, vol. 107, no. 2, pp. 364–375, 2019.

[66] Y. Han, B. D. Rao, and J. Lee, “Massive Uncoordinated Access With Massive MIMO: A Dictionary Learning Approach,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 1320–1332, Feb. 2020.

[67] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, “Massive access for 5g and beyond,” *CoRR*, vol. abs/2002.03491, 2020. [Online]. Available: https://arxiv.org/abs/2002.03491

[68] S. Fu and Y. Zhang, “CRAWDAD dataset due/packet-delivery,” Downloaded from https://crawdad.org/due/packet-delivery/20150401, Feb. 2020.