UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

South East European University
Faculty of Contemporary Sciences & Technologies

Adaptive Federated Learning for Resource-Constrained Devices

Prepared By:
M.Sc. Lamir Shkurti

Supervised By:
Prof. Dr. Mennan SELIMI

A Graduation Thesis submitted to the Faculty of Contemporary Sciences and Technologies at SEEU in partial fulfillment of the requirements for the degree of Doctor of Science in Computer Science

Tetova
March, 2025

Adaptive Federated Learning for Resource-Constrained Devices

A thesis submitted to the Faculty of Contemporary Sciences and Technologies

at South East European University (SEEU)

In partial fulfilment of the requirements for the degree of Doctor of Science

by

Lamir Shkurti

Thesis Committee:

Prof. Dr. Jaumin Ajdari, president _____

Prof. Dr. Mennan Selimi, member and mentor _____

Prof. Dr. Florije Ismaili, member _____

Prof. Dr. Xhemal Zenuni, member _____

Prof. Dr. Arsim Susuri, member _____

Abstract

The exponential growth of the Internet of Things (IoT) has created a network of interconnected devices generating huge amounts of data. Although this growth enables opportunities for the deployment of machine learning (ML) solutions, it also introduces significant challenges, particularly in resource-constrained environments where computational power, storage, and bandwidth are limited.

Federated Learning (FL), a decentralized ML paradigm, addresses key privacy concerns by enabling collaborative model training while keeping raw data localized on individual devices. FL ensures data privacy and enables decentralized model training; its application in heterogeneous and resource-constrained environments introduces significant challenges. Variability in computational capacity and data quality between devices can lead to CPU overload, excessive energy consumption, and inefficiencies in the training process. In addition, network instability, characterized by limited or fluctuating bandwidth, further complicates the exchange of model updates. These challenges highlight the need for adaptive mechanisms to dynamically adjust training parameters, ensuring efficient resource utilization and maintaining system performance across diverse environments.

This thesis presents four key components that enhance FL in resource-constrained environments: FederatedMesh, BACA (Bandwidth and CPU-Aware Adaptive FL), AdaptiveMesh (Adaptive Federated Learning for Resource Constrained Wireless Environments), and enhanced adaptive mechanisms. FederatedMesh is a collaborative FL system designed to facilitate medical data sharing in mesh networks while preserving privacy. BACA dynamically optimizes CPU usage and bandwidth allocation, ensuring efficient training without overloading devices. AdaptiveMesh is an adaptive FL algorithm that optimizes resource utilization and training efficiency by dynamically adjusting parameters based on CPU load, temperature, memory availability, network bandwidth, and model performance trends. Finally, enhanced adaptive mechanisms provide further improvements in workload distribution, ensuring stable training in het-

erogeneous and wireless environments. Extensive experimentation was conducted using physical IoT devices, such as Raspberry Pis, Mini PCs, and laptops, along with virtual machines in the Google Cloud. The results demonstrate that AdaptiveMesh outperforms naive, non-adaptive FL approaches by enhancing model accuracy, training stability, and resource efficiency. Specifically, AdaptiveMesh achieves significant reductions in CPU usage and bandwidth consumption while dynamically balancing computational loads across devices with varying capabilities. The algorithm's ability to prevent overheating and resource exhaustion ensures stable operation in environments with limited hardware resources and dynamic network conditions. In addition to introducing AdaptiveMesh, this research provides a comparative analysis of its performance against traditional FL approaches. Key metrics, including CPU utilization, temperature, training time, and model accuracy, are analyzed to highlight the benefits of adaptivity in FL. The findings emphasize that naive algorithms often result in inefficiencies, such as increased training time and device throttling due to unmanaged workloads, whereas AdaptiveMesh consistently ensures scalable and efficient training processes. This thesis combines the results of numerous studies to develop a framework for federated adaptive learning in resource-constrained environments. It makes a significant contribution to the rapidly evolving field of edge computing by showcasing the feasibility and practicality of implementing FL on real-world devices under heterogeneous conditions. Furthermore, it underscores the critical role of adaptive mechanisms in preserving data privacy, optimizing resource use, and enabling scalable ML in the healthcare and IoT domains.

Abstrakt

Zgjerimi i shpejtë i Internetit të Gjërave (IoT) ka çuar në krijimin e një rrjeti pajisjesh të ndërlidhura që gjenerojnë vëllime të mëdha të dhënash. Edhe pse kjo rritje ka krijuar mundësi për implementimin e zgjidhjeve të Machine Learning (ML), ajo gjithashtu paraqet sfida të rëndësishme, veçanërisht në ambiente me burime të kufizuara ku kufizimet e pajisjeve në fuqinë llogaritëse, kapacitetin e memories dhe gjerësinë e brezit janë të pranishme. Federated Learning (FL), një paradigmë e decentralizuar e ML, adreson shqetësimet kryesore të privatësisë duke mundësuar trajnimin bashkëpunues të modeleve duke e mbajtur të dhënat e papërpunuara të lokalizuara në pajisjet individuale. Megjithëse FL zgjidh çështjet e privatësisë dhe lejon trajnimin e decentralizuar të modeleve, aplikimi i tij në ambiente heterogjene dhe me burime të kufizuara sjell sfida të konsiderueshme. Ndryshueshmëria në kapacitetin llogaritës dhe cilësinë e të dhënave midis pajisjeve mund të çojë në mbingarkesë të CPU, konsum të tepërt të energjisë dhe joefikasitet në procesin e trajnimit.

Për më tepër, paqëndrueshmëria e rrjetit, e karakterizuar nga gjerësia e brezit të kufizuar, e ndërlikon më tej shkëmbimin e përditësimeve të modeleve. Këto sfida nënvizojnë nevojën për mekanizma adaptivë që të rregullojnë në mënyrë dinamike parametrat e trajnimit, duke siguruar shfrytëzim efikas të burimeve dhe ruajtjen e performancës së sistemit në ambiente të ndryshme.

Kjo tezë paraqet katër komponentë kryesorë që përmirësojnë FL në ambiente me burime të kufizuara: FederatedMesh, BACA (Bandwidth and CPU-Aware Adaptive FL), AdaptiveMesh (Adaptive Federated Learning for Resource Constrained Wireless Environments) dhe mekanizmat e përmirësuar adaptivë. FederatedMesh është një sistem bashkëpunues FL i krijuar për të lehtësuar shkëmbimin e të dhënave mjekësore në rrjetet mesh duke ruajtur privatësinë. BACA optimizon në mënyrë dinamike përdorimin e CPU-së dhe ndarjen e gjerësisë së brezit, duke siguruar trajnim efikas pa mbingarkuar pajisjet. AdaptiveMesh është një algoritëm adaptiv FL që optimizon shfrytëzimin e burimeve dhe efikasitetin e trajnimit duke rregulluar në mënyrë

dinamike parametrat bazuar në ngarkesën e CPU-së, temperaturën, disponueshmërinë e kujtesës, gjerësinë e brezit të rrjetit dhe trendet e performancës së modelit. Së fundi, mekanizmat e përmirësuar adaptivë ofrojnë përmirësime të mëtejshme në shpërndarjen optimale të ngarkesës së punës, duke siguruar trajnim të qëndrueshëm në ambiente heterogjene dhe pa tela.

Eksperimente të gjera janë kryer duke përdorur pajisje fizike IoT, si Raspberry Pi, Mini PC dhe laptopa, së bashku me virtual machines në Google Cloud. Rezultatet tregojnë se AdaptiveMesh tejkalon qasjet naive, jo-adaptive të FL duke përmirësuar saktësinë e modelit, qëndrueshmërinë e trajnimit dhe efikasitetin e burimeve. Në mënyrë specifike, AdaptiveMesh arrin reduktime të konsiderueshme në shfrytëzimin e CPU dhe konsumin e gjerësisë së brezit, ndërsa balancon në mënyrë dinamike ngarkesat llogaritëse në pajisje me aftësi të ndryshme. Aftësia e algoritmit për të parandaluar mbinxehjen dhe shterimin e burimeve siguron funksionim të qëndrueshëm në ambiente me burime harduerike të kufizuara dhe kushte dinamike të rrjetit.

Përveç prezantimit të AdaptiveMesh, ky hulumtim ofron një analizë krahasuese të performancës së tij kundrejt qasjeve tradicionale të FL. Metrikat kryesore, duke përfshirë shfrytëzimin e CPU, temperaturën, kohën e trajnimit dhe saktësinë e modelit, janë analizuar për të nxjerrë në pah përfitimet e adaptivitetit në FL. Gjetjet theksojnë se algoritmet naive shpesh rezultojnë në joefikasitet, si rritja e kohës së trajnimit dhe ulja e performancës së pajisjeve për shkak të ngarkesave të pakontrolluara, ndërsa AdaptiveMesh siguron vazhdimisht procese trajnimi të shkallëzueshme dhe efikase.

Kjo tezë kombinon rezultatet nga studime të shumta për të zhvilluar një kornizë të plotë për adaptive federated learning në ambiente me burime të kufizuara. Ajo jep një kontribut të rëndësishëm në fushën në zhvillim të edge computing duke demonstruar fizibilitetin dhe praktikën e implementimit të FL në pajisje të botës reale në kushte heterogjene. Për më tepër, ajo nënvizon rolin kritik të mekanizmave adaptivë në ruajtjen e privatësisë së të dhënave, optimizimin e përdorimit të burimeve dhe mundësimin e ML të shkallëzueshëm në domainet e shëndetësisë dhe IoT.

Апстракт

Брзиот раст на Интернетот на нештата (IoT) доведе до создавање на мрежа на поврзани уреди кои генерираат огромни количини на податоци. Иако овој раст создаде можности за имплементација на решенија за машинско учење, тој исто така носи значајни предизвици, особено во средини со ограничени ресурси каде ограничувањата во пресметковната моќ, капацитетот за складирање и пропусниот опсег се присутни. Федерирано учење, децентрализирана парадигма на МЛ, ги адресира главните загрижености за приватноста со овозможување на колаборативно обучување на модели додека ги чува необработените податоци локализирани на поединечните уреди. Иако ФЛ ги решава проблемите за приватност и овозможува децентрализирано обучување на модели, неговата примена во хетерогени и ресурсно ограничени средини носи значајни предизвици. Варијабилноста во пресметковниот капацитет и квалитетот на податоците меѓу уредите може да доведе до преоптоварување на CPU, прекумерна потрошувачка на енергија и неефикасност во процесот на обучување.

Покрај тоа, нестабилноста на мрежата, карактеризирана со ограничен или променлив пропусен опсег, дополнително го комплицира размената на ажурирањата на моделите. Овие предизвици ја нагласуваат потребата за адаптивни механизми кои динамички ги прилагодуваат параметрите за обучување, обезбедувајќи ефикасно искористување на ресурсите и одржување на перформансите на системот во различни средини.

Оваа теза го претставува AdaptiveMesh, адаптивен алгоритам FL дизајниран да го оптимизира искористувањето на ресурсите и да го подобри ефикасот на обучувањето во различни и ресурсно ограничени средини. AdaptiveMesh динамички ги прилагодува параметрите за обучување, во зависност од мерењата во реално време на уредите, како што се искористувањето на CPU, достапната меморија, пропусниот опсег на мрежата и трендовите во перформансите на моделот. Овие адаптивни механизми

спречуваат преоптоварување на уредите, намалување на времето за обука и одржување на оптимална распределба на ресурсите низ хетерогени уреди.

Се спроведени екстензивни експерименти со користење на физички IoT уреди, како Raspberry Pi, Mini PC и лаптопи, заедно со virtual machines во Google Cloud. Резултатите покажуваат дека AdaptiveMesh ги надминува наивните, неадаптивни пристапи на FL со подобрување на точноста на моделот, стабилноста на обучувањето и ефикасноста на ресурсите. Конкретно, AdaptiveMesh постигнува значително намалување на искористувањето на CPU и потрошувачката на пропусен опсег, додека динамички ги балансира пресметковните оптоварувања на уредите со различни капацитети. Способноста на алгоритмот да спречи прегревање и исцрпување на ресурсите обезбедува стабилна работа во средини со ограничени хардверски ресурси и динамични мрежни услови.

Покрај воведот на AdaptiveMesh, ова истражување нуди компаративна анализа на неговите перформанси во однос на традиционалните пристапи на FL. Клучните метрики, вклучувајќи искористување на CPU, температура, време на обучување и точност на моделот, се анализирани за да се истакнат придобивките на адаптивноста во FL. Наодите нагласуваат дека наивните алгоритми често резултираат со неефикасност, како што се зголемено време на обучување и намалување на перформансите на уредите поради неконтролирани оптоварувања, додека AdaptiveMesh постојано обезбедува скалабилни и ефикасни процеси на обучување.

Оваа теза ги комбинира резултатите од бројни студии за да развие сеопфатна рамка за adaptive federated learning во ресурсно ограничени средини. Таа дава значаен придонес во областа на edge computing што брзо се развива, покажувајќи ја изводливоста и практичноста на имплементацијата на FL на уреди од реалниот свет во хетерогени услови. Понатаму, таа ја нагласува критичната улога на адаптивните механизми во зачувувањето на приватноста на податоците, оптимизирањето на искористувањето на ресурсите и овозможувањето на машинско учење што може да се скалира во домените на здравствената заштита и IoT.

Declaration

I hereby declare my Dissertation

Adaptive Federated Learning for Resource-Constrained Devices

has been entirely written by me and has not been previously submitted for a degree or diploma at any university.

Lamir Shkurti

ATTESTATION

I, **Martin Domgjoni**, a **Court Interpreter for English–Albanian and vice versa, license No.186**, hereby confirm that I have thoroughly reviewed the PhD thesis titled **"Adaptive Federated Learning for Resource-Constrained Devices"** of the candidate Lamir Shkurti. With my signature and stamp, I attest that the thesis is fully and accurately written in the English Language, and I confirm that it satisfies all the requirements for defense and publication.

I attest that the thesis in Albanian Language is fully and accurately translated in English Language.

Martin Domgjoni A Court Interpreter for English-Albanian and vice versa No.186
Address: 20000 Prizren, Republic of Kosovo
Tel.: +383 44 851 841 / +383 48 851 841
E-Mail: martindomgjoni@gmail.com

REPUBLIKA E KOSOVËS PËRKTHYES GJYQËSOR MARTIN DOMGJONI SHQIP-ANGLISHT ANGLISHT-SHQIP REPUBLIKA KOSOVO

Prizren, 17.03.2025

Acknowledgments

Completing this Ph.D. has been a long and challenging journey in my life. It has been a period of continuous learning, perseverance, and personal growth, filled with both moments of difficulty and triumph. First and foremost, all praise and gratitude belong to Allah, who enabled me to complete my PhD. In addition, I am immensely grateful to the many individuals whose unwavering support, encouragement, and guidance made this achievement possible. Their contributions have been invaluable and I am deeply grateful for their role in my journey.

Firstly, I want to sincerely thank my advisor, Prof. Dr. Mennan Selimi, for his encouragement, important advice, and support during my doctoral studies. His deep knowledge, patience, and dedication were instrumental in the formation of this research. His willingness to share expertise and provide support at each stage of this journey made the research process much more enriching.

I am very thankful to my family for their constant support, patience, and belief in me throughout this journey. Their constant encouragement has been my source of strength and motivation, especially during the most challenging times. Their love and support have provided me with the foundation I needed to pursue my dreams with confidence and determination.

To my partner, my greatest supporter, I am forever grateful. Your patience, understanding and unwavering faith in me have given me the courage to overcome every obstacle. Without your support, this journey would have been much more difficult.

Finally, to my wonderful children, who have been my greatest source of joy and inspiration, this dissertation is dedicated to you. Your love and presence have given me the strength to keep pushing forward, even when the road seems long. I am deeply grateful for the moments I missed while working on this PhD and I hope my perseverance serves as an inspiration for you to pursue your dreams with determination and passion.

This PhD journey has been one of resilience and growth. I am profoundly thankful to everyone who has been part of this experience, and I carry their support with me as I step into the next chapter of my life.

Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
AdaptiveMesh	Adaptive Federated Learning for Wireless Mesh Environments
ANOVA	Analysis of Variance
BACA	Bandwidth and CPU-Aware Adaptive Federated Learning
CNN	Convolutional Neural Network
CPU	Central Processing Unit
ECDF	Empirical Cumulative Distribution Function
DNN	Deep Neural Network
FedAvg	Federated Averaging
FedAda	Federated Adaptive Training
FedALA	Adaptive Local Aggregation for Federated Learning
FL	Federated Learning
GPU	Graphics Processing Unit
IoT	Internet of Things
LSTM	Long Short-Term Memory
ML	Machine Learning
NAIVE	Non-Adaptive Federated Learning Algorithm
NN	Neural Network
RAM	Random Access Memory
RNN	Recurrent Neural Network
RTT	Round Trip Time
SVM	Support Vector Machine

Part I. Problem Context and Objectives

Chapter 1

Introduction

1.1 Introduction

The total number of devices interconnected in the Internet of Things (IoT) is projected to exceed 30.9 billion by 2030. These devices generate a large amount of information every day. These data can be highly private or sensitive coming from different IoT devices and organizations (hospitals, airports, industry, etc.) and may have the need for sharing, but privacy must be preserved. To have accurate results and to train the Machine Learning (ML) model, we must have a lot of quality data. Traditionally, ML methods collect data from connected devices, store it on a cloud server, train the model, and then apply it to all devices. This method has some limitations, such as privacy and communication between the device and the main server. In addition, IoT devices are often distributed in diverse environments, making it challenging to maintain a stable Internet connection, which can introduce problems in real-time applications. Furthermore, traditional ML methods raise concerns related to power consumption and efficiency in resource-limited devices.

FL is an advanced ML approach that tackles these challenges by allowing model training on data stored directly on the device, eliminating the requirement for centralized data collection. Instead of raw data, only model updates are sent to the central server. This approach guarantees information protection by keeping personal information on the device, significantly reducing the chance of security breaches. By reducing the amount of data transferred, FL minimizes bandwidth usage and enhances privacy while still allowing continuous learning and model updates. Also, FL can apply to many model architectures such as CNN, RNN, LSTM, etc.

Edge computing expands on cloud computing by enabling data processing on small devices,

such as wearables and IoT gateways, located near the boundary of the system where information comes from. Local processing on these edge devices can improve the response times of cloud services and reduce bandwidth consumption by transmitting less data to the cloud [1][2]. Edge computing is already widely implemented in both industrial and consumer-focused applications, allowing certain ML and AI tasks to be shifted from the cloud to edge devices.

The integration of ML models into low-power edge devices is increasingly common. Even small microcontroller-based systems can now perform basic tasks using models previously trained [3]. However, developing ML models requires considerably more computational resources, which presents significant difficulties for systems with limited capabilities. GPUs outperform CPUs in terms of speed and efficiency; they are often unavailable in low-capacity systems such as compact mini-PCs or single-chip computers (SCCs). Consequently, central processors are frequently tasked with handling the intensive computations required for model training, leading to much longer training durations compared to high-performance systems. These prolonged training times are impractical for time-sensitive applications, which require adaptive optimization techniques to improve efficiency.

The rise in the popularity of wearable devices and smartphones has led to the widespread use of intelligent learning applications. These devices continuously gather data related to users' daily activities, providing valuable insight to improve health monitoring and lifestyle management. The effectiveness of intelligent healthcare systems is significantly dependent on the ability to develop ML models using extensive data sets collected from smart wearables [4][5]. However, traditional ML methods, which rely on centralized data collection, face numerous challenges, including privacy issues, security risks, communication overhead, and latency problems. In wearable systems, privacy is a major concern when deploying data analytics algorithms. Users are often unwilling to share their personal data with cloud-based ML services. Figure 1.1 presents various devices that are resource-constrained and capable of performing ML tasks, such as Raspberry Pi boards, Arduino platforms, and tiny ML devices.

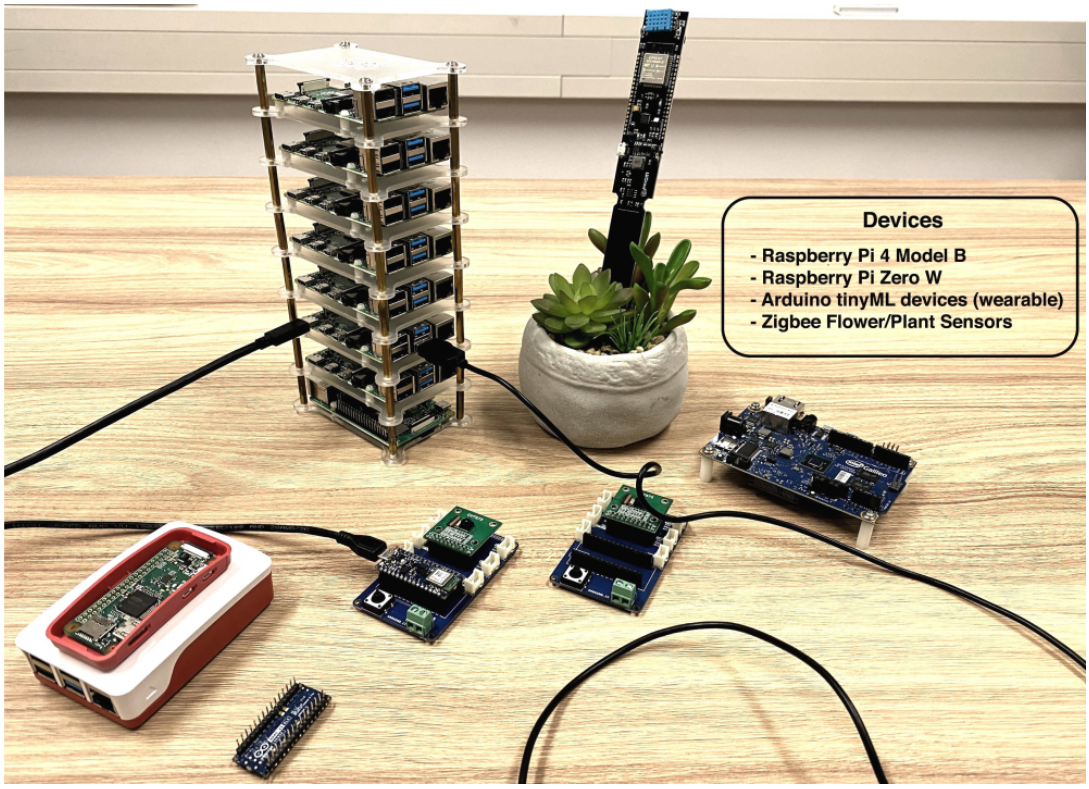

Figure 1.1: Low-constrained devices in the IoT world. Source: SEEU DSG Lab

FL is a distributed ML approach that enables model training on multiple resource-constrained mobile devices, such as computational power, storage capacity, energy, and bandwidth. This decentralized approach improves privacy, minimizes communication overhead, and reduces processing latency [6] [7]. Unlike traditional ML approaches, FL ensures that models are continuously updated without requiring raw data transfers, making it particularly suitable for IoT and edge computing scenarios. This decentralized approach operates through an iterative process, which can be summarized as follows.

- The main server initializes the universal model and distributes it among all worker nodes.
- Every worker node trains the model locally using its own data set and transmits the updated model attributes back to the main server.
- The server aggregates the updates from all the worker units to refine the shared model.
- This iterative method continues until a set number of training rounds is completed or convergence is achieved.

Wireless mesh networks can be used to support FL in these edge devices, but this can be challenging due to several factors [8]. These challenges include limited resources on devices with low computational power, unstable network performance, variations in device capabilities, and security vulnerabilities.

Training on such constrained devices requires a significant duration and frequently utilizes the full CPU capacity of the device. For FL to function effectively, the learning procedure should not solely aim for high accuracy but also prioritize minimizing training duration and optimizing efficient use of resources.

These requirements introduce significant challenges, especially in coordinating and aggregating model updates across devices within the FL system. Furthermore, ensuring stable and reliable communication between devices remains a significant challenge [9].

This dissertation aims to bridge these gaps by exploring FL deployment within a real-world wireless mesh system with restricted resources. We investigated the feasibility and performance of FL on IoT devices with restricted connectivity and hardware capabilities, linked through a wireless mesh system. Our objective is to provide fresh perspectives that enhance the practicality of FL on commonly used, resource-limited devices. We focus on optimizing FL performance for such devices and provide practical data on resource utilization (CPU, memory, network), along with recommendations for optimal training configurations to ensure efficient training on these constrained devices.

In the initial research phase, a secure and privacy-preserving FL system was developed and deployed at the edge of IoT networks. Experiments were conducted using Raspberry Pi devices, MiniPCs, and laptops to evaluate accuracy, CPU usage, RAM consumption, and network performance.

The computational performance of such limited resource devices varies depending on their hardware configurations and concurrent workload. In wireless environments, factors such as network bandwidth limitations—due to device location or fluctuating network conditions—and power consumption can introduce delays in the training workflow, affecting the general rate of learning of the global model [10], [11].

Although FL has demonstrated efficiency in decentralized scenarios, existing algorithms often face challenges in heterogeneous (e.g., wireless) and resource-constrained environments. These challenges are particularly pronounced in small, energy-limited devices, such as IoT and mobile devices, which cannot sustain long training periods without negatively impacting per-

formance. *Nonadaptive algorithms typically fail to manage CPU loads effectively, resulting in temporary shutdowns or significant performance drops due to resource overload*. As the amount of training information and the number of rounds increase, the computational demands on individual devices also increase, potentially causing inefficiencies as devices attempt to manage heat or workload by throttling performance [12].

In resource-constrained environments, intensive use of CPU resources can reduce the battery life of IoT and mobile devices, necessitating careful management of computational tasks to balance performance with energy consumption. This highlights the need for energy-efficient computing strategies, including adaptive task allocation and low-power hardware solutions. Ensuring efficient use of limited energy resources in IoT and edge devices emphasizes the need for energy-aware computing techniques such as task offloading, low power hardware, and dynamic resource allocation [13]. Another key consideration in FL is the balance between the accuracy of training and the use of resources. The increase in the number of training rounds and the expansion of the training data set enhance the precision of the model, but also prolong the training time and increase the risk of device overuse. Studies have highlighted this trade-off, showing that while accurate results can be achieved, resource and power constraints must be carefully managed to prevent performance drop in such environments [14]. Managing these factors is crucial to maintaining the longevity of the device and ensuring continuous and reliable operation. Implementing adaptive mechanisms that adjust the training workload based on real-time device conditions, such as CPU utilization thresholds, can mitigate these risks, although potentially at the cost of reduced accuracy [15].

These factors complicate the training process, as the devices may struggle to maintain performance. Fluctuating bandwidth often disrupts data transmission, leading to longer training times or lost connections. Consequently, standard FL algorithms with fixed parameters are unable to adapt to these variations, resulting in inefficiencies in both computational and network resources. Recent literature emphasizes the importance of adaptable FL algorithms that can dynamically respond to the limitations of resource-limited environments [16], [17].

Considering the above-mentioned challenges, adaptive FL is crucial because it helps models to perform well with different types of data on various devices with different hardware capacities. Since data on each device can be unique, adaptive algorithms allow models to adjust and perform better despite these differences [18]. This ensures that the learning process remains efficient even with varying device capabilities, internet connectivity, and resource con-

straints such as CPU load [19]. However, the effectiveness and efficiency of FL algorithms vary significantly based on computational load, memory requirements, and environmental conditions such as temperature and battery consumption.

To address these issues, we present the **AdaptiveMesh** algorithm (Adaptive Federated Learning for Wireless Mesh Environments). AdaptiveMesh dynamically modifies the training settings in response to the characteristics of the client and the network conditions. It evaluates CPU utilization, CPU temperature, training time, and bandwidth, adapting the training load to prevent overuse and ensure optimal performance [20]. This approach helps maintain device longevity and stable operation, even under varying conditions, making FL more efficient and scalable. This thesis introduces the AdaptiveMesh algorithm, which is specifically developed to improve training efficiency in heterogeneous wireless environments. The algorithm dynamically adapts the training parameters according to the client's performance metrics, with the aim of improving both efficiency and accuracy in these challenging settings [21], [22]. The objective is to design a flexible framework for FL users that addresses the diverse nature of connected devices in terms of processing power, storage, and wireless network capacity. [23], [24].

This work introduces an intelligent mechanism that dynamically adjusts training parameters, including the count of training images and the accuracy of the model, based on the conditions of the system. To improve training effectiveness, the adaptive algorithm monitors precision, processing load, and storage consumption during training iterations. It analyzes trends in model efficiency, including performance metrics and processing demand, to detect inefficiencies. When performance stagnates or CPU consumption exceeds a predefined threshold, the algorithm dynamically optimizes resource allocation. This adaptive resource management ensures that training remains efficient across all clients. By continuously adjusting the workload, AdaptiveMesh helps to maintain efficient FL operations in dynamic environments.

This thesis advances the study of FL and adaptive ML within resource-constrained environments through:

1. **Developing and deploying a Secure and Privacy-Focused FL System**: A secure and privacy-focused FL system was implemented at the edge of IoT networks. Tests were conducted to measure accuracy, CPU usage, RAM consumption, and network performance using Raspberry Pi devices, Mini PCs, and laptops. Based on these results, optimization strategies were developed for practical use on low-power devices, ensuring

the efficient operation of FL systems while preserving data privacy.

2. **Proposing the BACA Algorithm**: An FL optimization algorithm specifically focused on minimizing CPU load and bandwidth usage in wireless environments. BACA dynamically adapts training loads in response to real-time resource metrics, ensuring efficient model training and reducing computational overhead on low-power devices.

3. **Introducing and implementing AdaptiveMesh:** An FL framework that dynamically optimizes training parameters by adapting to device performance metrics and network conditions, ensuring efficient and scalable training in resource-constrained environments. AdaptiveMesh offers a flexible approach to managing heterogeneous devices, improving both training efficiency and model accuracy, while preventing device overloading.

4. **Conducting Experimental Validation on Real and Virtual Devices:** Extensive testing has been carried out on various resource-limited devices (e.g., Raspberry Pi, Mini PCs, and laptops) alongside virtual machines hosted in commercial cloud platforms. This study conducts a comparative assessment of these algorithms when executed in virtualized environments based on physical hardware versus cloud based environments, providing valuable observations on the balance between accuracy, duration of training, and resource utilization in various set-ups of systems.

5. **Performing a Comprehensive Analysis of Resource Usage and Adaptivity:** This research offers a detailed investigation of the balance between model accuracy, training iterations, processor load, and power consumption, delivering practical insights to improve FL efficiency. The flexibility of the proposed algorithms ensures their effectiveness in dynamic environments, accommodating variations in hardware capabilities and network performance.

The primary goal of this doctoral dissertation is to create an adaptive FL framework designed to tackle the challenges of deploying FL in resource-limited settings, with a specific focus on IoT devices and wireless networks. Current FL techniques often struggle with issues such as device heterogeneity, varying network conditions, and computational limitations, which result in inefficiencies and performance degradation.

The research has made notable progress, including peer-reviewed publications:

- **BACA Algorithm:** This algorithm focuses on optimizing bandwidth and CPU usage in wireless FL environments.
  - Key Results: Optimization: BACA adjusts training based on real-time CPU and bandwidth conditions, improving resource use, and preventing overloads.
  - Experimental Results: The tests on IoT devices showed increased model accuracy and reduced computational load, highlighting the effectiveness of BACA for edge computing applications.
- **AdaptiveMesh Algorithm:** Extends BACA by introducing additional adaptive mechanisms. This algorithm improves FL in environments with varying device capabilities by changing training settings based on resource measures such as CPU utilization, temperature, bandwidth, training time, and accuracy trends.
  - Key achievements: Dynamic Adaptation: AdaptiveMesh optimizes training by adjusting the image and epoch counts according to CPU, temperature, memory, bandwidth, and accuracy metrics, reducing overheating and improving efficiency.
  - Experimental Validation: Tests on low-power devices like the Raspberry Pi and MiniPCs show improved resource allocation, better model stability, and reduced CPU usage.
- The research expands by comparing the performance of AdaptiveMesh with a naive non-adaptive algorithm, as well as two adaptive approaches, FedAda and FedALA. The comparison focuses on key metrics such as CPU utilization, temperature, accuracy, training time, and network bandwidth. Our study examines the effectiveness of these algorithms on both physical devices, including Raspberry Pis, Mini PCs, and laptops, as well as virtual machine instances in the Google Cloud. The results demonstrate that AdaptiveMesh outperforms the naive approach and shows competitive advantages over FedAda and FedALA in terms of resource efficiency, thermal management, and adaptability to heterogeneous environments.

1.2 Problem Statement

Although FL solves the problem of privacy, communication, and generalization. However, a significant gap remains in assessing the practicality of this technique within IoT-enabled distributed systems composed of devices that are limited in resources. Although FL considerably reduces the volume of data transmitted to cloud servers, communication is still required to send training updates for each round. Therefore, optimizing the number of update rounds is crucial. Furthermore, despite FL being inherently designed to strengthen privacy, it is crucial to prevent confidential data from being unintentionally disclosed to unauthorized entities. Addressing these challenges is critical to making FL both practical and secure for IoT-enabled systems with limited resources.

Additionally, although FL is designed to maintain data privacy by keeping information decentralized on client devices, it still requires mechanisms to ensure that sensitive information is not accidentally exposed to other participants. This challenge becomes even more crucial in heterogeneous environments, where devices differ significantly in computational capabilities and the quality of their locally available data.

FL systems also face challenges related to resource heterogeneity, where devices exhibit significant variations in computational power, memory capacity, and energy availability. Low-capacity devices often experience performance bottlenecks, such as CPU overload, memory exhaustion, or excessive energy consumption, especially during intensive training tasks. These problems are further exacerbated when multiple rounds of training are required, leading to potential overheating or device failures.

Another critical challenge lies in the variability of network conditions. Limited or fluctuating bandwidth creates synchronization issues between devices and central servers, delaying model updates, and prolonging training time. Standard FL algorithms, which operate with fixed parameters, often fail to adapt to such dynamic conditions, resulting in inefficiencies and incomplete participation from resource-constrained devices.

FL operates on a distributed computing infrastructure, which naturally introduces heterogeneity in several forms:

- **Data Volume and Integrity:** Variation in local data availability and reliability between nodes may greatly influence the effectiveness of the global model. Some devices may have abundant, high-quality data, while others may have limited or noisy data sets. This

imbalance makes it difficult to ensure uniform model improvements across devices.

- **Hardware Variability:** The computational capacity of each participating device can differ drastically. Devices like Raspberry Pi, MiniPCs, and laptops have varying CPU, memory, and energy constraints, which can slow down the training process and create bottlenecks during model aggregation and communication.

- **Network Limitations:** Some nodes may encounter bandwidth restrictions when transmitting the trained model, either due to consistent network limitations or varying connectivity conditions. The heterogeneity of devices and data makes the problem even more complex, since we need the FL components (server and clients) to be adaptive.

The heterogeneity in the data, devices, and network conditions creates a complex problem that requires the adaptability of both the FL server and the clients. Without appropriate adaptive mechanisms, certain clients may fall behind due to their limited resources, leading to suboptimal training or even exclusion from the learning process.

Moreover, non-adaptive algorithms exacerbate these disparities, as they lack the flexibility to adjust workloads and resource demands dynamically. Consequently, such algorithms risk overloading resource-constrained devices or underutilizing powerful devices, undermining the efficiency and accuracy of the global model.

Furthermore, as IoT devices grow increasingly popular in important industries such as healthcare, smart cities, and environmental monitoring, real-time processing and responsiveness become critical. FL systems must balance trade-offs between training accuracy, resource utilization, and communication efficiency while maintaining strict data privacy and security protocols. Achieving this requires advanced adaptive mechanisms capable of dynamically modifying training parameters in real time based on device and network metrics.

Therefore, there is a critical need for FL systems that can dynamically adjust training parameters, such as the number of training images, epochs, or update frequencies, considering the real-time performance of each device. This adaptability allows all participating devices, regardless of their hardware constraints, to effectively contribute to the global model while ensuring efficient resource utilization and system balance. Addressing these challenges will enable the effective deployment of FL in diverse, resource-constrained settings, bridging the gap between theoretical research and real-world application. To address these challenges in FL, it is crucial to identify the key limitations that impact performance, scalability, and resource

efficiency. Below, we outline the major challenges faced by FL systems in resource-constrained environments:

- Privacy and Security: Despite FL's inherent privacy benefits, challenges remain in ensuring data confidentiality and protecting against adversarial attacks. Techniques like differential privacy and homomorphic encryption need further refinement to balance privacy and model accuracy.

- Communication Overhead: FL requires frequent communication between devices and the central server, which can strain network resources, especially in bandwidth-constrained environments. Optimizing communication protocols and reducing the size of model updates are critical areas of improvement.

- Device and Data Heterogeneity: The variability in computing power, memory, and data distribution between devices complicates the training process. Developing more robust algorithms that can handle non-IID data and diverse device capabilities is essential.

- Scalability: As the number of participating devices increases, managing the coordination and aggregation of model updates becomes increasingly complex. Scalable FL frameworks that can efficiently handle large-scale deployments are needed.

- Energy Efficiency: FL can be energy-intensive, particularly for battery-powered IoT devices. Energy-aware algorithms and hardware optimizations are necessary to extend the life of the device.

1.3 Purpose of study

This thesis aims to explore the FL process in a heterogeneous environment where devices differ in their computational capacities and workloads. The main goal is to propose a flexible framework for the FL server which effectively utilizes this variability by dynamically adjusting the workload distribution among participating nodes during training.

The proposed approach involves creating an FL server that, in each training round, determines the optimal workload capacity for each client node based on real-time resource availability (such as CPU, memory, and bandwidth) within a given time interval.

This adaptive mechanism will allow the server to send individualized training instructions to each client, ensuring a more balanced and streamlined training process. Using the unique capabilities of individual devices, this method seeks to enhance the overall effectiveness of the global model and ensure that all devices contribute meaningfully to the learning process, regardless of their resource constraints.

To assess the practicality of this method, a data set of health data collected from multiple hospitals will be utilized, with an emphasis on real-world applications within an IoT setting. Furthermore, the effectiveness of the proposed system in maintaining data privacy will be analyzed, ensuring that confidential information is protected throughout the training phase. The evaluation will be conducted on energy-efficient devices, including Raspberry Pi and other hardware with limited resources, to assess essential performance indicators, such as training duration, network overhead, energy usage, and memory consumption. The experiments will take place in IoT environments where managing communication load and protecting data privacy are critical factors.

The core aim of this thesis is to create and deploy an adaptive FL platform that can operate in real time and perform efficiently on devices with limited resources. By addressing challenges related to heterogeneity and resource limitations, this work seeks to support the creation of scalable, secure FL systems for practical deployment in IoT applications.

1.4 Research questions

We have formulated these research questions:

- **RQ 1:** Can FL be beneficial for scenarios with distributed low-cost edge devices and what factors affect FL performance?

  This research question deals with the feasibility of FL in environments with low-cost edge devices, such as Raspberry Pi and MiniPCs. The study demonstrates that FL can be beneficial through the implementation of FederatedMesh, a system designed for devices with limited resources. FederatedMesh successfully enables collaborative model training while preserving data privacy and optimizing resource usage. However, FL performance is affected by factors such as device heterogeneity, network bandwidth, CPU utilization, and data distribution.

- **RQ 2:** Does reducing the number of FL training rounds and epochs affect the accuracy of the model and energy consumption? What is the trade-off between rounds and epochs? The trade-off between model complexity and performance?

  The experiments show that reducing rounds and epochs can lower energy consumption but may also reduce model accuracy. Adaptive algorithms like AdaptiveMesh balance this trade-off by dynamically adjusting the number of epochs and training images based on device capabilities, ensuring efficient resource use without significantly compromising accuracy.

- **RQ 3:** How can we design adaptive communication strategies that allow low-resource clients to participate efficiently in FL while minimizing the use of network bandwidth, CPU, and energy?

  The proposed AdaptiveMesh and BACA algorithms address this by dynamically adjusting training parameters (e.g., epochs, batch sizes, and image counts) based on real-time metrics like CPU usage, temperature, and bandwidth. This ensures efficient participation of low-resource clients while minimizing network and computational overhead.

- **RQ 4:** How does optimizing algorithms improve FL efficiency in heterogeneous wireless environments?

The study shows that optimized algorithms like AdaptiveMesh and BACA significantly improve FL efficiency by dynamically balancing computational loads based on the performance and capacity of each device. This adaptive approach reduces communication overhead, prevents overheating, and ensures stable training in diverse environments. By modifying workloads to device capabilities, AdaptiveMesh and BACA optimize resource utilization and improve overall system performance.

- **RQ 5:** What impact does algorithm optimization have on CPU utilization, network bandwidth, and model accuracy in resource-constrained devices?

  The experiments demonstrate that AdaptiveMesh and BACA reduce CPU usage and bandwidth consumption while maintaining acceptable model accuracy. By dynamically adjusting training parameters, these algorithms prevent overheating and resource exhaustion, ensuring stable operation on resource-constrained devices.

1.5 Hypothesis

In this research study, we test the following hypotheses:

- **Hypothesis 1.** Device and network heterogeneity negatively impact FL performance on low-cost edge devices.
- **Hypothesis 2.** Reducing the number of FL training rounds and epochs negatively impacts model accuracy.
- **Hypothesis 3.** Offloading computation from low-resource clients to other good-performing clients can reduce CPU usage and energy consumption while maintaining communication efficiency.
- **Hypothesis 4.** Algorithm optimization significantly enhances FL efficiency in heterogeneous wireless environments by reducing communication overhead, balancing computational loads, and improving training time.
- **Hypothesis 5.** Optimized algorithms minimize CPU utilization and network bandwidth consumption while maintaining or improving model accuracy in resource-constrained devices, compared to non-optimized algorithms.

1.6 Important of thesis

In an era where technology is advancing at an unprecedented rate, concerns about privacy and data protection have become increasingly significant. The idea of adaptive FL has emerged as an innovative and key approach to address challenges in critical fields such as healthcare, the Internet of Things, and other resource-limited environments. These sectors require ML systems that are efficient, scalable, and secure, ensuring data protection without exposing sensitive information.

Adaptive FL integrates fundamental FL principles with adaptability, providing an effective method for training ML models across decentralized networks. By integrating mechanisms that adapt to dynamic conditions, changing data distributions, and diverse computational capabilities, the system ensures consistent optimal performance over time. This adaptability allows models to evolve and improve regardless of variations in hardware or network conditions. This thesis is dedicated to developing adaptive FL algorithms that improve efficiency, flexibility, and security in constrained environments. Its significance can be observed in several key aspects:

- **Efficiency and Adaptability:** By intelligently modifying training workloads based on real-time resource availability, including CPU, usage, temperature, memory, and network bandwidth. Adaptive algorithms guarantee that devices with limited resources can still participate efficiently. This approach enables the maintenance of high model accuracy and stable convergence in heterogeneous environments.

- **Privacy and Security:** As concerns about data privacy continue to increase, especially in sensitive fields such as healthcare, this research provides a framework that balances privacy preservation with efficient learning. Decentralized training eliminates the need to share raw data and ensures that privacy remains intact, even as models improve.

- **Real-World Relevance:** The adaptive algorithms developed in this thesis have practical applications in real-time systems, including healthcare monitoring, smart city infrastructure and IoT devices. Their efficiency in low-power devices makes them ideal for deployment in constrained environments where computational resources are limited.

By addressing the current challenges in FL for heterogeneous and resource-constrained environments, this thesis contributes to the advancement of scalable and privacy-preserving

ML systems. Its impact extends to a wide range of real-world applications, ultimately helping to build smarter and more efficient systems in healthcare, smart cities, and beyond.

1.7 Thesis contribution

This thesis makes several key contributions to the field of FL, particularly in the context of heterogeneous and resource-constrained environments. The research introduces novel adaptive algorithms and system optimizations that enhance FL efficiency, stability, and scalability. The main contributions are as follows:

- Development of AdaptiveMesh for Resource-Constrained Federated Learning Proposed and implemented AdaptiveMesh, an adaptive FL framework designed to dynamically adjust workload distribution based on real-time resource availability (CPU, memory, bandwidth). Demonstrated its effectiveness in optimizing FL performance, reducing training inefficiencies, and maintaining model accuracy despite device and network heterogeneity.

- Introduction of Adaptive Workload Management in FL Design an adaptive algorithm that enables the FL server to assign training tasks to client devices based on their real-time resource availability, allowing low-resource clients to contribute meaningfully without overloading their systems. Validated that offloading computations from low-power devices to more capable ones significantly reduces CPU usage and energy consumption while maintaining communication efficiency.

- Comparative Evaluation of FL in Physical and Virtualized Environments Conducted a comprehensive analysis comparing FL performance on physical edge devices (e.g., Raspberry Pi) versus cloud-based virtual machines. AdaptiveMesh demonstrated flexibility in both settings, ensuring stable operation on low-power devices while leveraging the scalability of cloud environments.

- Impact on Real-World IoT Applications Showcased the applicability of adaptive FL in mobile and industrial IoT settings, including use cases in healthcare monitoring, personalized learning, and predictive maintenance. The authors highlighted that adaptive mechanisms are crucial to ensure FL efficiency in decentralized machine learning applications where devices have varying computational capacities.

1.8 Publications

The research papers published as part of this dissertation include:

1. L. Shkurti, M. Selimi, and A. Besimi, “Federatedmesh: Collaborative federated learning for medical data sharing in mesh networks,” in Collaborative Computing: Networking, Applications and Worksharing. $https://doi.org/10.1007/978-3-031-54531-3_9$

2. L. Shkurti and M. Selimi, ”BACA: Bandwidth and cpu-aware adaptive federated learning for wireless environments,” in 2024 13th Mediterranean Conference on Embedded Computing (MECO). $https://doi.org/10.1109/MECO62516.2024.10577810$

3. L. Shkurti and M. Selimi, ”AdaptiveMesh: Adaptive Federate Learning for Resource-Constrained Wireless Environments” in International Journal of Online and Biomedical Engineering (iJOE). $https://doi.org/10.3991/ijoe.v20i14.50559$

4. Lamir Shkurti and Mennan Selimi, ”Enhancing Adaptive Behavior in Federated Learning: The AdaptiveMesh Algorithm for Interactive Mobile and Bandwidth-Limited Resource-Constrained Wireless Environments” in International Journal of Interactive Mobile Technologies (iJIM).

1.9 Structure of thesis

The structure of this thesis is organized as follows:

- **Chapter 1: Introduction** - This chapter introduces the problem domain, providing an overview of the thesis, including the problem statement, the purpose of the study, the hypothesis, research questions, the importance of the thesis, the thesis contribution, and the publications.

- **Chapter 2: Technical background** - The second chapter delves into the fundamental concepts of ML and provides an overview of key elements that will be used throughout the project. In addition, it explores existing research on distributed ML and examines how data distribution has been addressed in previous studies.

- **Chapter 3: System Model - Design and Implementation** - This chapter outlines the design considerations and describes the implementation details of the proposed FL system. It focuses on the structural aspects and methodologies used in this thesis.

- **Chapter 4: Adaptive Federated Learning** - This chapter focuses on an adaptive FL system that will implement an algorithm where the server sends individual training instructions to each client, which can accept the instructions or not based on the performance of the client to lead to higher-performing global models.

- **Chapter 5: Experiments and results** - This chapter discusses the tests conducted as part of the thesis, presenting the evaluation metrics and results obtained from these experiments.

- **Chapter 6: Conclusions and future work** - The final chapter summarizes the findings of this thesis, addresses research questions, and provides insights into potential future research directions and improvements.

- **References** – All references and resources that are the basis of this thesis, which is the starting point of our proposed implementation.

Part II. Research Context and State-of-the-Art

Chapter 2

Technical Background and Related Work

2.1 Introduction

Machine learning (ML) serves as a fundamental component of modern AI, equipping systems with the capability to extract information insights and generate decisions or forecasts without the need for explicit programming. With the continuous expansion of the IoT, large amounts of distributed information are produced, creating the need for efficient methods to process and analyze information. FL has gained recognition as an effective approach that decentralizes model training, enhancing data privacy, and minimizing communication overhead. Rather than sharing raw data, FL operates by transmitting model updates, thereby preserving privacy [25]. This section offers a summary of key ML concepts, techniques, and tasks, with a focus on their adaptation to resource-constrained environments.

2.2 Machine Learning Fundamentals

In computing, where precision is paramount, algorithms are the foundation for performing tasks and making decisions. Traditionally, these algorithms are designed by human programmers who instruct a computer how to respond to various inputs.

ML, instead of depending only on human-coded instructions, allows models to evolve and adapt by learning to create their algorithms. This paradigm change offers a dynamic advantage, the ability to respond intelligently to unanticipated scenarios without manual intervention [26]. ML introduces an element of adaptability, an algorithm not only executes predefined tasks, but also learns from new data. When presented with unforeseen input, it does not break, instead it learns. It updates itself, fine-tuning its behavior to produce the expected results.

In summary, ML is a new area where algorithms can learn and evolve, adapting to the ever-changing data. This evolution promises to redefine how we approach complex problem solving, offering a brief look into the future where intelligent algorithms continuously improve and adapt to the world around them.

2.2.1 Categories of Machine Learning

ML techniques include a variety of computational approaches that allow computers to extract patterns from data and develop the capability to make decisions or execute tasks without explicit programming. These approaches use algorithms and mathematical models to process information and generate insights. A fundamental principle of ML is its ability to classify, predict, and construct models based on available data.

ML is applied in numerous fields, including handwriting and speech recognition, robotics, video games, computational linguistics, and brain-computer interfaces. Over time, ML techniques have been successfully adopted in various technological sectors, including artificial intelligence research, data analysis, and information retrieval, among others. This advancement has significantly influenced the development of diverse applications that improve multiple aspects of everyday life.

In addition, ML techniques are frequently employed in data manipulation, model simulation, disease prediction, and air quality forecasting. Recent progress has introduced bio-inspired methods, such as Artificial Immune Systems, which mimic the human immune system to address challenges in anomaly detection, clustering, and classification. These emerging paradigms emphasize the need for adaptive and scalable ML solutions, especially in handling imbalanced datasets and dynamic environments.

By integrating traditional ML approaches with novel concepts like AIS, researchers are uncovering new possibilities in software personalization, predictive analytics, and advanced optimization [27]. Systems that learn from examples, such as artificial neural networks, adopt different learning strategies tailored to their requirements. ML is divided into three primary categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

Supervised learning

Supervised learning refers to an ML approach in which algorithms learn from data that has been labeled, including input variables paired with matching output labels. This method enables algorithms to detect patterns and generate predictions on new, unseen datasets.

Recognized as a widely used subset of ML, supervised learning entails developing a model using prelabeled datasets, where each data instance is correctly assigned to an output or target. The algorithm refines its ability to predict results by mapping input data to output values, leveraging the patterns embedded within the dataset.

Supervised learning is particularly useful for tasks such as classification and regression.

Classification: In classification tasks, algorithms are designed to assign data to predefined categories or groups. This approach is commonly applied in areas such as anomaly identification, disease diagnosis, and spam filtering, utilizing models like logistic regression, decision trees, support vector machines (SVMs) and neural networks.

Regression: This type of ML focuses on predicting numerical values, such as forecasting temperature changes or estimating energy usage. Common methods include Linear Regression, Polynomial Regression, and Support Vector Regression (SVR).

Unsupervised learning

Unsupervised learning involves techniques that allow algorithms to detect patterns, structures, or relationships in data that lack explicit labels. Unlike supervised learning, this approach does not rely on predefined outputs. It is often categorized into clustering and association rule learning [28].

Clustering: This method is widely applied in unsupervised learning, where related data points are classified according to intrinsic characteristics or commonalities. Unlike supervised learning, clustering does not require prior knowledge about group structures, which makes it valuable for various use cases including dimensionality reduction and pattern recognition.

Association: Association rule learning identifies meaningful relationships in datasets. It helps uncover connections between different data points, such as discovering that customers who purchase product A are also likely to buy product B.

Some widely used unsupervised learning methods include k-means clustering and the Apriori algorithm for learning association rules.

Unsupervised learning methods have extensive applications in different domains, including market segmentation, anomaly detection, and recommendation systems. In addition, feature selection methods are crucial in feature selection, data compression, and improving visualization. A key advantage of this learning approach is its ability to reveal hidden structures within datasets, making it a powerful tool for exploratory analysis.

Semi-Supervised Learning

Semi-supervised learning combines a small fraction of labeled data with a large volume of unlabeled data to enhance training. This method is particularly beneficial in cases where annotating data is costly, time-intensive, or impractical, while acquiring large amounts of unclassified data is comparatively easier.

By integrating annotated and unannotated data, semi-supervised learning enhances model performance beyond what is achievable through purely supervised learning, which depends solely on labeled datasets. The core idea is that models can extract useful information from unlabeled data, ultimately leading to more accurate predictions, even with a limited number of labeled examples.

In real-world applications, semi-supervised learning is widely utilized in areas such as image recognition, natural language processing, and speech recognition. In these fields, acquiring labeled data can be challenging, whereas large amounts of unlabeled data are frequently accessible. Various techniques are used to exploit these data, including self-training, where an initially trained model assigns labels to unlabeled data, and co-training, in which multiple classifiers label each other's data. Semi-supervised learning utilizes both labeled and unlabeled data to enhance model generalization and minimize the risk of overfitting to the limited labeled dataset. This technique enables the model to identify and extract meaningful patterns from large datasets effectively.

Reinforcement Learning (RL)

Reinforcement learning represents an AI paradigm that enables software agents and machines to determine optimal actions within a given environment or context to enhance performance. This approach, which uses a reward and reward feedback system, is designed primarily for decision making. Reinforcement learning aims to maximize rewards and reduce risks utilizing insights derived from interactions within the environment. It is particularly effective in training AI systems in applications such as robotics, autonomous vehicles, manufacturing systems, and supply chain logistics [29].

Reinforcement learning emphasizes an agent that interacts with its surroundings and acquires knowledge through experimentation, compared to supervised learning, which uses annotated data, and unsupervised learning, which discovers underlying structures.

Reinforcement learning involves an agent making a series of decisions to accomplish a spe-

cific goal. By taking actions, the agent engages with its surroundings and obtains feedback in the form of incentives or penalties. The agent's main objective is to formulate a strategy that identifies the most optimal choices for varied conditions, in order to achieve the highest possible long-term rewards. This strategy serves as a set of guiding principles that shape the agent's reasoning process.

Fundamental elements of reinforcement learning include:

- Agent: The decision-making entity that interacts with the environment.
- Environment: The external system that provides feedback based on the agent's actions.
- State: A representation of the current situation within the environment.
- Action: The options available to the agent for modifying the environment.
- Reward: The immediate response, positive or negative, that the agent receives after taking an action.

The agent continuously explores the environment, learns from its experiences, and refines its decision-making process over time. Reinforcement learning is applied in various domains, such as gaming, robotics, autonomous navigation, and recommendation systems, where sequential decision-making and learning from interactions are essential. Some of the most commonly used RL algorithms include Q-learning, Deep Q-Networks (DQN), and Proximal Policy Optimization (PPO).

2.2.2 Common Algorithms in Machine Learning

In this section, we will briefly introduce some of the most widely used algorithms in machine learning, which serve as the foundation for various applications, including Federated Learning.

- **Linear Regression** is a fundamental algorithm designed to estimate continuous numerical outputs by analyzing the correlation between independent features (inputs) and outcome variables (outputs). It is commonly used as a baseline method for various tasks involving regression analysis.

- **Logistic Regression** is a widely adopted algorithm in FL, especially for binary and multi-category classification problems. Owing to its simplicity along with efficiency, it is well suited for systems with constrained resources, requiring minimal computational capacity and storage. This method estimates outcome probabilities using a logistic curve, which can be easily updated and integrated between multiple clients in a federated setting.

- **Decision Trees** are versatile algorithms that divide data into branches based on feature values, producing intuitive and interpretable models for classification and regression. These trees are particularly advantageous in distributed settings because of their low computational requirements.

- **Support Vector Machines (SVMs)** Support Vector Machines are commonly used in FL, particularly for tasks that require precise classification. They function by identifying a hyperplane that effectively separates data points into distinct categories, making them suitable for datasets with clear separation margins. In FL, linear SVMs are often preferred in resource-constrained environments due to their lower computational demands compared to kernel-based SVMs.

- **Naïve Bayes** is a probability-based algorithm derived from Bayes' theorem, frequently applied in areas such as text categorization and spam identification. Its simplicity and minimal resource requirements make it an excellent choice for FL applications with limited computational capabilities.

- **Neural Networks (NNs)** are among the most adaptable algorithms in FL, capable of processing intricate and high-dimensional data. Their versatility makes them suitable for a wide array of applications, from visual recognition to language processing. In FL, Neural

Networks employ methods such as Federated Averaging (FedAvg) to consolidate model weights across clients, supporting decentralized training. However, their significant computational needs make them more suitable for devices with adequate resources. Despite these challenges, Neural Networks are indispensable in FL because of their ability to solve complex problems and deliver precise results.

- **Random Forest** is an ensemble-based approach that creates multiple decision trees and integrates their output to improve the reliability of the prediction. It excels in managing diverse data types and improving model accuracy in FL environments.

- **Gradient Boosting Algorithms** Techniques such as XGBoost and LightGBM fall under the category of Gradient Boosting Algorithms. These models enhance training by iteratively refining errors from previous iterations. They are especially effective in working with structured data and in achieving high accuracy in FL tasks.

- **K-Means Clustering** is one of the simplest and most widely used techniques. Divide data into a predefined number of clusters based on similarity, minimizing variance within each cluster. This method is widely applied in customer segmentation and pattern recognition. Another popular clustering method is Hierarchical Clustering.

- **Hierarchical Clustering** organizes data into a hierarchy of nested clusters. This approach is valuable in fields like bioinformatics and document clustering, where hierarchical relationships among data points are essential.

- **Density-Based Spatial Clustering of Applications (DBSCAN)**, on the other hand, focuses on grouping data points based on density, effectively identifying noise and outliers. This makes it particularly useful for spatial data analysis and anomaly detection.

- **Mean-Shift Clustering and Spectral Clustering** are other advanced clustering techniques used to address specific challenges in image processing and social network analysis, respectively.In addition to clustering, unsupervised learning includes algorithms for dimensionality reduction, which aim to simplify complex datasets while retaining critical information.

- **Principal Component Analysis (PCA)** is a widely used method that projects data onto lower-dimensional spaces, making it suitable for feature extraction and data visualization.

- **Linear Discriminant Analysis (LDA)**, is another method primarily used to maximize class separability in supervised learning, but can also assist in dimensionality reduction when labels are limited.

- **Non-Negative Matrix Factorization (NMF), Locally Linear Embedding (LLE), and Isomap** offer alternative approaches to reduce dimensions, each with unique advantages for tasks such as image analysis, topic modeling, and manifold learning. Association rule learning is another significant aspect of unsupervised learning, particularly in the identification of relationships between variables in large datasets.

- **Apriori, FP-Growth, and Eclat** are techniques developed to detect common sets of items and establish association rules. These methods are frequently used to analyze shopping baskets to identify trends in product purchases, which can improve recommendation engines. Modern tree-based approaches further boost the efficiency and scalability of mining association rules, making them well suited for managing large datasets of transactions.

2.3 Deep Learning

Deep Learning is an area within ML that employs multi-layered neural networks to capture intricate patterns in data. These networks, modeled after the architecture of the brain, consist of layers of connected nodes that handle information in a structured way. Deep learning models are particularly powerful in automatically deriving insights from raw data, making them highly efficient for tasks involving complex high-dimensional data such as images, sound, and text. A major strength of deep learning lies in its ability to autonomously learn feature representations from unprocessed information, eliminating the need for manual feature extraction.

Deep learning has significantly advanced domains such as computer vision, speech understanding, and natural language processing. For instance, architectures such as CNNs are extensively employed for tasks like visual data classification and object recognition, whereas RNNs and their variations, such as LSTM networks, are proficient in sequential data processing tasks like language modeling and translation. Breakthroughs like Transformer architectures (e.g., GPT and BERT) have further elevated performance in language-related tasks. These improvements are supported by the proliferation of extensive datasets, advancements in GPU technology, and refined optimization techniques, enabling state-of-the-art outcomes across

diverse domains.

Deep learning, a specialized subset of ML, employs multi-layered artificial neural networks to uncover hierarchical data patterns. Its prominence stems from its proficiency in processing unstructured, high-dimensional data (e.g., images, audio, text). Unlike traditional ML, which often depends on hand-crafted features, deep learning automates representation learning directly from raw input. This ability arises from the structured design of neural networks, where successive layers incrementally refine feature abstraction levels [30].

Central to deep learning is the backpropagation algorithm, which iteratively adjusts network parameters to minimize errors. Enhancements like gradient-based optimization, dropout regularization, and strategic parameter initialization have further enhanced its efficacy. For example, CNNs mimic biological visual processing to revolutionize image analysis, while RNNs and LSTMs dominate sequential tasks such as speech recognition.

Scalability is another hallmark of deep learning. Using computational advancements and vast datasets, these models achieve unparalleled performance. In healthcare, deep learning outperforms conventional methods in diagnosing conditions such as cancer through medical imaging. Similarly, transformer-based models have redefined benchmarks in NLP tasks such as sentiment analysis and automated question answering.

Beyond the foundational aspects, recent advances have introduced specialized architectures for specific tasks. For example, Generative Adversarial Networks (GANs) have become powerful tools for generating realistic data, such as images and music, by training two neural networks in opposition. This innovative approach has opened up new possibilities in creative AI applications and data augmentation techniques [31].

The merging of deep learning and reinforcement learning has led to the creation of deep reinforcement learning (DRL) algorithms. These frameworks empower agents to acquire optimal behaviors through environmental interactions, achieving notable success in challenging domains such as gaming and robotic control. The partnership between deep learning and reinforcement learning continues to propel advances in the building of intelligent systems capable of independent decision-making [32].

These models excel not only in generalizing to new, unseen data but also in thriving within intricate real-world environments. This versatility solidifies deep learning as a foundational pillar of modern AI frameworks.

2.3.1 Convolutional Neural Network (CNN)

A CNN is a deep learning architecture optimized for analyzing data arranged in grids, such as images or videos. These networks excel at tasks requiring visual understanding, leveraging layered operations like convolution and pooling to detect spatial hierarchies of features. The following are its core elements:

Convolutional Layers: These layers perform mathematical convolutions on input data using adjustable filters (kernels). Each filter isolates specific visual elements, such as edges, textures, or complex shapes, by systematically scanning the input and generating activation maps through dot product calculations.

Activation Functions: Non-linear transformations, such as the ReLU (Rectified Linear Unit) function, are applied post-convolution to enable the network to model intricate relationships within the data.

Pooling (Downsampling) Layers: These layers compress feature maps spatially while retaining critical information. Max-pooling (selecting maximum values) and average-pooling (calculating averages) are standard techniques.

Fully Connected Layers: Positioned after the convolutional and cluster layers, these layers integrate global characteristics for classification. They establish dense connections between all neurons in successive layers.

Output Layer: The final layer produces task-specific results, using activation functions such as softmax (for multiclass categorization) or sigmoid (for binary decisions).

CNNs autonomously learn hierarchical representations from raw input, making them indispensable for applications such as image categorization, object localization, and pixel-level segmentation.

2.3.2 Neural Networks and CNNs in Federated Learning

Neural networks (NNs) are computational models inspired by the human brain, consisting of interconnected layers of neurons that process and learn from data. These networks excel at recognizing patterns and making predictions, making them crucial for applications such as image processing, speech recognition, and language understanding.

Convolutional Neural Networks (CNNs), a distinct class of neural networks, are particularly effective in interpreting visual data due to their ability to capture spatial patterns through convolutional layers.

In FL, CNNs play a crucial role in collaborative privacy-focused training on distributed datasets, including images and videos. By training models directly on edge devices and transmitting only updates instead of raw data, CNNs contribute to maintaining data privacy.

CNNs find extensive use in areas like medical imaging, where patient-related data cannot be centralized, as well as in smart city infrastructure, where edge devices process video streams. However, despite these advantages, the use of CNNs in FL poses certain challenges, such as high computational demands on edge devices, handling data that are not IID (independently and identically distributed), and managing communication overhead during model aggregation.

2.4 Centralized Learning vs. Federated Learning

Traditional machine learning approaches, often referred to as centralized learning, rely on gathering all data into a single location, such as a cloud server or data center, where models are trained. This method simplifies the training process, as all data is readily accessible and powerful computational resources can be utilized to optimize models efficiently. However, this centralized approach comes with significant drawbacks, particularly in terms of data privacy, security, and communication costs. For example, transferring large volumes of sensitive data to a central server can expose them to potential breaches, while also consuming substantial network bandwidth.

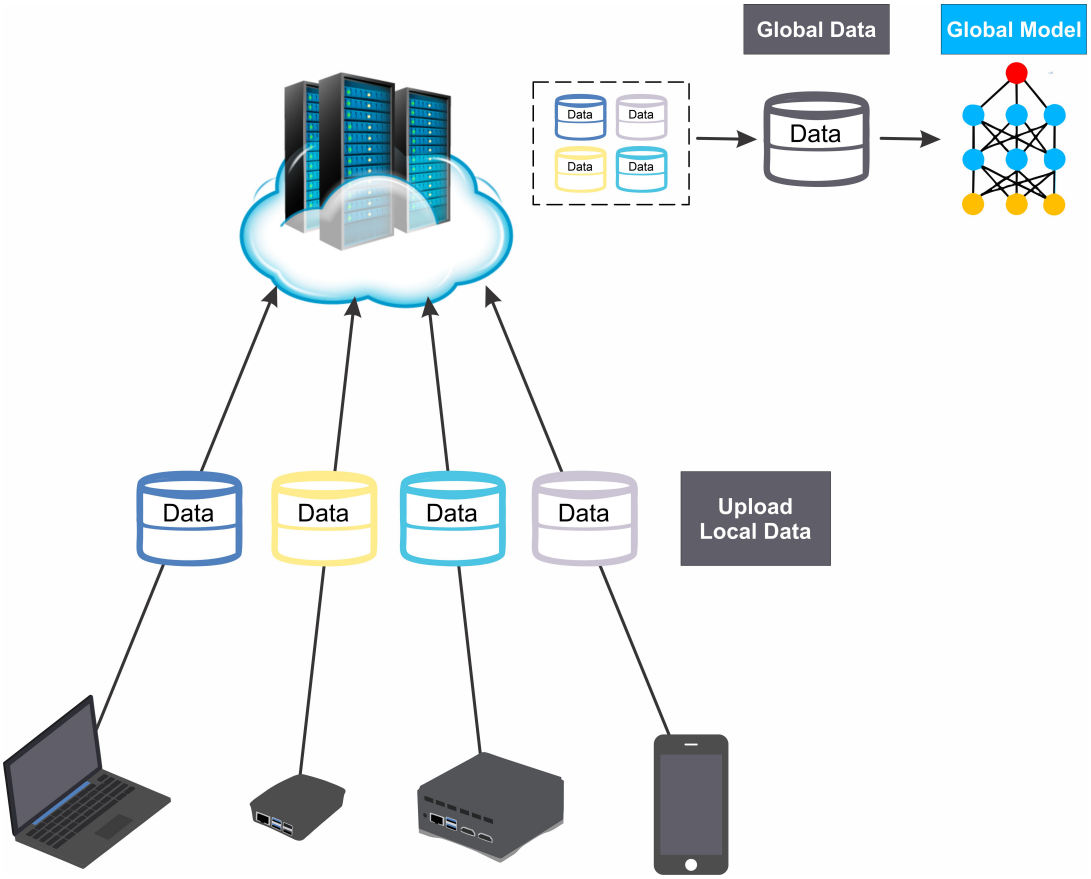

Figure 2.1: Centralized Learning

In contrast, FL offers a decentralized alternative that addresses these challenges. Instead of moving data to a central server, FL enables devices to train models locally using their own data. Only model updates are shared with a central server, which aggregates these updates to improve the global model. This approach ensures that raw data remain on the local devices, significantly enhancing data privacy and reducing the need for large-scale data transfers. As a result, FL is particularly well suited for applications involving sensitive information, such as healthcare, where patient data must remain confidential.

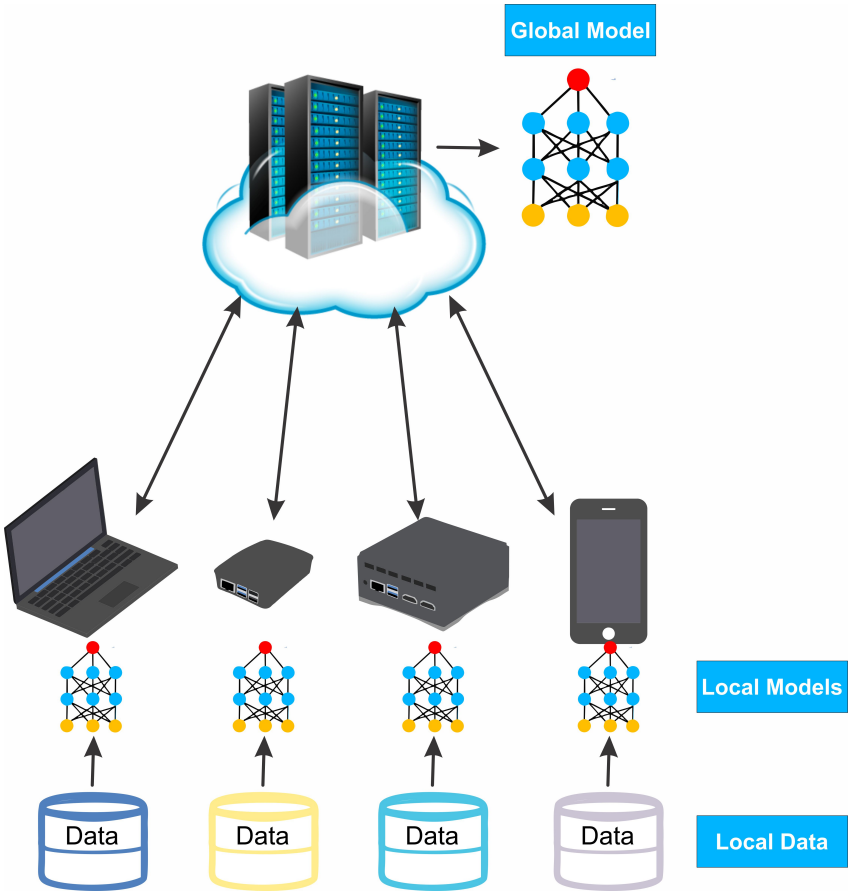

Figure 2.2: Distributed Learning

While centralized learning excels in environments where data can be easily consolidated and computational resources are abundant, FL shines in scenarios where data privacy, bandwidth efficiency, and scalability are critical. For example, in IoT ecosystems with thousands of edge devices, FL allows for collaborative model training without overloading the network or compromising user privacy. However, FL also introduces new challenges, such as managing device heterogeneity, handling non-IID (non-independent and identically distributed) data, and ensuring efficient resource utilization across devices with varying capabilities.

Advantages of Federated Learning Over Centralized Approaches One of the key advantages of FL is its ability to preserve data privacy. Since raw data never leaves the local device, the risk of data breaches or unauthorized access is minimized. This is especially important in industries like healthcare and finance, where sensitive information must be protected. Additionally, FL reduces communication overhead by transmitting only model updates instead of entire datasets, making it more efficient in bandwidth-constrained environments.

Another benefit of FL is its scalability. By leveraging the computational power of edge devices, FL can distribute the training workload across a large number of participants, enabling

the system to scale more effectively than centralized approaches. This is particularly useful in applications like smart cities or industrial IoT, where thousands of devices may contribute to the training process.

However, FL is not without its challenges. Device heterogeneity—where participants have varying computational power, storage capacity, and network connectivity—can lead to imbalances in the training process. Additionally, non-IID data distributions across devices can complicate model convergence, as the data on each device may not be representative of the overall dataset. To address these issues, adaptive mechanisms, such as those proposed in this study, are essential for optimizing resource allocation and ensuring stable training.

2.5 Federated Learning

In a conventional ML setup, model training is typically performed on a single machine using locally available data. However, FL introduces a decentralized approach by allowing a single ML model to be trained collaboratively on multiple distributed devices. A key principle in FL is that data remain in local devices, improving privacy and security. Rather than centralizing the data, FL shifts the focus to decentralizing the model itself. In this setup, the ML model is sent to each participating device, where training occurs using only locally collected data. This decentralized approach offers significant advantages, particularly in terms of safeguarding sensitive information. It allows clients to collaborate and harness the collective power of their devices without exposing raw data to external entities. Multiple FL algorithms have emerged, each tailored to specific requirements and trade-offs. Although all of them prioritize data privacy, they differ in their performance metrics. For example, some FL algorithms enable concurrent training on multiple devices, optimizing efficiency, while others follow a more sequential approach. One major benefit of FL is its ability to minimize data transmission requirements. Instead of transmitting large volumes of raw data to a central server, FL operates by distributing the model itself to edge devices. The amount of data transmitted depends on how often the model is updated and transferred between devices. By extending training durations and reducing the frequency of model transfers, organizations can effectively reduce overall transmission overhead, a crucial factor for environments with limited resources. FL is transforming the field of ML by introducing collaborative model training across multiple devices while prioritizing data privacy. This approach marks a fundamental shift from centralized data processing to decentralized model learning.

2.5.1 Federated Learning Techniques

Federated Learning (FL) Techniques encompass a variety of methods designed to address the challenges of training machine learning models over distributed devices while preserving data privacy. These techniques focus on optimizing communication, handling data heterogeneity, and ensuring robust model performance in resource-constrained environments. From foundational algorithms like FedAvg, which aggregates local updates to create a global model, to more advanced approaches like FedProx, which tackles device variability, these methods form the backbone of FL. By tailoring techniques to specific use cases and challenges, FL enables scalable and efficient training across diverse and decentralized systems.

FedAvg (Federated Averaging)

A cornerstone FL algorithm, FedAvg combines local model updates from distributed devices using weighted aggregation to create a global model. Its simplicity and effectiveness make it widely applicable in FL scenarios.

FedProx (Federated Proximal)

FedProx evolves FedAvg by introducing a regularization term to minimize discrepancies between local and global models, addressing variability in device capabilities or data distributions.

q-FedAvg

This variant prioritizes contributions from devices with higher quality or larger datasets, ensuring that the global model benefits more from reliable or substantial local updates.

FedOpt

A family of algorithms that use advanced optimizers such as SGD or Adam to improve global updates.

Personalized FL

Rather than creating a unified model, this technique focuses on developing local models tailored to specific devices, balancing global improvements with local needs.

Secure Aggregation

A method that ensures the privacy of updates using homomorphic encryption, guaranteeing that even the central server cannot access individual updates.

FedADA (Federated Adaptive Training)

FedADA is an adaptive FL framework designed to address system heterogeneity by dynamically assigning workloads to clients based on their computational capabilities and data characteristics. Unlike traditional methods like FedAvg, which use a fixed number of local training iterations, FedADA minimizes runtime gaps between clients and maximizes convergence speed. This approach is particularly effective in heterogeneous Mobile Edge Computing (MEC) environments, where clients have varying resources and network conditions. By optimizing the distribution of the workload, FedADA significantly reduces the convergence time and improves the efficiency of training.

FedALA (Federated Learning with Adaptive Local Aggregation)

FedALA tackles statistical heterogeneity by personalizing the global model for each client. Its core component, the Adaptive Local Aggregation (ALA) module, intelligently combines the

global model with local models to better align with the data distribution of each client prior to training. This enhances the generalization and accuracy of the global model for various clients. FedALA has demonstrated superior performance in both computer vision and natural language processing tasks, outperforming state-of-the-art baselines and offering a versatile solution for personalized FL.

2.5.2 Adaptive Mechanisms in Federated Learning

The Need for Adaptivity in FL

Traditional FL operates under the assumption that all participating devices contribute equally to the global model training. However, in real-world deployments, heterogeneous devices vary significantly in terms of computational power, network stability, storage capacity, and energy constraints. This disparity creates several challenges that can degrade model performance and training efficiency.

- **Straggler Effect:** Some devices, particularly those with limited processing power, take significantly longer to complete local training rounds, delaying the entire training process.
- **Resource Exhaustion:** Low-power devices may experience excessive CPU and memory usage, leading to thermal throttling, system slowdowns, or unexpected crashes.
- **Unstable Network Conditions:** FL relies on model updates from multiple devices, but intermittent or low-bandwidth connections may cause synchronization failures, dropped updates, or increased communication overhead.
- **Non-IID Data Distribution:** Unlike traditional centralized learning, FL involves non-identically distributed (non-IID) data, where different clients may have biased or imbalanced datasets. This leads to slower convergence and fairness issues in global model aggregation.

To address these limitations, adaptive mechanisms are introduced in FL frameworks to dynamically adjust training parameters, ensuring efficient resource utilization and maintaining system performance across diverse environments.

Types of Adaptation in Federated Learning

Adaptivity in FL can be implemented at various levels, each targeting a specific challenge related to device heterogeneity, network conditions, or computational constraints. The following are the primary types of adaptation mechanisms used in FL:

Adaptive Learning Rate

The choice of learning rate significantly impacts the convergence of the model and the stability of the training. In traditional FL, a fixed learning rate is used across all clients, which can lead to:

- Slow convergence in resource-limited devices due to insufficient parameter updates.
- Instability in high-performance devices, where the model may oscillate without stabilizing.

Solution: Adaptive learning rate strategies adjust the step size dynamically based on the training progress, allowing devices to converge faster while reducing computational overhead. Examples include:

- FedAdam and FedYogi, which extend adaptive optimization techniques (Adam, Yogi) to FL.
- Momentum-based adaptation, which adjusts the learning rate based on historical gradient trends.
- AdaptiveMesh (Proposed in this thesis) - Adjusts participation criteria based on real-time resource metrics (CPU, bandwidth, memory).

Dynamic Client Selection

Since not all devices have equal training capabilities, selecting the right clients per round is crucial to improve FL efficiency. Naive FL selects clients randomly, which can lead to suboptimal training performance if weaker devices dominate participation.

Solution: Dynamic client selection techniques include the following:

- FedCS (Client Selection Strategy) – Prioritizes clients with stable network connections and sufficient computational resources.
- FedIQ - Uses reinforcement learning to intelligently and dynamically choose clients based on their historical performance.

- AdaptiveMesh (Proposed in this thesis) – Adjusts participation criteria based on real-time resource metrics (CPU, bandwidth, memory).

Resource-Aware Workload Distribution

FL traditionally assigns the same batch size and number of epochs to all devices, assuming a uniform computational capacity. This approach is inefficient for heterogeneous environments where:

- High-end devices are underutilizing their resources.
- Low-end devices struggle to complete training in reasonable timeframes.

Solution: Adaptive workload distribution strategies tailor batch sizes, epoch counts, and training image distribution based on device-specific constraints:

- FedProx: Addresses diverse workloads by adding a proximal term to the FL optimization algorithm.
- FedNova: Normalizes the contributions of different clients, ensuring balanced updates.
- AdaptiveMesh: Dynamically adjusts training epochs and image sample sizes based on CPU load and bandwidth availability.

Bandwidth-Aware Model Aggregation

FL requires frequent communication between clients and the central server. In environments with limited or fluctuating network bandwidth, naive model updates can lead to:

- Increased communication overhead.
- Updates were dropped due to unstable connections.
- Significant energy consumption in battery-powered devices.

Solution: Several techniques have been developed to optimize bandwidth usage:

- Compression-based updates (e.g., FedDrop, FedPAQ): Reduce the size of transmitted model updates.
- Asynchronous FL (e.g., AsyncFL, FedAsync): Allows clients to contribute at different rates, reducing waiting times for slower devices.
- BACA (Bandwidth and CPU-Aware FL): Adapts training frequency according to network conditions, ensuring a fair contribution without overloading weaker clients.

Adaptive Federated Learning Algorithms

Federated Learning has seen significant advancements with the development of adaptive algorithms designed to address challenges such as device heterogeneity, non-IID data, and dynamic network conditions. While there are numerous algorithms in the literature, we will focus on a few key adaptive approaches.

FedProx (Federated Proximal Algorithm): This algorithm incorporates a proximity constraint within the loss function to manage local updates, minimizing drastic model deviations caused by varying computational capacities among devices. By regulating the extent to which local updates diverge from the global model, FedProx ensures that weaker clients do not significantly hinder global convergence. It is especially useful in FL scenarios where client participation fluctuates due to resource constraints.

FedNova (Normalized Averaging): Designed to improve fairness in FL, this approach normalizes client contributions during model aggregation. Unlike traditional FL methods, which assume equal participation among all clients, FedNova addresses real-world discrepancies where some devices perform more local updates than others due to computational disparities. Balances this by adjusting updates according to the frequency of local iterations, ensuring that no single device exerts an excessive influence on the global model.

FedOpt (Federated Optimization with Adaptive Learning Rates): This algorithm enhances optimization strategies by extending techniques such as Adam and Yogi to FL environments. Unlike standard FL approaches that maintain a fixed learning rate, FedOpt dynamically modifies learning rates based on model performance and observed gradient trends. In doing so, it improves the efficiency of training in various hardware configurations. This approach is very useful for training deep learning models in distributed settings with restricted computational capacity.

FedADA (Federated Adaptive Training): FedADA addresses the heterogeneity of the system by adaptively assigning workloads to clients based on their computational capabilities and data characteristics. It minimizes run-time gaps between clients and maximizes convergence speed, making it particularly effective in heterogeneous MEC environments. FedADA significantly reduces the convergence time compared to traditional methods like FedAvg.

FedALA (Federated Learning with Adaptive Local Aggregation): FedALA tackles statistical heterogeneity by personalizing the global model for each client. Its Adaptive Local Aggregation (ALA) module combines global and local models to better align with each client's data distribu-

tion, improving generalization and accuracy. FedALA has demonstrated superior performance in both computer vision and natural language processing tasks.

Some algorithms mentioned in the literature, such as FedAvg and FedSGD, are not inherently adaptive. They rely on fixed training parameters and do not dynamically adjust to device or network conditions. While these algorithms are foundational to FL, they often struggle in heterogeneous environments where device capabilities and network conditions vary significantly.

2.5.3 Advantages and Challenges of Federated Learning

Advantages

- **Data Privacy:** By ensuring that raw data never leave the local device, FL significantly improves data privacy and meets stringent regulatory requirements (e.g. GDPR, HIPAA).
- **Bandwidth Efficiency:** FL reduces the need to transmit large data sets by focusing instead on sharing model updates, saving bandwidth and reducing communication overhead.
- **Scalability:** FL is inherently scalable, allowing training across a wide network of devices without centralizing the data.
- **Inclusivity:** FL captures diverse data distributions by training on heterogeneous datasets from multiple devices, improving the generalization of the model.
- **Utilization of Local Resources:** FL leverages the computational power of edge devices, reducing the dependency on central servers.

Challenges

- **Heterogeneous Data:** Data across devices often exhibit nonuniform statistical distributions (non-IID), complicating the training of a cohesive global model.
- **Device Variability:** Participants in FL systems may have widely different computational capacities, memory constraints, or energy limitations, resulting in uneven training outcomes.
- **Communication Overhead:** Frequent coordination of model updates between devices

can strain the network bandwidth, particularly in environments with limited connectivity.

- **Security Risks:** While FL inherently protects data privacy, the decentralized framework is vulnerable to malicious activities such as adversarial data manipulation or attacks compromising privacy.
- **Algorithm Convergence:** Ensuring convergence in distributed systems with asynchronous updates and data heterogeneity remains a significant technical challenge.

2.5.4 Privacy and Security in Federated Learning

Privacy and security are critical concerns in FL, as the decentralized nature of the paradigm introduces unique vulnerabilities and risks.

One of the primary advantages of FL is its ability to preserve data privacy by keeping raw data on local devices. However, the model updates exchanged between devices and the central server can still leak sensitive information. For example, an adversary could infer private data by analyzing the gradients or weights of the model updates. To mitigate this risk, differential privacy has been widely adopted in FL. Differential privacy adds noise to model updates, ensuring that individual data points cannot be reconstructed while still allowing the model to learn from the aggregated data.

Secure multi-party computation (SMPC) is another technique used to enhance privacy in FL. SMPC allows multiple parties to jointly compute a function over their input while keeping those inputs private. In the context of FL, SMPC can be used to securely aggregate model updates without revealing the individual contributions of each device. This approach is particularly useful in scenarios where multiple organizations or institutions collaborate on a shared model.

Homomorphic encryption is a cryptographic technique that enables computations to be performed on encrypted data without decrypting it. In FL, homomorphic encryption can be used to protect model updates during transmission and aggregation. Although this method provides strong privacy guarantees, it is computationally expensive and may not be suitable for resource-constrained devices.

Another important consideration in FL is the prevention of adversarial attacks. Adversaries may attempt to manipulate the training process by injecting malicious data or updates to the models, leading to biased or inaccurate models. Robust aggregation techniques, such as Byzantine-resilient algorithms, have been developed to detect and mitigate the impact of adversarial contributions. These algorithms ensure that the global model remains accurate even in the presence of malicious actors.

In addition to these technical solutions, FL frameworks must also address legal and regulatory requirements related to data privacy. For example, the General Data Protection Regulation (GDPR) in the European Union imposes strict rules on the collection, storage, and processing of personal data. FL systems must be designed to comply with such regulations, ensuring that data privacy is maintained throughout the training process.

Despite the progress made in improving privacy and security in FL, several challenges remain. For example, the trade-off between privacy and model accuracy is a persistent issue. Adding noise or encryption to protect data can reduce the quality of the model, leading to lower accuracy. Researchers are actively exploring ways to balance these competing objectives, such as developing more efficient encryption methods or adaptive privacy preservation techniques.

Another challenge is the scalability of privacy and security solutions in large-scale FL systems. As the number of participating devices increases, the computational and communication overhead of privacy-preserving techniques can become prohibitive. To address this, researchers are investigating lightweight encryption methods and distributed privacy-preserving algorithms that can scale to thousands or even millions of devices.

Using techniques such as differential privacy, secure multi-party computation, and homomorphic encryption, FL systems can protect sensitive data while facilitating collaborative model training. However, ongoing research is needed to overcome challenges related to accuracy, scalability, and regulatory compliance, ensuring that FL remains a viable and secure solution for decentralized machine learning.

Additional Aspects of Security in FL:

- Model Inversion Attacks: These attacks involve reconstructing training data from the trained model. Even though data is kept on local devices, an attacker can use the global model to infer sensitive information.

- Membership Inference Attacks: In these attacks, an adversary attempts to determine whether a specific data point was used in the training of the model. This can reveal whether a specific individual contributed to the training data.

- Data Poisoning Attacks: In these attacks, an adversary introduces false or harmful data into the training process to negatively influence the performance of the global model. This can occur in FL, where individual devices may be compromised.

- Model Extraction Attacks: An attacker may attempt to extract the entire model or parts of it by analyzing the model's outputs. This is particularly relevant in FL, where the global model is shared among many devices.

- Communication Security: The security of communication channels between devices and

the central server is crucial. Man-in-the-Middle (MITM) attacks can be used to intercept or manipulate model updates during transmission.

- Device Compromise: If a local device is compromised, an attacker can gain access to local data and manipulate the model updates sent to the central server. This can lead to an inaccurate or harmful global model.

- Fairness and Bias: Security is not only about protecting against attacks but also ensuring that the global model is fair and unbiased. If some devices contribute biased data, the global model may reflect these biases, which can have significant implications in real-world applications.

- Auditability and Transparency: An important aspect of security is the ability to audit and verify the training process. This is particularly important in FL, where many different parties contribute to the global model.

- Resource Exhaustion Attacks: An attacker may attempt to exhaust the resources of local devices (e.g., battery, CPU) by sending numerous or heavy requests, making these devices unable to contribute effectively to the training process.

- Regulatory Compliance: In addition to GDPR, there are other data privacy laws and regulations in different regions of the world (e.g., CCPA in California, PIPEDA in Canada) that FL systems must comply with.

2.5.5 Federated Learning Applications

FL is an advanced technology that provides innovative solutions to several challenges faced by traditional ML models. In this article, we will explore its practical applications and use cases, shedding light on how it is revolutionizing various industries.

- Healthcare: One of the most promising applications of FL is in the healthcare sector. Privacy and data security are paramount when dealing with sensitive patient information. FL allows hospitals and research institutions to collaborate without sharing patient data. In this approach, local hospitals maintain their patient data and the ML model is trained across these local datasets. The global model is updated without centralizing patient information. This has significant implications for predicting disease outbreaks, optimizing treatment plans, and drug discovery. Researchers and healthcare providers can collaborate more effectively, ultimately improving patient care while preserving privacy.

- Smartphones and Personalization: Your smartphone knows a lot about you, your daily routine, your interests, and even your typing habits. FL enables personalization without exposing your data to the cloud. For example, predictive text on your phone can improve without sending everything you type to a central server. Instead, your device uses FL to train models locally, so you get more accurate suggestions and auto-corrections while preserving your privacy. The same concept can be applied to other personalization features, such as app recommendations and voice assistants.

- Financial Services: The financial sector deals with large amounts of data, and data protection and cybersecurity are critically important. FL has significant potential to revolutionize fraud detection and risk assessment. Banks and financial institutions can collaborate on training ML models to detect fraudulent transactions while keeping individual transaction details private. Additionally, FL can be used to analyze market trends without exposing sensitive trading data. This helps achieve effective risk management and more informed investment choices.

- Manufacturing and IoT: It is becoming increasingly prevalent in manufacturing. Machines, sensors, and smart systems produce enormous volumes of data, which can be harnessed to improve efficiency and predictive maintenance. FL allows these devices to share information without sharing raw data. Manufacturing plants can collectively train

models to anticipate machinery breakdowns, improve workflow efficiency, and minimize operational halts. This technology helps save money and improves overall productivity in the manufacturing industry.

- Traffic Management Traffic management: It is a critical challenge in urban planning and transportation. FL can help optimize traffic signal timings, route planning, and congestion management. In this scenario, data from various sources, such as traffic cameras, GPS data, and public transportation schedules, can be used to collectively improve traffic flow and reduce gridlocks. However, data sources do not need to reveal specific vehicle or personal information, ensuring privacy while improving city-wide transportation systems.

Federated Learning in Healthcare

FL is transforming healthcare by allowing institutions to work together securely while protecting patient privacy. It enables AI models to be trained across decentralized data sources, keeping confidential patient information stored locally and secure. By addressing critical privacy and governance challenges, FL facilitates the development of robust AI solutions [33] [34]. Several studies and research efforts have explored the application of FL in healthcare. Here is a brief overview of some related work with corresponding references:

- Enhancing Medical Image Analysis FL has been successfully applied to medical image analysis, enabling hospitals to collaboratively improve diagnostic models without sharing raw data. This approach has been particularly impactful in fields such as brain tumor magnetic resonance analysis, where FL allows the integration of various datasets to create more generalizable models [35]. Techniques like Abstention-Aware Federated Voting (AAFV) have further enhanced FL's capabilities, ensuring that only high-confidence predictions are shared while maintaining data confidentiality [36].

- Facilitating Distributed Clinical Trials

  The decentralized nature of FL has found applications in distributed clinical trials, where multiple institutions collaborate on data-driven studies without transferring sensitive data across borders. This innovation is crucial to improving drug efficacy evaluations and addressing safety concerns, all while maintaining compliance with local data regulations [34], [35].

- Developing Predictive Models for Healthcare

  FL supports the creation of predictive models that analyze patient outcomes and forecast health risks. For example, FL-trained models have shown success in predicting hospital readmissions and managing chronic diseases such as diabetes and cardiovascular diseases [36].

- Addressing Privacy and Security Challenges While FL significantly mitigates risks associated with centralized data sharing, it introduces new privacy concerns related to model updates. Potential risks, such as membership inference attacks, can reveal sensitive details about individual datasets. Advanced privacy-preserving techniques such as differential privacy, secure aggregation, and encryption protocols are used to counteract such vulnerabilities [36].

- Overcoming Data and Model Heterogeneity The inherent heterogeneity of healthcare data poses challenges in training unified models. FL tackles this by allowing institutions to maintain locally trained models while contributing to a global model through parameter aggregation. This ensures that the global model benefits from diverse data sources while respecting individual institutional constraints [33] [35].

- Future Directions for FL in Healthcare

  FL has the potential to shape the future of digital healthcare by enabling large-scale collaborations that respect privacy and governance. Its applications extend beyond medical imaging and clinical trials, offering opportunities for disease surveillance, integration of wearable health data, and pandemic response strategies. To fully exploit FL's potential, future efforts must focus on improving communication efficiency, reducing computational overhead, and developing standardized frameworks for healthcare applications.

Federated Learning in Environmental Monitoring

Environmental monitoring plays a fundamental role in understanding and preserving natural resources. It helps track pollution levels, monitor biodiversity, and evaluate the impact of human activities on ecosystems. Maintaining accurate and regularly updated data is essential for making well-informed decisions, implementing effective environmental protection strategies, and ensuring the well-being of both humans and wildlife.

In conventional centralized ML, data is typically collected, processed, and stored in a single location. However, in environmental monitoring, data sources are often distributed across multiple organizations and locations. FL provides a solution by enabling collaboration and knowledge exchange while maintaining data privacy and minimizing communication costs. This approach has been successfully adapted for various environmental monitoring applications. For example, it has been used to predict air quality trends based on distributed sensor data, track deforestation rates using satellite imagery, and analyze wildlife migration patterns using data from different conservation organizations.

There are specific studies on the application of FL in Environmental Monitoring, such as the work [37], which explores its potential and examines the ongoing challenges related to its implementation in smart city environments. The first paragraph outlines the primary benefits of FL in this context, including improved data security, privacy protection, reduced communication overhead, and improved management of distributed data and system complexity. The following section highlights the critical challenges and possible solutions for addressing issues such as security, energy efficiency, and the adaptability of FL algorithms to the dynamic and intricate nature of smart cities. This material will be used as a reference source in my work to argue for the benefits and challenges of employing FL in the context of smart city development.

In [38], the authors introduce a proposed framework for an integrated system designed to evaluate environmental and health-related impacts. This system is intended to address the consequences of air pollution on human well-being. Using FL, the framework aims to assess the possible health risks associated with exposure to air pollutants. It underscores the importance of analyzing the results of these evaluations to enhance the management and oversight of healthcare services. In addition, it highlights the role of AI alongside FL, integrating advanced ML techniques to monitor environmental variables that influence health conditions. The framework consists of multiple stages to ensure a thorough assessment of pollution and its health effects. The study presents FL as a key methodology for constructing predictive models within this system. It emphasizes its adaptability and ability to provide real-time forecasts, showcasing its relevance in monitoring air quality. In summary, the authors focus on the development of an integrated evaluation system for environmental and health impacts, underlining the role of AI and FL in mitigating the adverse effects of air pollution on public health. They also highlight its potential to improve health services, address research gaps, and improve environmental policies.

This paper [39] examines FL as a decentralized ML framework that allows multiple devices to jointly contribute to a global model while ensuring that data remain on local devices. It classifies FL into three distinct types: horizontal FL, vertical FL, and federated transfer learning, explaining their distinctions. Furthermore, the study explores the application of FL in multiple fields, such as IoT, wireless technology, healthcare, and cybersecurity. The chapter also discusses the main challenges in FL, such as communication overhead, device compatibility, and variability in data sources. Furthermore, the authors explore how FL contributes to predicting air quality index (AQI) and propose that a novel approach, referred to as multi-model FL, could be leveraged to enhance AQI forecasting techniques. They suggested that this emerging method could significantly improve air quality prediction and environmental assessment efforts. In addition, the authors discuss the role of FL in AQI prediction and suggest that a novel approach called Multi-Model FL has the potential to enhance AQI prediction moving forward.

The authors explain that multi-model FL is an emerging domain and propose that the DNN architecture should be used in client models. The paper proposes a potential development direction to improve AQI prediction using Multi-Model FL.

Federated Learning Use-Cases Considered in the Thesis

FL is predominantly used in situations that require a significant emphasis on protecting privacy and optimizing resource allocation. The healthcare and medical sector is a major field in which FL finds extensive applications. This section outlines the medical datasets used in the study. The dataset in the thesis is selected from the healthcare applications mentioned below.

Healthcare Industry: FL has great potential in the healthcare industry, but there are ways to make it even better:

- Standardized Data Formats: Healthcare data can be messy and come in different formats. Standardizing how data are structured can make it easier for different hospitals and clinics to share and use their data together.
- Better Communication: Making sure that all machines and devices in healthcare can easily talk to each other is important. In this way, data can be shared and combined more effectively for better patient care.
- Data Quality Checks: The quality of the data is crucial. This means making sure it is accurate and up to date so the ML models can learn from the best information.
- Stronger Privacy Protections: Patient privacy is a major concern. Strengthening safeguards to protect personal information during FL is essential to keep healthcare data safe.
- More Collaboration: Encouraging hospitals, doctors, and researchers to work together on using FL can lead to better healthcare information. When more people share their knowledge, the results can be more accurate and useful.

Numerous hospitals, AI-driven organizations, and regulatory bodies play a crucial role in safeguarding users' confidential medical data. In the healthcare sector, various wearable medical devices are employed to track patient health status, detect irregularities, and manage medical conditions.

For example, hospitals require extensive collection of electronic health records (EHR) to develop robust medical models. However, due to the sensitive nature and confidentiality of medical information, obtaining an authentic dataset remains a challenge. FL addresses this issue by ensuring data remains anonymous, thereby eliminating several obstacles related to data exchange.

IoT and Edge Computing: FL is crucial in IoT environments, where devices collaboratively train models for real-time decision making. Edge devices in air pollution monitoring can take advantage of FL for improved performance and privacy. FL applications in air pollution monitoring can revolutionize the field by facilitating the collaborative examination of information from multiple sensors and sources, including stationary monitoring stations, mobile units, and citizen-contributed data, while ensuring data security. This approach facilitates real-time monitoring, predictive modeling, and localized insights, allowing immediate detection of pollution spikes, informed policymaking, and targeted interventions. FL also optimizes resource allocation and encourages community participation, ensuring a comprehensive and responsive approach to combating air pollution challenges on a global scale while respecting individual data privacy.

2.6 Split Learning

In distributed ML, where diverse devices and servers collaborate, Split Learning emerges as a groundbreaking paradigm. Unlike traditional approaches that run the entire machine learning model on a single device, Split Learning ingeniously partitions the model, enabling segments to operate independently on different computing entities. The essence of Split Learning lies in its model division strategy. The architecture of the ML system is broken down into separate components, with the first section covering the input layer and a subset of hidden layers. The remaining hidden layers and the essential final layer make up the second segment. This separation allows computational tasks to be divided, laying the groundwork for collaborative and privacy-conscious ML.

Training a split model follows a unique procedure. The first segment of the model processes the input data and produces intermediate outputs. Then, the second segment utilizes these intermediates to finalize the model's computations and generate the ultimate result. This not only distributes computational effort but also enhances data privacy. The data handled in the initial phase is obfuscated, strengthening security and confidentiality.

Split Learning offers numerous advantages, including enabling ML models to run on edge devices with constrained computational capacity, as the intensive computations are shifted to a central server. This method is particularly beneficial in situations where maintaining data privacy is critical, as sensitive details remain dispersed and fragmented between devices. Furthermore, Split Learning promotes efficient resource utilization in distributed environments,

unlocking scalability and adaptability.

Split Learning represents a transformative approach to distributed ML. By dividing ML models and orchestrating a balance between edge devices and servers, it bridges the gap between computational efficiency and data security. Split Learning is poised to be a key innovation in the domain of distributed ML, offering cutting-edge solutions for protected and cooperative model training.

2.7 Optimization Techniques

Adam Optimize: Adam (short for Adaptive Moment Estimation) is an optimization algorithm designed to modify the weight parameters of the neural network during training. It merges elements of two well-known optimization strategies: RMSprop and momentum.

Adaptive Learning Rates: Adam dynamically fine-tunes the learning rate for each parameter separately, allowing the network to reach convergence rapidly across various dimensions.

Momentum: Similar to other momentum-based optimization techniques, Adam maintains a record of past gradients to speed up convergence. It incorporates both a gradient and a squared gradient component.

Bias Correction: To counteract the initial bias of the moving averages, Adam is extensively utilized for training deep neural networks because of its effectiveness in handling non-stationary and noisy objective functions. It is especially appropriate for scenarios where the data distribution or gradients may change over time, making it a great option for convolutional neural networks (CNNs) and other advanced deep learning architectures.

2.8 Frameworks and Tools

2.8.1 TensorFlow

is a tool that emerges to empower developers. This programming interface acts as a link connecting the complexities of ML algorithms with the diverse world of heterogeneous computing. At its core, TensorFlow acts as a conduit, simplifying the complexities inherent in ML workflows. Its design is rooted in abstraction, enabling users to express and implement ML algorithms without delving into the minutiae of low-level operations. This abstraction elevates the general ease and efficiency of setting up both training and production environments.

One of TensorFlow's standout features is its adaptability. It is compatible with multiple programming languages, most notably Python and JavaScript. This cross-language capability allows TensorFlow to integrate seamlessly with a wide array of computing platforms. The impact of TensorFlow reverberates in various computing environments. Its versatility transcends the boundaries of traditional ML, extending its reach to specialized hardware accelerators, cloud-based clusters, and even resource-constrained edge devices. This adaptability enables experts to harness the full potential of their hardware, optimizing performance for particular applications. TensorFlow stands as a pillar of assistance in the field of ML. By abstracting com-

plexity and fostering adaptability across programming languages and computing platforms, it simplifies the journey from algorithmic conception to real-world deployment. TensorFlow's presence continues to reshape the ML landscape, enabling innovation and progress in a dynamic and heterogeneous computing world.

2.8.2 PyTorch

PyTorch is a robust framework designed to construct and train deep learning models, originating from Facebook's AI Research Lab (FAIR). It provides a flexible and intuitive interface for neural network development and stands out for its support of dynamic computational graphs, permitting modifications during the training process.

Key Features of PyTorch

- Dynamic Computational Graph: Unlike traditional frameworks, PyTorch builds graphs at runtime, offering greater flexibility for complex algorithms.
- GPU Support: PyTorch provides optimized support
- for GPU computations, speeding up model training significantly.
- Integration with Python Ecosystem: Built on Python, PyTorch is user-friendly and integrates seamlessly with libraries such as NumPy, SciPy, and Pandas.
- TorchScript: Enables exporting models for production use by converting them into an optimized form independent of Python.

Usage in Federated Learning PyTorch is well suited for FL due to its flexibility and modularity. Many platforms, such as PySyft and Flower, leverage PyTorch to implement FL, providing tools for model sharing and synchronization across distributed devices.

2.8.3 Docker

Docker is a powerful platform that enables the creation, management, and execution of applications using containers. Containers are a way of packaging an application along with all its dependencies and isolating it from the environment in which it runs. This makes Docker an ideal tool for building and deploying applications in a consistent and portable manner.

When using Docker for server and client nodes, each node can be placed in a separate container. This approach offers several advantages.

- Isolation: Each container is isolated from others, meaning that issues in one node do not affect other nodes.
- Portability: Containers can run on any system with Docker installed, without requiring additional configurations.
- Reproducibility: Using a Dockerfile, you can precisely define how a container should be built, making it easy to replicate the environment across different setups.
- Scalability: Docker simplifies scaling applications by allowing you to add or remove nodes as needed.

2.9 Related Work

This section explores studies on the deployment of FL in low-capacity devices, techniques to ensure data privacy, and adaptive strategies to improve efficiency in environments with limited resources.

Low-constrained Devices and Federated Learning: FL enables joint training over distributed systems while maintaining data confidentiality by keeping data local. Several studies have demonstrated the implementation of FL in practical settings.

Y. Gao et al. [40] carried out a thorough evaluation of two sophisticated AI model development approaches: FL and Partitioned Neural Networks (SplitNN). Their research explored multiple datasets, experimented with diverse model architectures, incorporated various user devices, and used different benchmarks to assess system performance. The study focused on two distinct data distributions: one with imbalanced data and another where data varied between devices. Model training was performed using Raspberry Pi systems while tracking processing power and RAM consumption, network overhead, and training duration. The results demonstrated that FL generally outperformed SplitNN, mainly due to its lower communication overhead.

Our research builds upon some of Gao's experimental findings on individual Raspberry Pi devices, particularly leveraging their setup using 1 and 5 epochs as benchmark epoch indicators to establish a baseline system.

Several studies have explored FL for edge computing environments, as reviewed in [41]. Specific categories of edge devices have been analyzed, including those within the Flower ecosystem, which incorporates devices such as Raspberry Pi, Android smartphones, and NVIDIA Jetson [42]. The Flower system tackles device diversity by implementing tailored software solutions for different clients. For example, the FL client for Android phones is developed in Java using TensorFlow Lite Model Personalization for Android Studio, while the FL clients for Raspberry Pi and NVIDIA Jetson are implemented in Python.

A review of existing research highlights various approaches that aim to optimize FL to reduce computational resource consumption. These strategies involve modifications to model training procedures and offloading certain tasks to alternative platforms. However, there remains a notable gap in evaluating FL's effectiveness in real-world wireless mesh network environments. Our study aims to bridge this gap by deploying FL on resource-limited devices

operating within mesh networks. This hands-on approach allows us to gather critical insights into optimizing FL for diverse user scenarios.

The study in [43] investigates the classification of multiclass chest disease by analysis of chest radiographs using a proposed CNN model. In addition, the authors present a dataset containing 28,833 chest X-ray images, covering COVID-19, other viral infections, bacterial pneumonia, lung opacity, and normal cases, collected from publicly available sources. FL was used to train the suggested models and experiments were conducted comparing centralized learning, distributed FL, and an FL approach optimized for lower communication costs. The findings indicate that FL can achieve high classification accuracy for chest diseases, positioning it as a viable alternative to conventional centralized training. The study results offer meaningful information that could encourage medical institutions to implement FL for diagnostic purposes.

The study presented in [44] introduces a theoretical structure that integrates peripheral computing for the analysis of medical data, relying on the information provided by the users. By incorporating remote computing, peripheral computing, the Internet of Things, wearable technology, and FL, the framework aims to give individuals more power over their personal health management. The primary goal is to promote the participation and responsibility of patients in the tracking and prevention of diseases, which is crucial for the sustainability of healthcare systems. The suggested framework enables the streamlined and flexible merging of user-contributed health along with behavioral insights.

Imteaj et al. [45] present a detailed examination of existing FL methodologies and the obstacles associated with their deployment on IoT devices, which often suffer from limited processing capabilities, constrained bandwidth, and insufficient storage capacity. Their work highlights the necessity of enhancing FL algorithms to manage these constraints effectively and proposes new research avenues to improve FL performance in such environments. This particular research delivers key understanding of how FL can be adapted for IoT systems possessing constrained computational capabilities, aligning well with the broader objectives of this study.

The authors in [46] analyze the use of FL in computing environments with limited resources, specifically deploying Raspberry Pi devices within a wireless network infrastructure. The study focuses on training a deep learning model based on a CNN with medical imaging data, particularly chest radiographs, while maintaining data privacy. The findings reveal that increasing the number of training iterations enhances the precision of the model, but also results in increased CPU consumption. Their study highlights the trade-offs between performance and resource ef-

ficiency, emphasizing the relevance of FL in privacy-sensitive domains such as healthcare. This research contributes practical knowledge on optimizing FL for low-power devices in real-world scenarios, offering valuable insights for future implementations.

The study by Zhang et al. [47] offers a thorough investigation of the relationship between energy consumption and resource management in ML operations, particularly in IoT devices. The research highlights the importance of balancing computational demand to prevent overheating and excessive power usage, which is crucial to ensure the longevity and effectiveness of IoT networks. The paper presents strategies to dynamically manage resource allocation based on workload and temperature fluctuations, ensuring that devices do not overheat while maintaining optimal performance. This strategy is especially significant for cases involving continuous ML operations, where overheating and high energy consumption are common issues.

The authors in [48] also examined FL for applications such as detecting depression via smartphone-based sensing. This approach keeps user data locally stored while enabling collaborative training of ML models, demonstrating that privacy can be maintained without compromising usability. The study also highlights the trade-offs between centralized and FL-based healthcare systems.

The study [49] discusses advancements in healthcare systems, such as the Smart Healthcare Monitoring System, which utilizes sensors and microcontrollers such as Raspberry Pi to track health parameters in real time. By integrating cloud storage with mobile applications, these solutions support continuous monitoring of health conditions and faster medical decision-making, highlighting the revolutionary role of the IoT in contemporary healthcare.

When considering FL for devices with limited resources, it is crucial to ensure that these technologies maintain privacy and data protection. The protection of sensitive information, whether personal or medical, is a fundamental aspect of FL and is the key to ensuring that these systems are reliable and secure. The following research works explore how FL can improve data protection and protect users from potential threats.

Privacy and Security through Federated Learning:

A crucial element of FL is confidentiality, which ensures that sensitive data remains stored locally on user devices. Several studies have suggested techniques to improve privacy within FL frameworks.

The study in [50] emphasizes the role of privacy in the training of medical data sets and introduces Dopamine, a system that aims to ensure FL that preserves privacy. This method

leverages medical data sets to develop deep neural networks (DNN) for medical diagnosis by incorporating FL with Differentially Private Stochastic Gradient Descent (DPSGD), allowing for secure and distributed model training.

In another research effort [51], the authors combined blockchain technology with FL to develop ML models while ensuring data confidentiality. This method was particularly significant for the Industrial Internet of Things (IIoT), where large-scale data is continuously generated by interconnected devices. The study stressed the need to prevent data breaches in industrial environments and introduced a secure data sharing protocol utilizing blockchain while integrating privacy-focused techniques within FL. Their findings demonstrated that blockchain enhances model sharing in FL while ensuring data confidentiality.

Similarly, the BlockFL system, introduced in [52], incorporated blockchain into FL by replacing a centralized node with a distributed ledger to exchange model updates between mobile devices. The study evaluated the learning efficiency of this decentralized approach, where multiple mobile devices locally trained models while leveraging blockchain for secure and transparent communication.

In a healthcare-specific scenario, Passerat-Palmbach et al. [53] explored how blockchain could facilitate the orchestration of FL within healthcare organizations. Their approach ensured privacy and data integrity by enabling contributors to improve ML models while maintaining anonymity. They implemented a tracking mechanism to monitor network activity without exposing the identities of the participants.

In contrast, the work in [54] proposed an alternative to the traditional FL framework by using the Interplanetary File System (IPFS) instead of a centralized server. This decentralized approach allowed multiple nodes to participate and manage the FL process. Furthermore, ML models were partitioned and distributed among different nodes, which makes this method particularly beneficial for resource-limited devices.

The study presented in [55] introduces Communication-Optimized Federated Averaging (CE-FedAvg), a refined version of the FedAvg method, designed for FL applications within IoT environments. This approach improves data security by ensuring that FL operations occur on edge servers and gateways, removing the need to transfer raw information to a centralized location. CE-FedAvg integrates a refined Adam optimization technique and advanced compression methods to decrease network exchange cycles and reduce the volume of data transmitted to users. Comprehensive evaluations using datasets such as MNIST and CIFAR-10, tested

under both IID and non-IID scenarios, indicate that CE-FedAvg achieves faster stability and significantly lowers overall data transmission compared to the standard FedAvg. Moreover, real-world testing conducted with an experimental set-up of a Raspberry Pi validates that CE-FedAvg achieves target accuracy faster, reinforcing its value in enhancing FL efficiency for edge-based computing applications.

In [56], researchers utilize FL to classify medical images, with a strong emphasis on improving data security in AI of healthcare. Their proposed method, FedSLD, leverages label distribution information to address challenges arising from data variability in FL settings. Using shared label distribution details, FedSLD enhances model stability and accuracy in medical image classification. The study assesses its effectiveness with datasets including MNIST, CIFAR-10, OrganMNIST, and PathMNIST. Although this research presents an innovative approach to training ML models in decentralized and privacy-sensitive medical environments, it focuses primarily on model performance while not comprehensively addressing privacy and security concerns, both crucial factors in healthcare-oriented ML systems.

Research [57] introduces the P2FLF framework, a novel strategy to improve privacy in FL within fog computing environments. This framework maintains data security while preserving model accuracy by processing data locally on fog nodes and integrating techniques like differential privacy and encryption. The findings reveal that this method minimizes communication latency, positioning it as an effective solution for IoT-based applications.

The study in [58] explores privacy-preserving and secure FL approaches for medical imaging. It investigates technologies such as FL, data privacy protection, encryption that allows computations on encrypted data, and secure computation involving multiple parties, as well as hardware-based security solutions. The paper underscores the necessity of multidisciplinary research to facilitate the broad integration of secure and private AI, which is vital to advance innovations in healthcare and beyond.

Reviewing the literature on privacy in FL, it is evident that there are multiple techniques to enhance data security. A widely used approach is differential privacy, which helps protect individual data points while contributing to the overall model. Furthermore, external technologies such as blockchain provide further security by enabling decentralized and tamper-resistant data exchange. Moreover, ML applications that rely on sensitive personal data, including medical records and private sensor data from IoT devices, can significantly benefit from FL by preserving privacy while enabling collaborative learning.

Although privacy-preserving techniques are crucial, resource limitations in low-capacity devices demand further exploration, as discussed in the next category.

Adaptive FL for Devices with Limited Capacity:

Addressing the challenge of training on devices with limited capacity, existing studies focus on methods that emphasize data privacy, which is essential to protect sensitive personal health data throughout the training process.

FL faces multiple challenges, especially in resource-constrained IoT devices, particularly in wireless environments where computational capacity, network bandwidth, and energy resources are limited.

FL has evolved with the development of adaptive algorithms aimed at optimizing performance in resource-constrained and heterogeneous environments. Traditional FL approaches, such as those based on static training parameters, often struggle in dynamic environments where device capabilities and network conditions vary significantly. To address these challenges, recent research has focused on introducing adaptability to FL models.

The paper [59] investigates optimal resource allocation strategies for FL workflows within virtualized environments. The study addresses the challenge of determining how to allocate resources effectively within FL workflows, considering factors such as computational and network capabilities, dataset complexity, and FL workflow specifics. Two scenarios are explored: FL execution across a limited set of testbed nodes and hosting multiple parallel FL workflows on the same nodes. The research highlights sub-optimal configurations of cloud orchestrators for FL workflows and identifies variations in computational footprints across different ML models. The proposed strategies aim to mitigate computational interference and improve overall performance of the FL pipeline, contributing to more efficient resource allocation in virtualized managed environments designed for FL.

This paper [60] explores the performance of FL under a novel scheduling strategy for workload distribution. The study includes experiments using the MNIST and CIFAR-10 datasets, analyzing independent and dependent training data distributions. It evaluates the ”FC” (Fast Converge) policy compared to standard policies and centralized training methods, focusing on the parameter ϕ and its role in influencing performance. The results indicate that the FC policy consistently exceeds the baseline strategies, effectively balancing the latency and precision levels.

The work presented in [61] introduces FL as a method to train deep neural networks across

distributed data sources, while ensuring data security and reducing communication costs. This approach lays the foundation for cooperative model training in dispersed computing environments. Certain aspects of this work serve as inspiration for our research.

Furthermore, [62] the authors address the evolving nature of FL frameworks by proposing an adaptive optimization algorithm tailored for FL settings. This method modifies the learning rate values along with the hyperparameters according to user-specific attributes and network states, thus improving the stability of model training and overall efficiency.

In a different domain, [63] presents a novel FL method specifically designed for wireless mesh networks. To address the difficulties related to training machine learning models within decentralized settings that have restricted computational capacity and inconsistent connectivity, the authors propose a server-side adaptive framework. This approach allows the main server to dynamically fine-tune model training, taking into account network stability and client capabilities, ensuring optimized resource utilization and enhanced model performance. Expanding on these principles, our research focuses on accommodating node diversity by dynamically adjusting the computational workload, ultimately leading to improvements in the overall accuracy of the global model.

Building on the investigation of bandwidth and CPU-aware FL adaptation, the third paper broadens the adaptive strategy by considering diverse client environments and real-time resource limitations. ”AdaptiveMesh” introduces additional enhancements to refine model optimization for efficient training across a wireless mesh network where devices have varying processing capabilities.

The FedTCR study [64] addresses inefficiencies in communication and delays caused by late client participation within federated learning by introducing structured computational clusters and adopting a group-based training approach. This method reduces communication overhead and ensures that even devices with lower processing power can still participate effectively in training. Although FedTCR aims to reduce communication overhead and maintain an equilibrium in client participation, AdaptiveMesh is designed for real-time adaptation by efficiently managing resources. Prevents overheating and optimizes resource utilization, leading to improved training stability and convergence in various settings.

Similarly, FedAdapt: Adaptive Offloading for IoT Devices in FL [65] introduces a novel framework that enhances localized training by transferring layers of the deep learning model to centralized servers. Using reinforcement learning and clustering techniques, it determines

which layers should be offloaded per device, thus reducing training time while adjusting to network variations. While FedAdapt emphasizes offloading layers for improved training, AdaptiveMesh optimizes training hyperparameters, including image batch sizes and the number of epochs, based on device processing capabilities, considering CPU usage and accuracy trends. This mechanism mitigates excess workload issues and overheating while promoting efficient allocation of computational resources, ultimately improving overall model performance.

A research effort [66] introduces Fed-RAC, which uses clustering based on resource availability to enhance both training and communication efficiency on various devices. By dynamically determining the optimal number of groups using Dunn Index calculations, Fed-RAC adapts to different degrees of data diversity, ensuring effective aggregation and training performance. It employs a multi-layered learning approach utilizing knowledge transfer to enhance the training quality of lightweight models within grouped environments. While Fed-RAC relies on resource-driven clustering, AdaptiveMesh dynamically adjusts learning configurations based on real-time client device performance metrics to optimize resource usage and model precision. Unlike Fed-RAC, which depends on a hierarchical structure to support low-powered models, AdaptiveMesh reduces client-side burdens by implementing real-time parameter modifications.

Furthermore, [61] explores federated learning as a reliable solution for training complex neural networks across geographically dispersed data sources, while ensuring secure data handling, which is crucial for collaborative model development in decentralized environments.

In terms of FL's adaptive mechanisms, [62] suggests an optimization technique specifically designed for FL architectures. This strategy refines hyperparameter tuning, including learning rate adjustments, in response to client-specific conditions and fluctuations in network connectivity, leading to superior model stability and performance. The study also considers network-related factors, such as bandwidth restrictions.

Another significant contribution [67] explores the integration of data from various healthcare organizations while protecting security and privacy. The study presents a dynamic client selection approach (ACS), which seeks to enhance client sampling for training in each round. This approach considers the capabilities of the device and the properties of the data set, thus increasing the overall effectiveness of FL within the medical field. The study focuses primarily on medical datasets that include chest radiographs.

An innovative approach [68] focuses on efficiently distributing computational tasks in FL

at the edge of wireless networks, reducing latency while maintaining optimal performance. Using Lyapunov-based probabilistic optimization, the authors fine-tune wireless transmission configurations and resource distribution to balance energy usage, training speed, and overall learning effectiveness. The proposed technique is applied to adaptive least mean squares estimation, alongside experimental simulations, demonstrating its efficiency in enabling low-latency and cost-effective FL at the wireless network edge.

A recent study [69] introduces a method to enhance FL performance by fine-tuning client-specific hyperparameters. The key elements of this technique involve dynamically adjusting variables such as batch size and learning rate, based on each client's computational power and data characteristics.

Similarly, RingSFL [70] suggests a flexible FL framework designed for various client environments, aligning with our research focus. This system addresses challenges related to variations in data distributions and differences in client performance by implementing a hierarchical ring-based structure to categorize clients effectively. This model promotes seamless communication and collaboration in distributed networks. The intelligent client sampling mechanism chooses participants based on data patterns and efficiency metrics, guaranteeing equitable participation in global learning. To maintain secure communication protocols, the participants in this study contribute to training based on their processing power, memory capacity, and network constraints.

In the field of medical image analysis, [71] addresses the challenges of FL by proposing FedRSMax, an innovative aggregation strategy that improves the integration of collaborative learning in FL. The primary advancement of FedRSMax lies in its adaptive aggregation system, which effectively balances the impact of locally trained local models from various healthcare institutions. Unlike conventional FL aggregation techniques that rely on uniform averaging, FedRSMax dynamically adjusts the aggregation parameters, tailoring them to the performance quality of each model. This intelligent system guarantees that more accurate models contribute with greater weight to the final global model.

Furthermore, [72] proposes an FL solution designed for wireless mesh networks, addressing the complexities of training ML models in decentralized environments with restricted resources and unreliable connectivity. The study suggests a server-side adaptive mechanism that enables the central server to fine-tune the training process based on real-time network status and client capabilities. This adjustment strategy improves resource management and

promotes better model convergence in wireless mesh networks. The experimental findings confirm that this technique improves the accuracy of the model while minimizing communication overhead.

Although recent research has investigated adaptive strategies to improve FL efficiency, many fail to adequately consider network variability and hardware limitations. In addition to the existing methods discussed in these studies, our research prioritizes adapting to differences in client node capabilities through dynamic workload distribution, ensuring that more powerful nodes process a larger share of computations, ultimately leading to improved global model performance.

FedAda [73] presents an FL algorithm designed for mobile edge computing environments. The main aim of FedAda is to improve the speed of convergence by allocating workloads across devices with different capabilities. This method becomes significant in situations where device abilities vary and quick convergence is essential for maintaining system performance.

FedAda can adjust to the capabilities of devices and network conditions to achieve convergence and efficiently optimize resource usage. A crucial feature for applications in environments with limited resources.

FedALA [74] introduces an aggregation approach that enhances the learning process by dynamically modifying local updates to efficiently accommodate different heterogeneous client capabilities. This technique proves beneficial, in tailoring FL for clients with resource limitations such as IoT devices or mobile edge nodes. FedALA significantly reduces communication overhead and improves convergence speed by optimizing local aggregation weights. It guarantees that even devices, with resources, can make contributions without impeding the overall system performance significantly. By effectively managing device heterogeneity, FedALA addresses challenges like high CPU load and bandwidth fluctuations, aligning well with the goals of this research in optimizing FL for real-world applications.

Another contribution is the study [75] on FL with adaptive aggregation in edge-cloud environments. The study focuses on reducing communication in FL by using aggregation methods. Its aim is to strike a balance between communication efficiency and the optimization of learning outcomes in edge cloud environments with resources. By reducing communication costs and upholding precision levels simultaneously, this initiative enhances the system performance, proving beneficial, in situations where bandwidth limitations are a concern.

Moreover, FedLC [76] proposes a structure to speed up FL by handling device diversity

and reducing the delay, between updates. This study focuses on improving FL by addressing device diversity and reducing the waiting time between updates. By introducing methods, the FedLC framework effectively manages updates, improving the learning pace and productivity of federated systems.

This is especially crucial in environments where devices have varying computational capabilities, as the framework helps preserve learning accuracy while reducing resource strain.

The research discussed in the publication [77] introduces a FL model specifically designed for IoT settings characterized by devices, with resources. The innovative FedPARL model adopts a three-layered strategy to tackle FL obstacles like variations in system and statistics, among clients, and constraints related to capacity and the potential inclusion of inaccurate client data updates.

The paper [78] introduces FedCAda, an adaptive client-side optimization algorithm for federated learning (FL) that aims to accelerate convergence while maintaining stability. Unlike traditional FL methods like FedAvg, which lack adaptivity and can suffer from client drift, FedCAda leverages an Adam-like approach to adjust first and second moment estimates on the client side. The algorithm aggregates adaptive parameters on the server side to reduce heterogeneity and improve convergence. Experiments on public datasets (CIFAR-10, Fashion-MNIST, and Shakespeare) demonstrate that FedCAda outperforms state-of-the-art methods in adaptability, convergence, and stability, particularly in non-IID data settings. The paper also explores different adjustment functions for adaptive parameters and highlights the importance of balancing acceleration and stability in FL.

The paper [79] introduces a framework for FL in heterogeneous wireless networks, addressing challenges such as non-IID data, computational limitations, and asynchronous model updates. The framework includes Adaptive-Mixing Aggregation (AMA) to stabilize training by balancing global and local models, Feature-Extractor Sharing (FES) to reduce computation for resource-limited devices, and an asynchronous aggregation scheme to handle delayed updates. Experiments with UAVs as FL clients show that AMA-FES improves accuracy and stability without extra computation or communication costs, especially in non-IID settings. The framework is energy-efficient and scalable, making it suitable for real-world applications such as image classification in dynamic wireless environments.

The paper [80] introduces Multi-task Clustering personalized Federated Learning, a framework addressing data heterogeneity in FL by clustering nodes with similar data distributions

and aggregating models within clusters. It incorporates multi-task learning, dividing cluster models into expert layers for global knowledge sharing, and uses adaptive weights for personalized local models. Using carbon emission prediction data, MCFL outperforms FedAvg and traditional clustering FL in metrics such as MAE, RMSE, and MAPE, demonstrating robustness across diverse data distributions. This approach improves personalized predictions, helping to reduce carbon.

In the paper [81] the authors propose a novel algorithm called AdaFedAdam to improve fairness and convergence in federated learning (FL), particularly in non-independent and identically distributed (non-IID) data settings. By formulating the problem as a dynamic multi-objective optimization (DMOO), they ensure that the model performs fairly across all participants while maintaining fast convergence. AdaFedAdam dynamically adapts hyperparameters based on the certainty of local updates, reducing bias, and accelerating the training process. Experiments on standard datasets demonstrate that AdaFedAdam outperforms existing algorithms such as FedAvg, FedAdam, FedNova, and q-FedAvg, offering a better balance between fairness and convergence. Furthermore, the algorithm shows robustness to data and resource heterogeneity, making it a promising solution for real-world FL applications.

In the paper [82], the authors address the challenge of efficiently selecting clients in federated learning (FL) systems, particularly in resource-constrained environments such as mobile edge computing (MEC). They propose a novel approach using Deep Reinforcement Learning (DRL) to adaptively select clients for model training, aiming to minimize energy consumption and training delay while maintaining model accuracy. The authors model the client selection problem as a Markov Decision Process (MDP) and employ a Double Deep Q-Network (DDQN) algorithm to learn optimal client selection policies without prior knowledge of the system dynamics. Experimental results in MNIST and Fashion-MNIST datasets demonstrate that the proposed approach significantly reduces energy consumption (up to 50%) and training delay (up to 20.70%) compared to static algorithms such as random and greedy selection. The method also shows robustness across different tasks and non-IID data settings, making it a promising solution for real-world FL applications in resource-constrained environments.

In the paper [83], the authors propose a novel approach to address the challenges of resource optimization and data privacy in federated learning (FL) systems, particularly for real-time image classification tasks in IoT environments. The authors introduce an adaptive model communication scheme that leverages Deep Q-Learning (DQL) to optimize resource allocation

in a Network Functions Virtualization (NFV)-enabled Mobile Edge Computing (MEC) system. The proposed scheme dynamically adjusts virtual resources based on network conditions, such as congestion states, to ensure efficient computation offloading and global model aggregation in FL. By integrating Software-Defined Networking (SDN) and NFV, the system optimizes the placement of virtual network functions (VNFs) and flow configurations, reducing communication delays and improving the quality of service (QoS). Experimental results demonstrate that the proposed approach significantly reduces packet drop ratios, improves packet delivery ratios, and decreases control delays compared to traditional FL schemes, while maintaining high classification accuracy. This work highlights the potential of combining FL with DRL and NFV for real-time, resource-constrained IoT applications.

In the paper [84], the authors propose a novel approach to improve the efficiency and privacy of federated learning (FL) in healthcare applications. The method integrates Adaptive Gaussian Clipping Differential Privacy (AGC-DP) with a Discrete Fourier Transform (DFT) aggregator to reduce communication costs and enhance data security. By applying DFT to compress and encrypt gradients before adding noise, the approach significantly reduces the size of transmitted data, leading to lower communication overhead. The authors evaluate their method on healthcare datasets, such as the PIMA Indian Diabetes dataset and Breast Cancer Histopathology images, demonstrating improved accuracy, reduced communication costs, and faster training times compared to existing differential privacy techniques such as RDP, DP-SGD, ZcDP, LDP-Fed, and DP-AdapClip. This work addresses the challenges of implementing FL in resource-constrained healthcare settings, offering a balanced solution between privacy preservation and computational efficiency.

In the paper [85] the authors propose FedSIM, a novel method for personalized federated learning (FL) that takes advantage of server-side data to improve metagradient calculations, thus improving personalization performance while reducing computational overhead on the client side. Unlike traditional personalized FL approaches, which rely heavily on clients to compute computationally expensive second-order gradients, FedSIM shifts this burden to the server by utilizing its computational resources and independent data. The method introduces a custom loss function with L_2 regularization, approximates first-order meta-gradients using weight differences, and estimates second-order meta-gradients through Hessian-free approximation using server data. Empirical evaluations on benchmark datasets demonstrate that FedSIM achieves higher accuracy, faster convergence, and reduced communication costs

compared to existing methods, making it particularly suitable for resource-constrained environments. This work addresses the challenge of non-i.i.d. data distributions in FL by effectively utilizing server-side information, offering a balanced approach between privacy preservation and computational efficiency.

In the paper [86] the authors propose a novel federated learning (FL) strategy called FedNets, designed to address the challenges of non-independent and identically distributed (non-IID) data and privacy concerns in FL. Unlike traditional FL approaches that share model weights, FedNets employs ensemble learning, allowing clients to share entire models rather than just parameters. This approach leverages graph embedding theory to reduce the complexity of deep neural networks (DNNs) by representing them as graphs and clustering their embeddings. The method introduces a custom loss function with L_2 regularization, approximates first-order meta-gradients using weight differences, and estimates second-order meta-gradients through Hessian-free approximation using server data. Experimental results on the Federated CIFAR100 dataset demonstrate that FedNets outperforms state-of-the-art FL algorithms like Federated Averaging (FedAvg) and Adaptive Federated Optimisation (FedYogi) in terms of accuracy, achieving up to 63% and 92% improvement, respectively. Additionally, FedNets enhances privacy by sharing only a subset of ensemble models, reducing the risk of sensitive data exposure. The approach is particularly suitable for resource-constrained edge devices, offering a balance between accuracy, privacy, and computational efficiency.

In [87] work, the authors introduce a novel approach to enhance energy efficiency in Federated Learning (FL) through adaptive sparsification and quantization techniques. Their method, termed Sparsified and Quantized Federated Learning (SQFL), aims to minimize energy consumption during the training of machine learning models across diverse devices, while considering latency constraints and desired accuracy. They develop an optimization framework that incorporates quantization for both local computations and transmissions, as well as sparsification of the transmitted parameters. Using an iterative algorithm, they identify the optimal quantization and sparsification parameters, ensuring a balance between energy consumption, execution time, and model accuracy. Numerical results demonstrate that SQFL achieves a significant reduction in energy consumption compared to baseline FL approaches, while maintaining high accuracy and adhering to latency constraints.

The reviewed studies collectively advance FL by addressing privacy, resource constraints, and adaptivity. However, gaps remain in integrating these aspects into a unified framework

suitable for dynamic and resource-constrained environments. This thesis contributes to filling these gaps through the development and evaluation of the AdaptiveMesh algorithm, designed to optimize FL in real-world scenarios.

Part II. System Model and Algorithms

Chapter 3

System Model

3.1 Introduction

FL has attracted considerable attention within the research community as a collaborative training framework that allows clients to build models together while preserving the confidentiality of their individual data. This methodology has emerged as a crucial focus area in domains involving smart devices and wire-free communication technologies, including 5G networks, where distributed nodes offer valuable insights while ensuring confidentiality [88], [89]. FL functions as a distributed machine learning method in which multiple participants or computing units, such as mobile and wearable devices, collaborate to develop a shared model under the oversight of a centralized cloud-hosted server. Figure 3.1 illustrates this concept. Training information remains stored locally on individual devices, ensuring both privacy and security.

The FL algorithm follows a structured sequence: Initially, the server establishes a global model and shares it with all participants involved. Each participant then refines the global model by leveraging its own locally available data over multiple training rounds, commonly referred to as epochs. After the local training phase has ended, the participant sends the revised model back to the central server. The server then compiles all the modifications received from the clients and integrates them to produce an enhanced global model. This process continues iteratively until the model achieves stability or reaches a predefined count of training rounds (repetitions). Each participant retains a dataset containing training examples, as depicted in the associated figure.

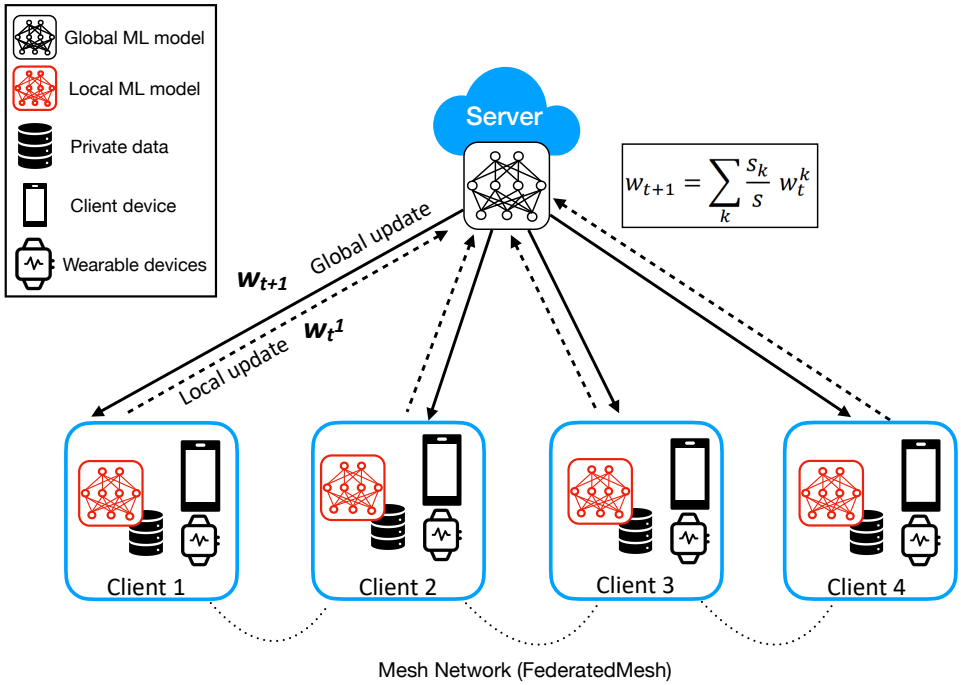

Figure 3.1: FL architecture involves interaction between the central server and client devices, such as smartphones, wearable technology, and similar IoT-enabled systems.

The need for privacy-preserving machine learning at the edge has led to the development of several adaptive FL algorithms to optimize resource usage while maintaining model accuracy. This chapter presents a unified framework for FL in resource-constrained environments, synthesizing methodologies from four studies to address privacy, adaptability, and scalability. The system model integrates hardware configurations, adaptive algorithms, and experimental validations across heterogeneous devices and cloud platforms, aligning with the five key contributions of the thesis. In the following, we detail the architecture, algorithms, and workflows that underpin these contributions.

The proposed system is based on four key contributions:

- FederatedMesh, a secure and privacy-preserving FL system for edge devices.
- BACA, a bandwidth- and CPU-aware adaptive FL algorithm.
- AdaptiveMesh, an adaptive FL algorithm for optimizing resource usage in heterogeneous environments.
- A comparative evaluation of real and virtual devices to analyze resource trade-offs.

These contributions have been evaluated through extensive experimental setups, using a range of heterogeneous devices including Raspberry Pi, Mini PCs, laptops, and virtual machines.

3.2 System Architecture and Design

This chapter outlines the structural design of our system, including the hardware configurations and the software stack utilized in the core studies of this thesis. Each component of the system introduces advances in FL for resource-constrained settings, beginning with privacy-focused approaches such as FederatedMesh, progressing to adaptive and bandwidth-efficient strategies such as BACA and AdaptiveMesh, and concluding with a comparative evaluation of FL performance across various platforms. The study of adaptive federated learning is made up of four key components:

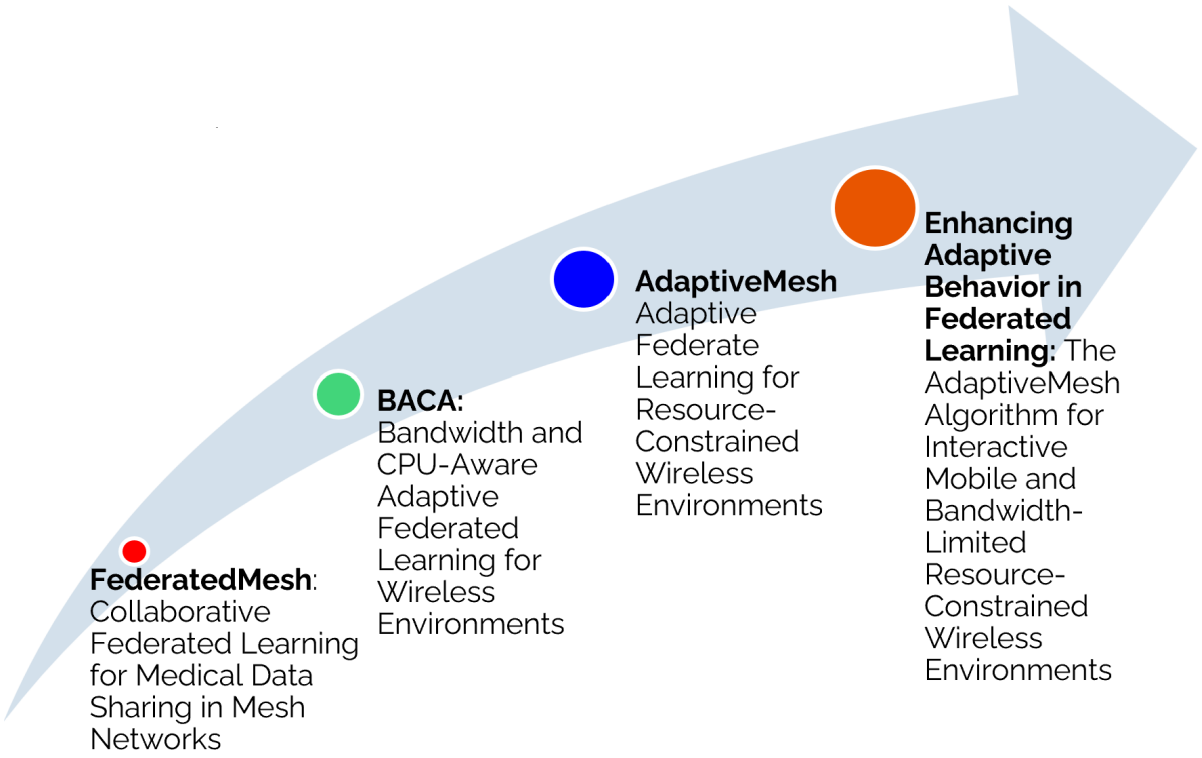

Figure 3.2: System components (algorithms) proposed.

1. **FederatedMesh:** Collaborative Federated Learning for Medical Data Sharing in Mesh Networks. Initially developed as a FL system to analyze the performance of resource-constrained devices. It aimed to enable collaborative learning in a distributed mesh network while preserving data privacy in sensitive applications like healthcare. The focus was on minimizing communication overhead while ensuring decentralized model training in a heterogeneous IoT environment.

2. **BACA:** Bandwidth and CPU-Aware Adaptive Federated Learning for Wireless Environments. As challenges in device heterogeneity and network instability were identified,

the BACA algorithm was introduced to dynamically adjust training parameters based on real-time CPU utilization and bandwidth availability. This adaptation allowed efficient training on various edge devices, ensuring balanced computational loads and improved energy efficiency.

3. **AdaptiveMesh:** Adaptive Federated Learning for Resource-Constrained Wireless Environments. The AdaptiveMesh algorithm extended BACA by integrating temperature monitoring, training time optimization, and context-switching mechanisms. Dynamically adjusts model aggregation and workload distribution across devices, preventing overheating, reducing training time, and improving the overall stability of federated learning in low-power environments.

4. **Adaptive Behavior in Federated Learning**. Enhancing Adaptive Learning Strategies for Scalable FL Deployment. This phase focused on further refining AdaptiveMesh by introducing advanced statistical models to analyze relationships between resource metrics and model accuracy. The system was compared with traditional FL algorithms (FedAda, FedALA, and NAIVE FL), demonstrating significant improvements in resource efficiency, training stability, and model convergence.

3.2.1 Federated Learning Framework

FL functions through a decentralized network architecture, where numerous clients work together to train a shared model without exchanging their raw data. Instead, each client trains a local model and sends only the essential model updates to a central server for aggregation. This approach ensures data privacy while facilitating collaborative learning.

The system includes Raspberry Pi devices, Mini PCs, and laptops that collaborate to train ML models using FL. Key performance indicators such as CPU usage, temperature, bandwidth consumption, and memory utilization are analyzed during the FL training process on healthcare datasets. Furthermore, the study evaluates the accuracy of the model with varying numbers of edge devices and examines the training time, considering the impact of real-world wireless mesh networks.

FederatedMesh: Secure FL System for Edge Devices. The FederatedMesh system [46] was designed to securely train ML models in a wireless mesh network consisting of IoT devices. It aimed to address:

1. Addressing data privacy issues by ensuring that raw data remain on client devices.
2. Network efficiency in bandwidth-constrained environments.
3. Experiments were conducted using Raspberry Pi devices to assess accuracy, CPU load, and network performance.

3.2.2 Adaptive Federated Learning Algorithms

To improve efficiency and reduce computational costs, two adaptive FL algorithms were designed: BACA and AdaptiveMesh.

BACA: Bandwidth and CPU-Aware Adaptive FL

BACA [20] adaptive approach by incorporating the network bandwidth as a key parameter. It ensures that:

1. Devices with higher CPU and bandwidth availability train on more data.
2. Devices with limited resources receive reduced workloads, preventing overloading.
3. The system dynamically adjusts training time, epochs, and data volume based on real-time monitoring.

AdaptiveMesh: An Adaptive Learning Algorithm AdaptiveMesh [90] The approach extends the Baca framework by introducing an adaptive training approach that dynamically adjusts the training parameters based on:

1. CPU utilization
2. CPU Temperature
3. Bandwidth usage
4. Memory availability
5. Model accuracy trends

By monitoring system resources, AdaptiveMesh optimally allocates workload among clients, preventing device overload while improving model convergence.

3.2.3 Algorithm Comparison

AdaptiveMesh compared with other FL algorithms, including a non-adaptive (NAIVE) FL algorithm, as well as two existing adaptive approaches: FedAda and FedALA. The goal is to analyze their efficiency, adaptability, and resource utilization in different environments.

The comparison focuses on:

1. Performance evaluation of AdaptiveMesh against a baseline NAIVE FL.
2. Deployment differences between physical devices (Raspberry Pi, Mini PCs, Laptops) and virtual cloud environments (Google Cloud VMs).
3. Comparison of AdaptiveMesh with two existing adaptive approaches: FedAda and FedALA.

3.3 Experimental Setup

To evaluate the effectiveness of FederatedMesh, AdaptiveMesh, and BACA, extensive experiments were conducted on both physical and virtualized environments. In the following, we explicitly outline the exact device counts, types, and network conditions for each study to avoid conflation and highlight variations.

3.3.1 FederatedMesh: Collaborative FL for Medical Data Sharing

The FederatedMesh study aimed to evaluate the performance of federated learning in wireless mesh networks, using Raspberry Pi devices as FL clients. It analyzed CPU usage, memory consumption, network performance, and model accuracy in different configurations.

System Architecture The system consists of:

- FL Server: A centralized node that coordinates the aggregation of the global model.
- FL Clients: Eight Raspberry Pi 4 Model B devices acting as training nodes.
- Wireless Mesh Network: A distributed communication network with Ubiquiti Nanostation M5 devices.

1. **Hardware Configuration**
    - Server: Laptop (Intel Core i5-8250U, 8GB RAM, Windows 11).
    - Clients: Raspberry Pi 4 Model B (Quad Core Cortex-A72, 1.5GHz, 8GB RAM, 128GB storage).
    - Network: Two wireless access points (APs), each connecting four Raspberry Pi devices.
2. **Software**
    - OS: Raspbian GNU/Linux 10 (buster).
    - ML Framework: PyTorch 1.8.0 with Keras API.
    - Programming Language: Python 3.7.3.
    - Deployment: Docker containers for server and client nodes.
3. **Experimental Setup**

- Dataset: Chest X-Ray Images (5,863 images, 6.7GB).
- Model: 6-layer CNN (420,000 parameters).
- Training Rounds: 100 iterations, batch size = 10 images.

4. **Findings**

- Increasing the number of clients and rounds improved the model accuracy but increased the CPU load.
- The wireless mesh network introduced communication delays, impacting synchronization.
- Resource constraints (CPU, RAM) led to a need for adaptive FL mechanisms → paving the way for BACA and AdaptiveMesh.

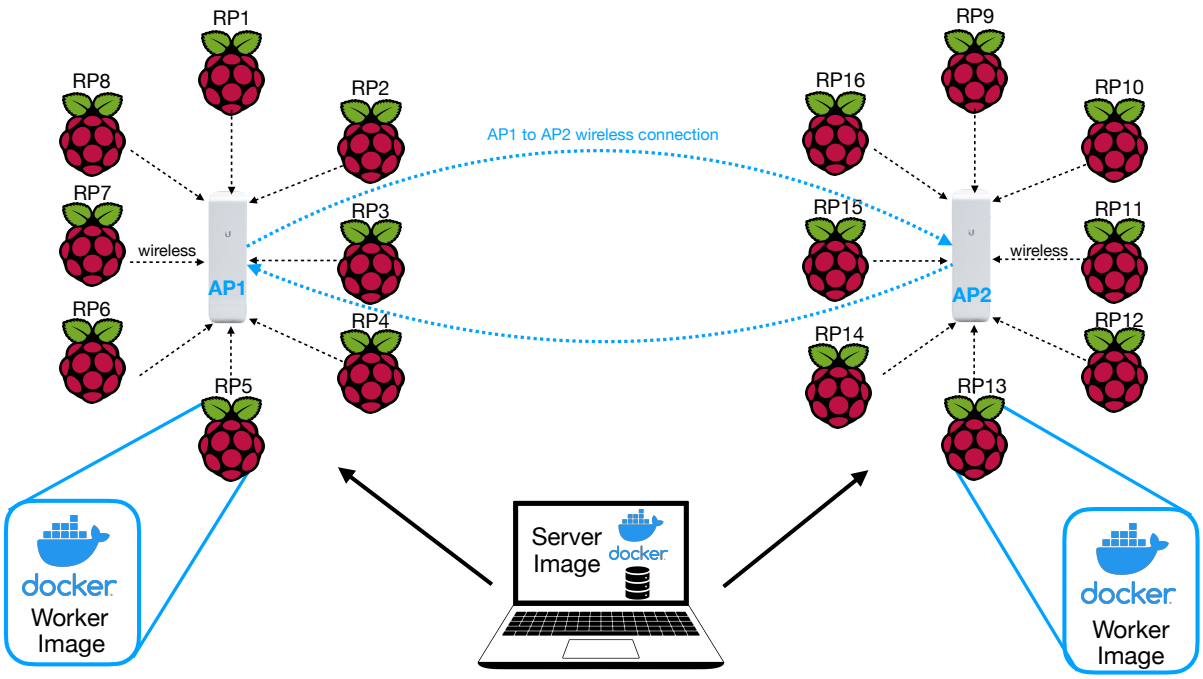

Figure 3.3: The devices utilized in the experimental environment for testing (FederatedMesh)

3.3.2 BACA: Bandwidth and CPU-Aware Adaptive FL

Objective

BACA (Bandwidth and CPU-Aware Adaptive FL) was designed to optimize federated learning by dynamically adjusting training loads based on real-time CPU and bandwidth conditions. The goal was to reduce computation overhead and improve FL performance in bandwidth-limited environments.

System Architecture

- Adaptive FL Server: Aggregates model updates and dynamically adjusts workload per client.
- Clients (heterogeneous devices): Different hardware platforms to simulate a real-world IoT scenario.
- Wireless network: A Wi-Fi-based network connecting all nodes, monitored for bandwidth fluctuations.

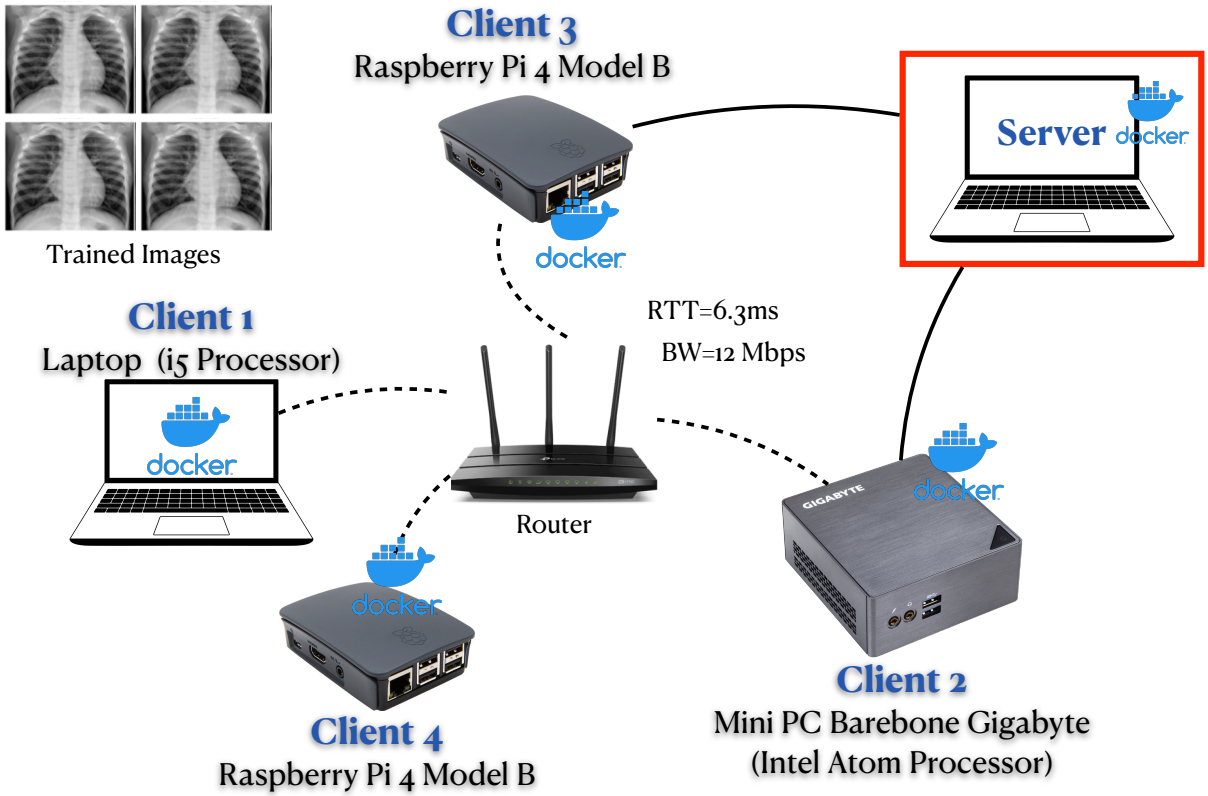

Figure 3.4: Setup nodes utilized in the experimentation (BACA).

1. **Hardware Configuration**
    - Server: Laptop (Intel Core i5-8250U, 8GB RAM, Windows 11).
    - Clients
        - MiniPC NEO Z83-4 (Intel Atom x5-Z8350, 4GB RAM, Windows 10).
        - Laptop (Intel Core i5-1035G1, 8GB RAM, Windows 11).
        - Two Raspberry Pi 4 Model B (Quad Core Cortex-A72, 8GB RAM, Ubuntu 22.04 LTS).
2. **Software Stack**
    - OS: Windows 11 (Server), Windows 10, Linux Ubuntu 22.04 (Clients).
    - ML Framework: PyTorch 1.8.0 with Keras API.
    - Programming Language: Python 3.8.10.
    - Monitoring Tool: A real-time thread continuously measures CPU, RAM, and bandwidth during training.
3. **Experimental Setup**
    - Dataset: Chest X-Ray Images (5,863 images).
    - Model: 6-layer CNN (420,000 parameters).
    - Training Adaptation:
        - If CPU > 80
        - If Bandwidth < 2Mbps → Reduce model update frequency.
        - If CPU < 80
4. **Findings**
    - BACA successfully prevented CPU overload by dynamically adjusting the training.
    - Bandwidth-aware adaptation improved model synchronization, reducing delays.
    - Devices with limited resources contributed efficiently without being overwhelmed.

3.3.3 AdaptiveMesh: Adaptive FL for Resource-Constrained Environments

Objective AdaptiveMesh introduced a flexible federated learning framework that dynamically adjusts training parameters (images, epochs, batch size, model updates) based on real-time device performance metrics.

System Architecture

- **Adaptive FL Server:** Tracks client resource usage and dynamically optimizes workload distribution
- **Clients:** A set of heterogeneous devices running different OS and hardware configurations.
- **Wireless Network:** Connected via Wi-Fi with real-time CPU/bandwidth monitoring.

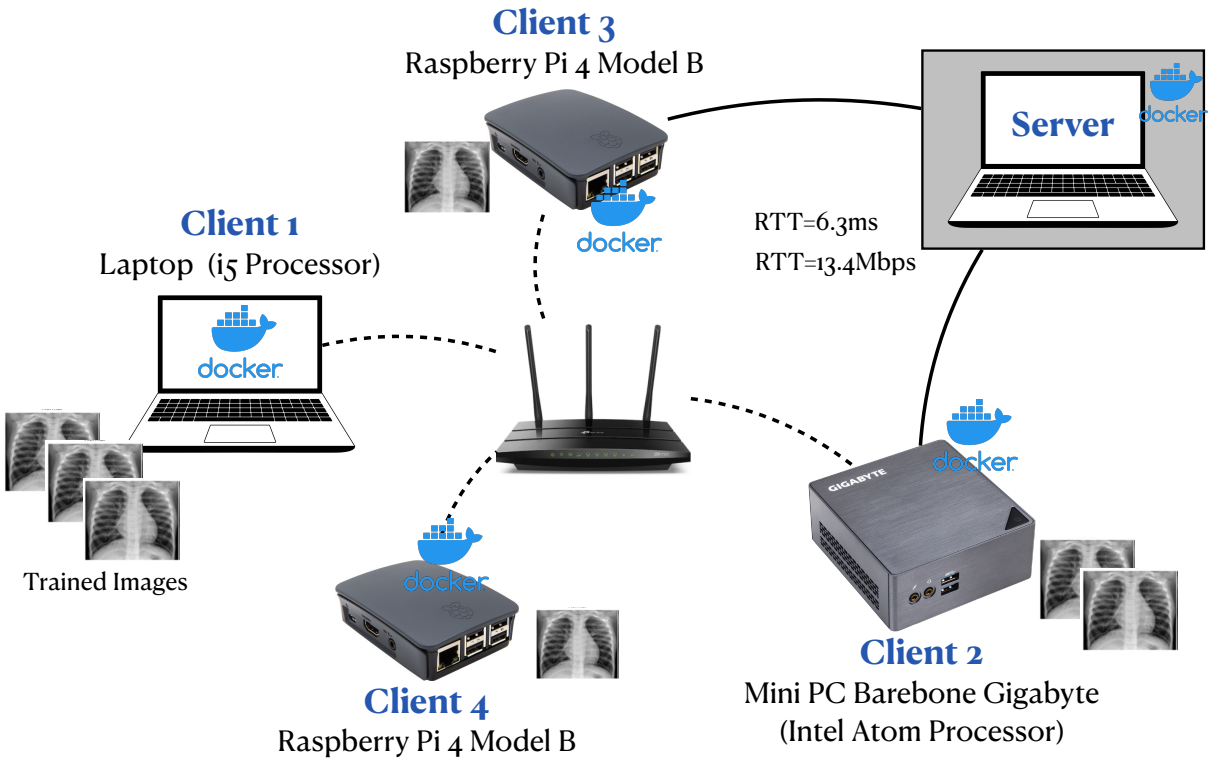

Figure 3.5: Setup nodes utilized in the experimentation (AdaptiveMesh).

1. **Hardware Configuration** Same as the BACA setup, with additional performance log for client-device comparisons.

2. **Software**

    - OS: Windows/Linux mix for diverse real-world simulations.
    - ML Framework: PyTorch 1.8.0 + TensorFlow for adaptive scheduling.
    - Programming Language: Python 3.8.10.
    - Statistical Analysis: ANOVA and regression models to analyze adaptive behavior.

3. **Experimental Setup**

    - **Training adjustments based on real-time monitoring:**
        - If CPU load > 80% → Reduce resources.
        - If CPU load < 80% → Increase resources.
        - If CPU temperature > 85% → Reduce resources.
        - If CPU temperature < 85% → Increase resources.
        - If bandwidth < 2 Mbps → Reduce communication frequency.
        - If training time > 30 minutes → Optimize resources.

4. **Findings**

    - AdaptiveMesh improved FL stability, reducing computational overhead while maintaining high accuracy.
    - Statistical analysis confirmed significant improvements over non-adaptive FL.
    - This study laid the foundation for further fine-grained adaptive control in FL.

3.3.4 Enhancing Adaptive Behavior in FL

Objective

This study performed a comparative analysis between FL performance on:

1. Physical devices (Raspberry Pi, MiniPC, Laptop).
2. Virtualized Cloud Instances (Google Cloud VMs).

The goal was to understand the trade-offs between accuracy, training time, and resource consumption.

System Architecture

- FL Server: Hosted in both physical and virtualized cloud environments.
- FL Clients: A combination of real and virtual devices.
- Network: aggregation of federated cloud models.

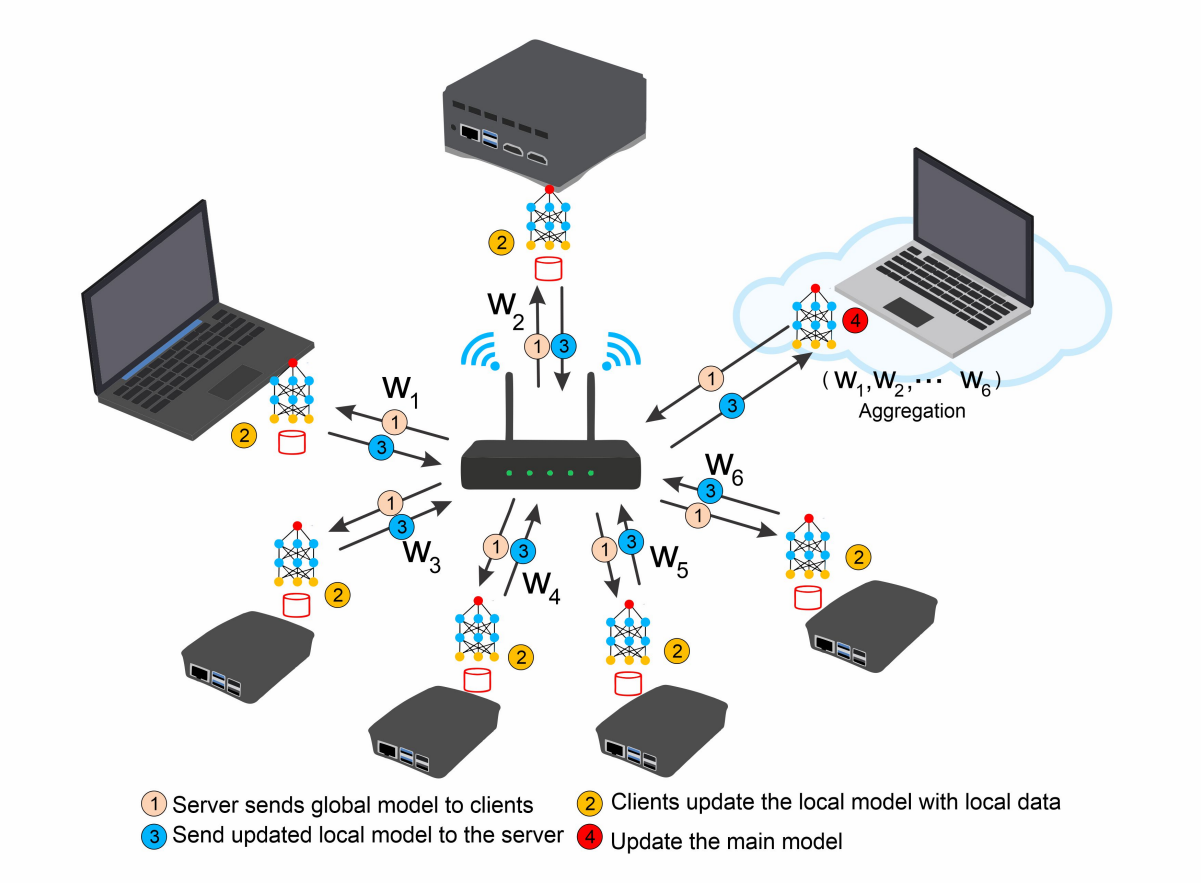

Figure 3.6: Setup nodes utilized in the experimentation (AdaptiveMesh - 6 clients and 1 server).

1. **Hardware Configuration**

    - Physical devices: Same as previous setups, with more Raspberry Pi devices as clients and additional performance logging for client-device comparisons.
    - Cloud VMs: Instances with varying CPU, RAM, and storage capacities.

2. **Software**

    - OS: Windows, Linux.
    - ML Framework: PyTorch + Keras API.
    - Cloud Environment: Google Cloud VMs with FL deployment.

3. **Experimental Setup**

    - Dataset: Chest X-Ray Images.
    - Comparison Metrics
        - Training speed on physical vs. virtual clients.
        - Energy consumption - resource allocation.
        - Accuracy trade-offs based on computation power.

4. **Findings**

    - Cloud-based FL models trained faster but required more communication overhead.
    - Physical devices had lower latency but required careful adaptation.
    - Hybrid approaches combining edge and cloud FL showed promising results.

3.4 Data Set

For our experimental analysis, we used a dataset comprising 5,863 chest radiographic images stored in JPEG format and divided into two categories: Pneumonia and normal. These scans were obtained from young pediatric patients, aged one to five years, at Guangzhou Women's and Children's Medical Center [91].

The sensitive nature of medical data raises serious privacy concerns, making FL a promising solution, as it facilitates collaborative training of machine learning models without exposing raw patient information. This research uses chest radiographs to assess the efficiency of AdaptiveMesh in healthcare settings, particularly for IoT-enabled and mobile health monitoring applications, which often operate under limitations with respect to computational capacity and energy usage.

Training deep learning models in high-resolution medical images requires significant computational resources, and ensuring precise classification, such as pneumonia detection, is crucial to improving patient outcomes. The study findings emphasize that AdaptiveMesh efficiently handles resource allocation while maintaining model accuracy, which makes it particularly effective for low-power edge devices in medical imaging applications.

3.5 Methodology

This study used a structured methodology to develop and evaluate adaptive FL algorithms in resource-constrained environments. The approach integrated insights from previous work on decentralized learning, resource optimization, and privacy preservation, while introducing novel mechanisms for dynamic workload adaptation. The methodology was organized into four interconnected phases:

- System Modeling and Architecture The FL framework was designed using Python and PyTorch for model training and aggregation, with Docker containers deployed to ensure consistency across heterogeneous devices (Raspberry Pi 4, MiniPCs, laptops).

- Development of Adaptive Algorithms Two adaptive algorithms were designed to address device heterogeneity and dynamic network conditions: BACA (bandwidth and CPU-Aware FL), which dynamically adjusts training parameters based on real-time CPU utilization and bandwidth availability, and AdaptiveMesh extended BACA by integrating temperature monitoring, training time, and context switching logic.

- Experimental setup and data set experiments were conducted in physical and virtual environments. Physical devices: Raspberry Pi 4 (ARM Cortex-A72), MiniPC (Intel Atom), and laptops (Intel Core i5) trained a 6-layer CNN for pneumonia detection using 5,863 chest X-ray images. Virtual Environments: Google Cloud instances (N2, E2, and Tau T2A series) mirrored physical hardware configurations to evaluate scalability.

- Evaluation and analysis. Performance was assessed using quantitative metrics and statistical methods. Metrics: CPU utilization, temperature, RAM consumption, training time, bandwidth usage, and model accuracy. Statistical analysis ANOVA tested variations in CPU, accuracy, and epochs across devices. Regression models analyzed relationships between resource metrics and training outcomes.

- Comparative Evaluation: AdaptiveMesh was compared with NAIVE FL (non-adaptive), FedAda, and FedALA to highlight advantages in resource efficiency and thermal management.

This methodology ensured the rigorous validation of adaptive FL mechanisms, balancing computational efficiency with model accuracy in real-world scenarios. By integrating Docker

for reproducibility, PyTorch for scalability, and MySQL for centralized logging, the framework provided a robust foundation for deploying FL in privacy-sensitive applications like healthcare IoT.

Chapter 4

Adaptive Federated Learning and Algorithms

4.1 Introduction

FL is a distributed machine learning method that allows multiple devices to jointly train a shared model while preserving data privacy. In contrast to traditional centralized learning, which requires all data to be sent to a central server for processing, FL keeps the data locally on individual devices, sharing only model updates for aggregation. This technique is particularly beneficial for applications that require the preservation of privacy, such as mobile devices, IoT networks, and distributed sensor systems.

However, implementing FL in diverse and resource-limited environments presents several challenges. These include:

- Computational constraints: Many edge devices, including IoT sensors and embedded systems, have limited processing power and memory, slowing down training operations.
- Network limitations: Frequent exchanges between devices and the central server can strain network resources, particularly in mobile or wireless environments, where bandwidth is often inconsistent.
- Heterogeneous client capabilities: Devices in an FL network can vary in computational power, network speed, and availability due to factors such as battery life and operating conditions.

- Non-uniform data distribution: In contrast to traditional machine learning methods, which assume data is independent and identically distributed (IID), FL clients frequently possess non-IID data, indicating that their datasets can differ substantially in both content and size.

- Energy Efficiency Concerns: Since FL relies on distributed computing across multiple devices, energy consumption becomes a crucial factor, particularly for battery-operated IoT devices.

- Scalability limitations: As the number of participating clients increases, communication overhead and resource allocation complexities grow, making it challenging to maintain optimal performance.

Adaptive Federated Learning is a smart way to use computers to learn and improve from data, but it comes with some challenges. Let us discuss these problems and some ways to solve them. The problem we are addressing is how to make federated learning more adaptable. In federated learning, a central server works with a group of clients, but these clients can be quite different from each other. They may have varying Internet speeds, use different devices, and have different levels of computing power.

A significant limitation of conventional FL implementations is that they typically use a uniform training approach across all clients, disregarding the unique capabilities and constraints of each device. Adaptive Federated Learning (AFL) addresses this issue by introducing mechanisms that dynamically adjust training workloads based on real-time device performance and network conditions. Instead of applying a one-size-fits-all model, AFL customizes training parameters such as batch size, number of epochs, and data allocation per client, thereby optimizing learning efficiency while ensuring system stability.

In this setup, the central server is like the conductor of a symphony, and the clients are the musicians. After each ”round” of training, the server combines what each client has learned to create a new ”group” lesson. Then, this new lesson is sent back to the clients along with some instructions for the next round of learning.

What we are suggesting here is that the server should be more flexible. Instead of always using the same instructions for all clients, it should customize the instructions for each client, depending on how they’re doing. So, it is not just about sending and receiving lessons; it’s also about adjusting how each client learns. To make these adjustments, the server checks how

things are going after each round of training. An important thing to look at is how much work each client can handle, which is different for each client. This measure includes how many lessons they've completed, how much time it took to send and receive lessons, and how much time they spent actually learning the material.

This approach can be likened to an orchestra, where the central server acts as the conductor, guiding the learning process, while clients function as musicians, each contributing uniquely based on their capabilities. After each training round, the server aggregates the updates from clients and fine-tunes the model before redistributing it. The key distinction in AFL is that the server does not just blindly distribute uniform instructions, but tailors them to the specific conditions of each client.

In addition, AFL can integrate techniques such as gradient compression to reduce communication costs, asynchronous updates to improve training speed, and priority-based scheduling to ensure that critical devices receive sufficient computational resources. These enhancements make AFL more adaptable and efficient in real-world scenarios.

In the first phase of this research, a secure and privacy-preserving FL system was implemented on the edge of IoT networks using devices such as Raspberry Pi, MiniPCs, and laptops. Initial experiments measured model accuracy, CPU usage, RAM consumption, and network performance. These experiments highlighted the need for optimized strategies tailored to low-power devices, leading to the development of adaptive algorithms designed to dynamically adjust training parameters based on the resource availability of each device.

In this chapter, we examine two state-of-the-art AFL algorithms: AdaptiveMesh and BACA. These algorithms improve FL by dynamically adjusting training strategies, optimizing model convergence, and efficiently managing computational and network resources.

4.2 BACA Algorithm

The **BACA: Bandwidth and CPU-Aware Adaptive FL** algorithm, outlined in Algorithm 2, begins by defining key training parameters such as global and local learning rates, total epochs, number of input images, batch size, and other related settings. The process follows these steps:

1. **Initialization**: Clients are enrolled to participate in the collaborative model training process.

2. **Training Setup**: The central server begins the training by assigning initial configurations to heterogeneous clients (with $min\_epochs = 1$). A predefined time constraint is applied to each round to ensure that the training duration stays within limits.

3. **Monitoring**: Throughout each round, system metrics such as processor usage (CPU), memory consumption, bandwidth, training duration, and performance accuracy are tracked and recorded for further assessment.

4. **Adaptive Mechanism Activation**: After completion of three training iterations, an adaptive mechanism is activated to analyze whether any adjustments are necessary to the training parameters.

5. **Adaptation Process**: The adaptive component analyzes the variation in CPU usage in different rounds, evaluates model performance trends, and monitors CPU usage patterns to decide whether adaptation is necessary. It also considers factors such as bandwidth consumption and training duration. If the training period exceeds the predefined limit or bandwidth consumption exceeds the threshold, the algorithm dynamically adjusts the number of epochs and training images to improve resource efficiency. The minimum training image count is set to 6 ($min\_images = 6$).

   (a) **Case 1: Resources Available**: f adaptation is needed and there are sufficient computational resources (e.g., CPU usage remains below 80%), the number of training images increases, with a cap of 234 images.

   (b) **Case 2: Resources Unavailable**: If adaptation is required, but computational resources are limited (for example, CPU usage exceeds 80%), the number of training images is decreased for affected clients.

The adaptive decision-making approach of the BACA algorithm plays a vital role in enhancing model performance across diverse clients operating in wireless mesh environments. This adaptive mechanism ensures that resource allocation remains flexible and responsive.

Additionally, by structuring the configuration of parameters for FL, the algorithm provides a well-thought-out approach to managing collaborative training, which is crucial for advancing distributed machine learning in real-world applications.

Algorithm 1: BACA: Bandwidth and CPU-Aware Adaptive Federated Learning Algorithm for Wireless Environments

Require: round, cpu_limit, min_epochs, min_images, max_images, accuracies[], cpu_history[], client_images[], client_epochs[], training_time[], bandwidth_history[], max_training_time, max_bandwidth, no_images_by_client, no_epochs_by_client

Result : new_epochs, new_images

$i \leftarrow$ round;

if $i > 3$ **then**

  delta_cpu1 $\leftarrow$ cpu_history$[i-1]$ − cpu_history$[i-2]$;

  delta_cpu2 $\leftarrow$ cpu_history$[i-2]$ − cpu_history$[i-3]$;

  diff $\leftarrow$ delta_cpu2 − delta_cpu1 ;

  if $(training\_time[-1] \geq max\_training\_time$ ***OR*** $bandwidth\_history[-1] > max\_bandwidth)$ **then**

    new_images $\leftarrow$ max(client_images$[-1]$ − no_images_by_client, min_images) ;

    new_epochs $\leftarrow$ max(client_epochs$[-1]$ − no_epochs_by_client, min_epochs) ;

  else if $(accuracies[-1] > accuracies[-2])$ ***AND*** $(accuracies[-1] < 100)$ **then**

    if $(diff \leq 0)$ ***AND*** $(cpu\_history[i-1] \geq cpu\_limit)$ **then**

      new_images $\leftarrow$ max(client_images$[-1]$ − no_images_by_client, min_images) ;

      new_epochs $\leftarrow$ max(client_epochs$[-1]$ − no_epochs_by_client, min_epochs) ;

    else

      new_images $\leftarrow$ min(client_images$[-1]$ + no_images_by_client, max_images) ;

      new_epochs $\leftarrow$ client_epochs$[-1]$ + no_epochs_by_client ;

  else

    if $(cpu\_history[i-1] \leq cpu\_limit)$ **then**

      new_images $\leftarrow$ min(client_images$[-1]$ + no_images_by_client, max_images) ;

      new_epochs $\leftarrow$ client_epochs$[-1]$ + no_epochs_by_client ;

    else

      new_images $\leftarrow$ max(client_images$[-1]$ − no_images_by_client, min_images) ;

      new_epochs $\leftarrow$ max(client_epochs$[-1]$ − no_epochs_by_client, min_epochs) ;

fl_config.epochs $\leftarrow$ new_epochs ;

fl_config.training_images $\leftarrow$ new_images;

4.3 AdaptiveMesh: An Adaptive Federated Learning Algorithm for Wireless Mesh Networks

The **AdaptiveMesh** algorithm dynamically adjusts to the device conditions in real time by altering essential training configurations, such as the total number of epochs and training images, to prevent performance deterioration. This flexibility offers a notable improvement over static FL algorithms, which often struggle to efficiently utilize system resources.

Initially, AdaptiveMesh is set up with key training parameters, including global and local learning rates, epoch count, image quantity, batch size, and other relevant settings. The algorithm then proceeds with the following steps:

1. **Initialization**: Clients are registered to participate in the collaborative training process.
2. **Training Setup**: The central server begins the training by distributing initial configurations to heterogeneous clients.
3. **Monitoring**: During each training round, a background thread is launched to monitor metrics such as CPU utilization, CPU temperature, memory usage, bandwidth, training time, and accuracy. These metrics are logged for further evaluation.
4. **Adaptive Mechanism Activation**: After completion of three training iterations, an adaptive mechanism is activated to analyze whether any adjustments are necessary to the training parameters.
5. **Adaptation Process**:The adaptive component evaluates whether predefined thresholds for training time, bandwidth, CPU temperature, and fluctuations in CPU usage between training rounds have been exceeded. If any of these thresholds are breached, the algorithm dynamically modifies the number of epochs and training images to optimize resource utilization.
    - **Case 1: Resources Available**: If adaptation is needed and sufficient computational resources are present (CPU usage remains below 80%, the CPU temperature stays below 85 ° C and bandwidth usage is below 2 Mbps), the number of training images is increased, up to a maximum of 112 images, and the number of epochs and batch sizes is optimized to enhance model performance without overloading the system.

- **Case 2: Resources Unavailable**: If adaptation is required but computational resources are constrained (e.g., CPU usage exceeds 80%, CPU temperature rises above 85°C, or bandwidth consumption exceeds 2 Mbps), the number of training images is reduced and the epochs and batch sizes are adjusted to ensure efficient resource utilization while maintaining acceptable model accuracy.

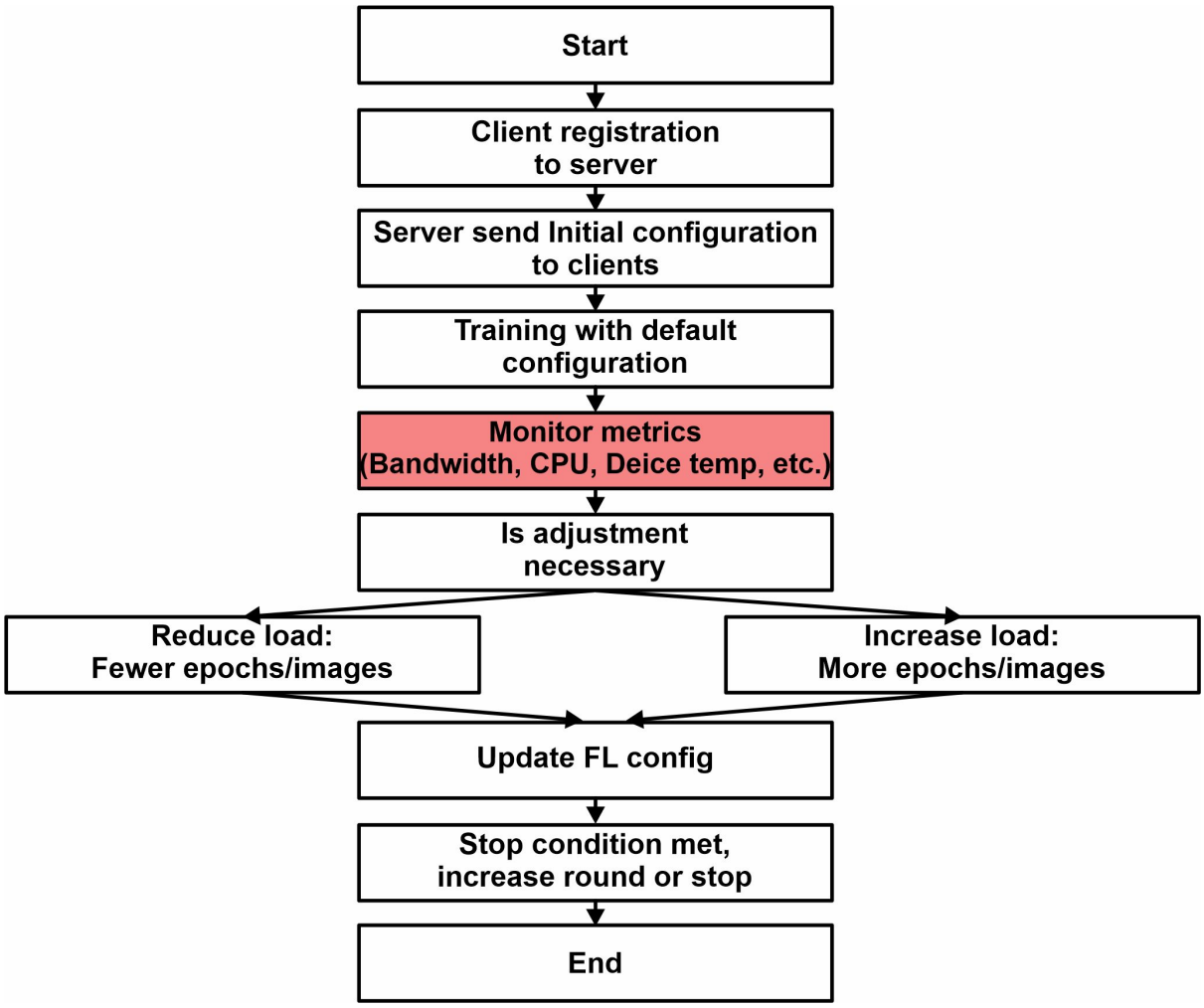

Figure 4.1: AdaptiveMesh's basic flowchart.

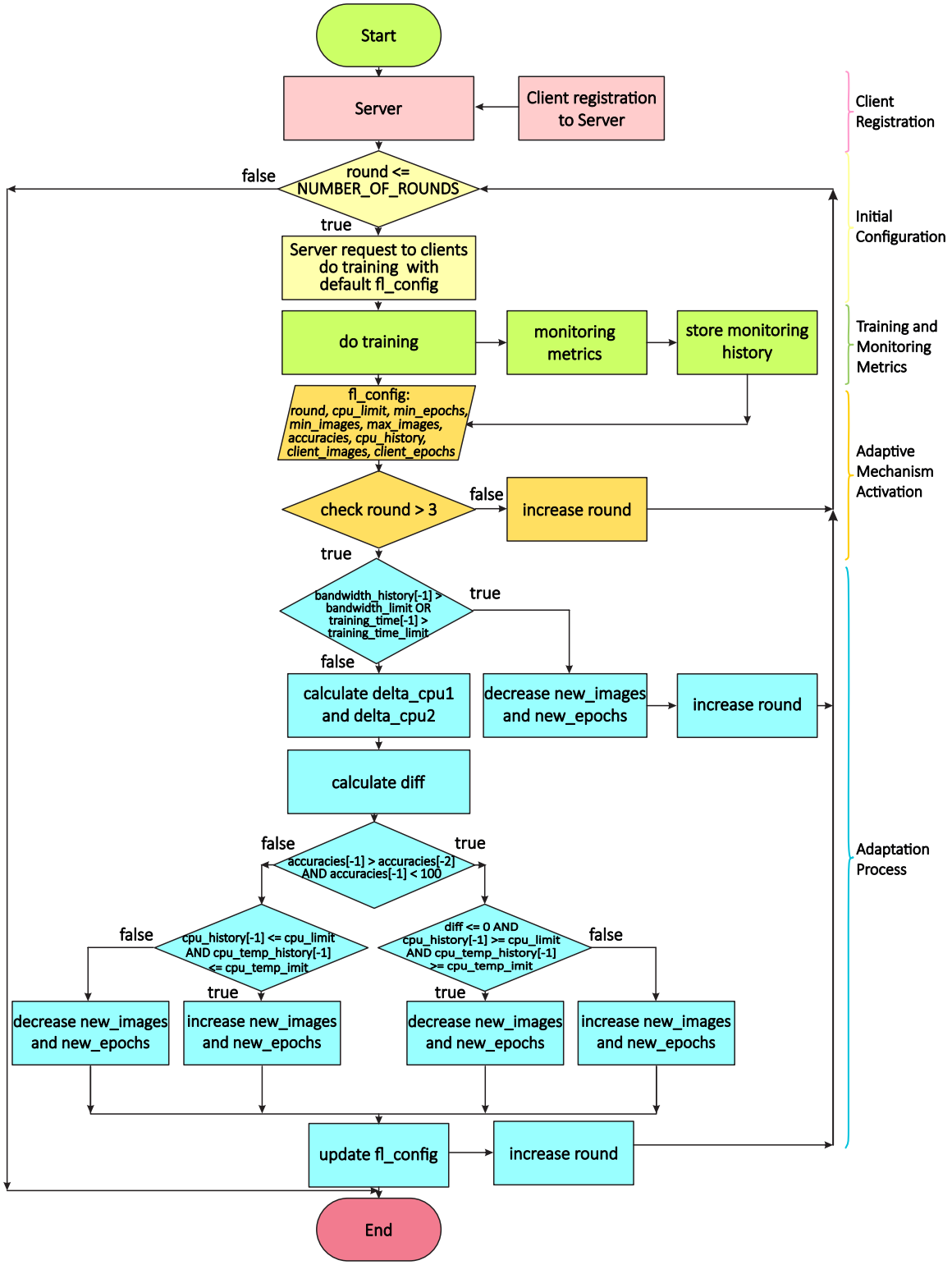

Figure 4.2: AdaptiveMesh's flowchart algorithm.

Algorithm 2: AdaptiveMesh: Client-side Adaptive Federated Learning for Resource-Constrained Wireless Environments

Require: round, cpu_limit, temp_cpu_limit, training_time_limit, bandwidth_limit, min_epochs, min_images, max_images, accuracies[], cpu_history[], temp_cpu_history[], bandwidth_history[], training_time[], client_images[], client_epochs[], ΔE, ΔI

Result : new_epochs, new_images

$i \leftarrow$ round;
previous_cpu $\leftarrow$ cpu_history$[i-1]$;
previous_temp_cpu $\leftarrow$ temp_cpu_history$[i-1]$;
previous_training_time $\leftarrow$ training_time$[i-1]$;
previous_bandwidth $\leftarrow$ bandwidth_history$[i-1]$;
previous_images $\leftarrow$ client_images$[i-1]$;
previous_epochs $\leftarrow$ client_epochs$[i-1]$;
if $previous\_training\_time \geq training\_time\_limit$ ***OR*** $previous\_bandwidth > bandwidth\_limit$ **then**
    new_images $\leftarrow$ max(previous_images $- \Delta I,$ min_images);
    new_epochs $\leftarrow$ max(previous_epochs $- 1,$ min_epochs);
else if $(accuracies[-1] > accuracies[-2])$ ***AND*** $(accuracies[-1] < 100)$ **then**
    if $previous\_cpu \geq cpu\_limit$ ***OR*** $previous\_temp\_cpu \geq temp\_cpu\_limit$ **then**
        new_images $\leftarrow$ max(previous_images $- \Delta I,$ min_images);
        new_epochs $\leftarrow$ max(previous_epochs $- \Delta E,$ min_epochs);
    else
        new_images $\leftarrow$ min(previous_images $+ \Delta I,$ max_images);
        new_epochs $\leftarrow$ previous_epochs $+ \Delta E$;
else
    if $previous\_cpu \leq cpu\_limit$ ***AND*** $previous\_temp\_cpu \leq temp\_cpu\_limit$ **then**
        new_images $\leftarrow$ min(previous_images $+ \Delta I,$ max_images);
        new_epochs $\leftarrow$ previous_epochs $+ \Delta E$;
    else
        new_images $\leftarrow$ max(previous_images $- \Delta I,$ min_images);
        new_epochs $\leftarrow$ max(previous_epochs $- \Delta E,$ min_epochs);
fl_config.epochs $\leftarrow$ new_epochs;
fl_config.training_images $\leftarrow$ new_images;

4.3.1 CPU Load Calculation

At any given training round t, the CPU usage history of a device is maintained for the duration of the round. The average CPU usage $\bar{U}_i$ for device i is calculated as:

$$\bar{U}_i = \frac{1}{T} \sum_{j=1}^{T} U_{ij} \tag{4.1}$$

where:

- T signifies the total count of samples processed within the training round.
- U_{ij} is the CPU usage of device i at sample j.

If the average CPU usage exceeds a predefined threshold (80%):

$$\bar{U}_i > Threshold. \tag{4.2}$$

4.3.2 Temperature Load Calculation

At any given training round t, the temperature history of a device is maintained for the duration of the round. The average temperature $\bar{T}_i$ for device i is calculated as:

$$\bar{T}_i = \frac{1}{T} \sum_{j=1}^{T} T_{ij} \tag{4.3}$$

where:

- T signifies the total count of samples processed within the training round.
- T_{ij} is the temperature of device i at sample j.

If the average temperature exceeds a predefined threshold (85°C):

$$\bar{T}_i > Threshold. \tag{4.4}$$

4.3.3 Bandwidth Load Calculation

At any given training round t, the bandwidth usage history of a device is maintained for the duration of the round. The average bandwidth usage $\bar{B}_i$ for device i is calculated as:

$$\bar{B}_i = \frac{1}{T} \sum_{j=1}^{T} B_{ij} \tag{4.5}$$

where:

- T signifies the total count of samples processed within the training round.
- B_{ij} is the bandwidth usage of device i at sample j.

If the average bandwidth usage exceeds a predefined threshold (2 Mbps):

$$\bar{B}_i > Threshold. \tag{4.6}$$

4.3.4 Context-Switching Trigger

If the average CPU usage $\bar{U}_i$, temperature $\bar{T}_i$, or bandwidth usage $\bar{B}_i$ exceeds their respective thresholds, the system reduces both the local training epochs E_i and the number of processed images I_i to balance the computational load:

$$E' = \begin{cases} E_i - \Delta E & \text{if } \bar{U}_i > Threshold \text{ or } \bar{T}_i > Threshold \text{ or } \bar{B}_i > Threshold, \\ E_i + \Delta E & \text{otherwise.} \end{cases} \tag{4.7}$$

$$I' = \begin{cases} I_i - \Delta I & \text{if } \bar{U}_i > Threshold \text{ or } \bar{T}_i > Threshold \text{ or } \bar{B}_i > Threshold, \\ I_i + \Delta I & \text{otherwise.} \end{cases} \tag{4.8}$$

where:

- ΔE represents a fixed reduction in local training epochs.
- ΔI represents a fixed reduction in the number of processed images.

By defining these thresholds mathematically, the model enables devices to dynamically adjust key parameters, ensuring consistent performance under different conditions. When these thresholds are exceeded, AdaptiveMesh automatically decreases the training duration

and the number of images processed to maintain stability throughout the distributed learning process.

The decision-making framework within AdaptiveMesh plays a key role in enhancing the efficiency of the model among diverse clients in wireless mesh networks. This flexible mechanism guarantees that resource distribution is continuously optimized to accommodate the changing needs of training models. Furthermore, the system configuration for federated learning reflects a well-structured strategy for synchronizing collaborative learning, which is essential for advancing distributed machine learning in real-world scenarios.

4.4 Comparison of BACA and AdaptiveMesh Algorithms

In this section, we compare the two main Federated Learning (FL) algorithms proposed in this thesis: **BACA** (Bandwidth and CPU-Aware Adaptive Federated Learning) and its evolved version, **AdaptiveMesh**. Although BACA initially focused on optimizing CPU and bandwidth usage, AdaptiveMesh extends this by incorporating temperature monitoring and additional adaptive mechanisms. In the following, we provide a detailed comparison of their features, strengths, and weaknesses.

4.4.1 Overview of BACA and AdaptiveMesh

BACA is an adaptive FL algorithm designed to optimize CPU and bandwidth usage in wireless environments. Dynamically adjusts training loads based on real-time resource metrics, ensuring efficient model training while preventing device overload.

AdaptiveMesh is an enhanced version of BACA that introduces temperature monitoring and more sophisticated adaptive mechanisms. Dynamically adjusts training parameters (e.g. epochs, training images, and batch sizes) based on CPU usage, temperature, and bandwidth, making it suitable for environments where devices are prone to overheating or resource exhaustion.

4.4.2 Comparison Table

The following table provides a detailed comparison of the key characteristics of BACA and AdaptiveMesh:

4.4.3 Key Differences

- **BACA** focuses on optimizing CPU and bandwidth, making it suitable for environments with unstable network conditions. However, it does not monitor the temperature, which limits its use in devices prone to overheating.
- **AdaptiveMesh** extends BACA by incorporating temperature monitoring and more sophisticated adaptive mechanisms. This makes it ideal for devices like the Raspberry Pi and MiniPC, which are sensitive to overheating.
- **AdaptiveMesh** may sacrifice some model accuracy to prevent overloading and overheating, while **BACA** maintains high accuracy, but can increase computational load on devices.

Table 4.1: Comparison of BACA and AdaptiveMesh

Feature	**BACA**	**AdaptiveMesh**
Focus	Optimizes CPU and bandwidth	Optimizes CPU, temperature, bandwidth and training time
Load Adaptation	Based on CPU and bandwidth	Based on CPU, temperature, and bandwidth
Temperature Management	No	Yes
Bandwidth Management	Yes	Yes
Resource Usage	Adjusts update frequency and training load	Dynamically adjusts epochs, batch sizes, and training images
Monitoring Parameters	CPU, bandwidth	CPU, temperature, memory, bandwidth, training time
Adaptive Mechanisms	Yes	Yes
Model Accuracy	Maintains high accuracy but may increase device load	May decrease to prevent overloading and overheating
Training Time	Reduced due to bandwidth optimization	Reduced due to adaptive load balancing

4.4.4 Conclusion

BACA and **AdaptiveMesh** represent an evolution in adaptive FL algorithms. BACA provides a robust solution for optimizing CPU and bandwidth in wireless environments, while AdaptiveMesh improves this by adding temperature monitoring and more dynamic load adaptation.

Part III. Experimental Data Analysis and Results

Chapter 5

Experimental Data Analysis

To assess the effectiveness of FL algorithms, we performed experiments in heterogeneous environments, involving both physical and virtual devices. The experiments used a Raspberry Pi 4, a Mini PC NEO Z83-4 and a laptop as clients, while the server was deployed on a separate laptop. In addition, cloud-based virtual machines were used to compare performance differences.

The training model was a six-layer CNN designed for pneumonia classification using chest radiograph images in patients with pneumonia as highlighted in [92]. We tested multiple FL approaches, including AdaptiveMesh, BACA, FederatedMesh, and a Naive baseline, in different scenarios to assess their impact on resource utilization and model performance.

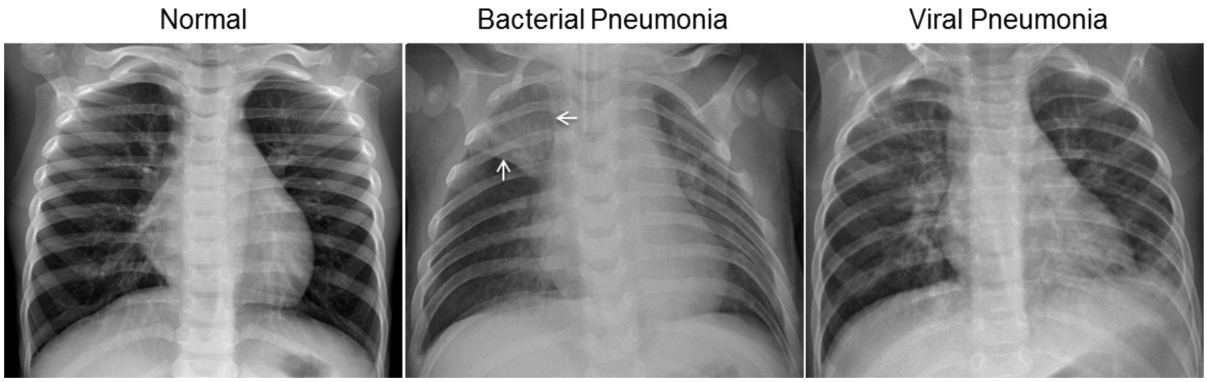

Figure 5.1: Chest X-ray images of people with pneumonia

5.1 Performance Metrics

The following key metrics were evaluated to measure the effectiveness of various FL approaches:

- **CPU Utilization (%):** The load percentage of the processor during training.
- **CPU Temperature (°C):** Temperature variations in response to training workloads.
- **Accuracy (%):** The percentage of correct predictions by the model.
- **Training Time (s):** The total time taken for one training round.
- **Bandwidth Utilization (Mbps):** The network consumption rate during model transmission.

5.2 Experimental Results - FederatedMesh

In the experiments, our aim is to measure how accurate the FL model is when used with different numbers of clients in mesh networks. First, we characterize the network used and also test how much stress the edge devices can handle and measure how much CPU and RAM they use.

Network Characterization Initially, our objective was to evaluate the performance of the existing mesh network. By analyzing bandwidth and round-trip time (RTT) through Empirical Cumulative Distribution Function (ECDF) graphs, we gain insight into data distribution across wireless connections. This helps identify any inconsistencies or connectivity issues that could impact the feasibility of FL in this network.

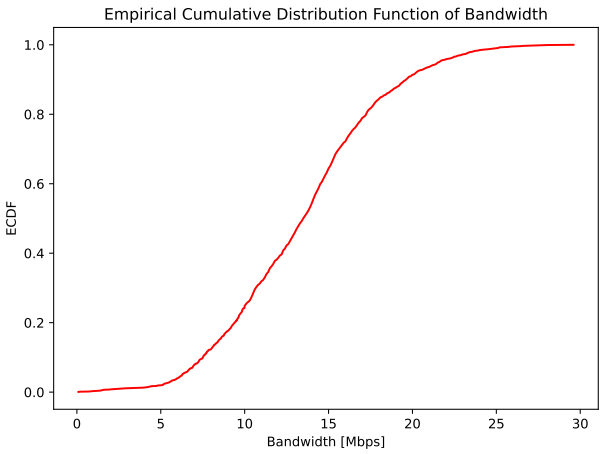

Figure 5.2: ECDF - Network Bandwidth

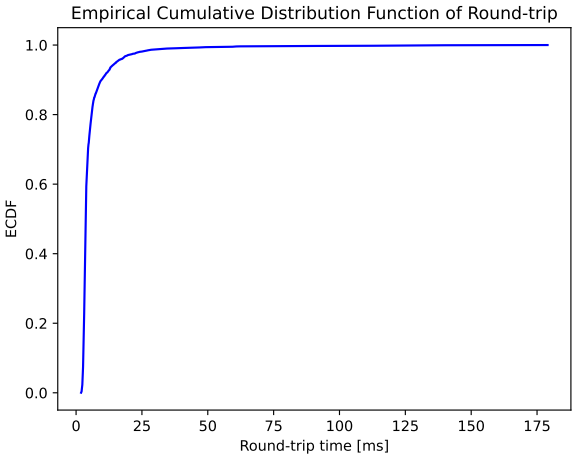

Figure 5.3: ECDF - Network RTT

Figure 5.2 illustrates the ECDF representation of the link bandwidth during FL training. The results indicate that 95% of the observed links exhibit bandwidths between 5 and 30 Mbps. The ECDF curve suggests a positive correlation between increasing bandwidth and cumulative probability, which means that higher bandwidth values are associated with a larger proportion of measured data. These findings highlight significant bandwidth fluctuations, ranging from 5 Mbps to 30 Mbps. Based on this, we conclude that the available network bandwidth is sufficient to transmit model updates, making FL deployment viable in this setting.

Figure 5.3 shows that most RTT values remain low, mainly ranging between 3 ms and 6 ms, indicating that the network operates efficiently with minimal delays. Since FL depends on the transmission of model updates between client devices and the central server, a lower latency enhances these exchanges and ultimately improves the overall performance of the system.

Testing Accuracy: Figure 5.4 shows the impact of varying the participation of the edge device (2 vs. 4 devices) on the accuracy of the test. The results suggest that accuracy generally improves as the number of training cycles (epochs) increases in both configurations.

However, when training is conducted with 4 devices, accuracy remains consistently higher compared to training with only 2 devices, regardless of the number of epochs. In the early training stages (Epochs 1 and 2), the accuracy difference between using 2 and 4 devices is particularly significant. As the number of epochs increases (from 4 to 100), the accuracy gap between the two configurations narrows, although the four-device setup continues to yield superior results.

These observations suggest that increasing the number of participating devices in FL can enhance the accuracy of the model. However, performance improvements may plateau af-

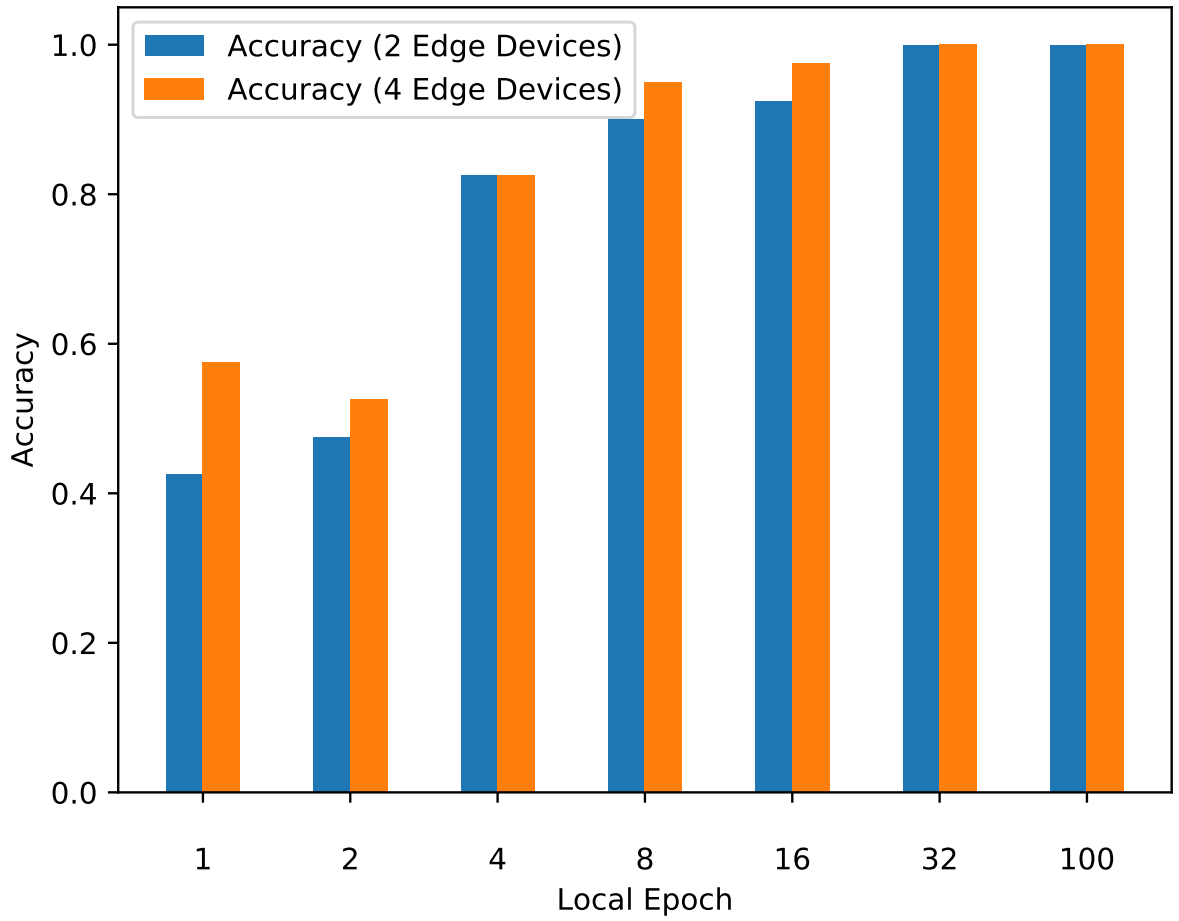

Figure 5.4: Accuracy Evaluation in Relation to Local Epochs (Using 2 and 4 Edge Devices)

ter a certain number of epochs. In addition, factors such as communication limitations and variations in device capabilities can also influence the overall effectiveness of training.

CPU Usage: Figure 5.5 illustrates the average CPU usage during training on different numbers of devices. The analysis reveals that CPU usage fluctuates based on both the number of devices and the number of completed epochs. Furthermore, CPU consumption generally increases as training progresses, but it remains independent of the number of devices.

The highest CPU load is observed during computationally intensive iterations, yet even in such cases, no device exceeds 90% usage.

Overall, it seems that using more devices for training can lead to increased CPU consumption, but this could also reduce the total duration of training. Additionally, peak CPU load occurs while exchanging model updates between client devices and the central server.

Memory Usage: Figure 5.6 presents the pattern of memory utilization during model training in different devices. The findings indicate a slight increase in memory consumption as more devices participate per epoch. However, the difference in memory usage between training

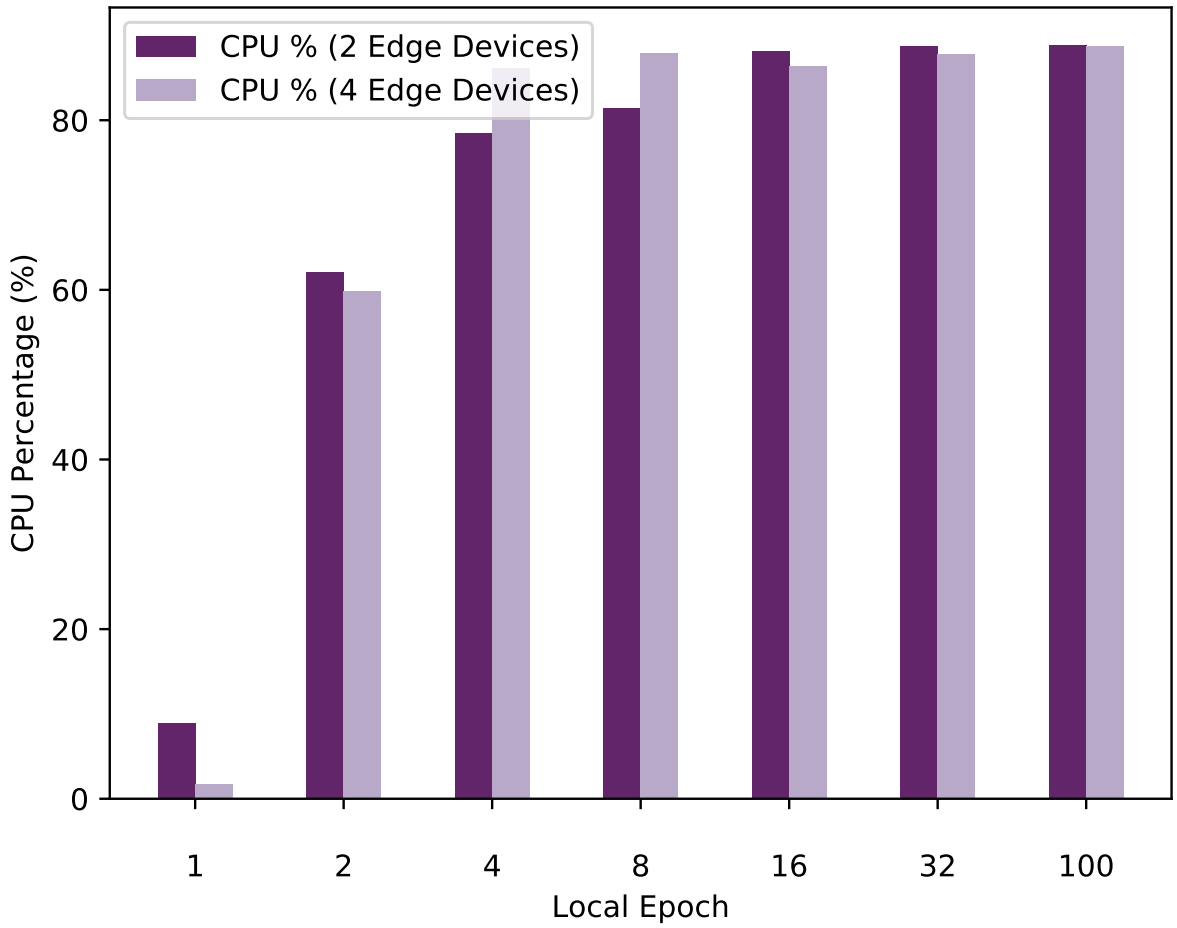

Figure 5.5: CPU Utilization in Relation to Local Epochs (Using 2 and 4 Edge Devices)

with 2 and 4 devices remains relatively small.

The memory capacity of devices used for FL is crucial, as it affects both efficiency and stability during training. In this study, Raspberry Pi devices equipped with 8 GB of RAM effectively handled memory requirements, ensuring consistent memory usage across different epochs and configurations.

The **CPU temperatures** of the four Raspberry Pi devices were monitored during the FL model training. As shown in Figure 5.7, training FL models demands substantial computational power, which causes the CPU to operate at full capacity for extended periods, with an average temperature of 80 degrees. The experimental results suggest that optimizing the training process and reducing the computational load could help mitigate *overheating*. Overheating in Raspberry Pi devices is a critical issue as it may result in system shutdowns and disrupt ongoing experiments or tasks.

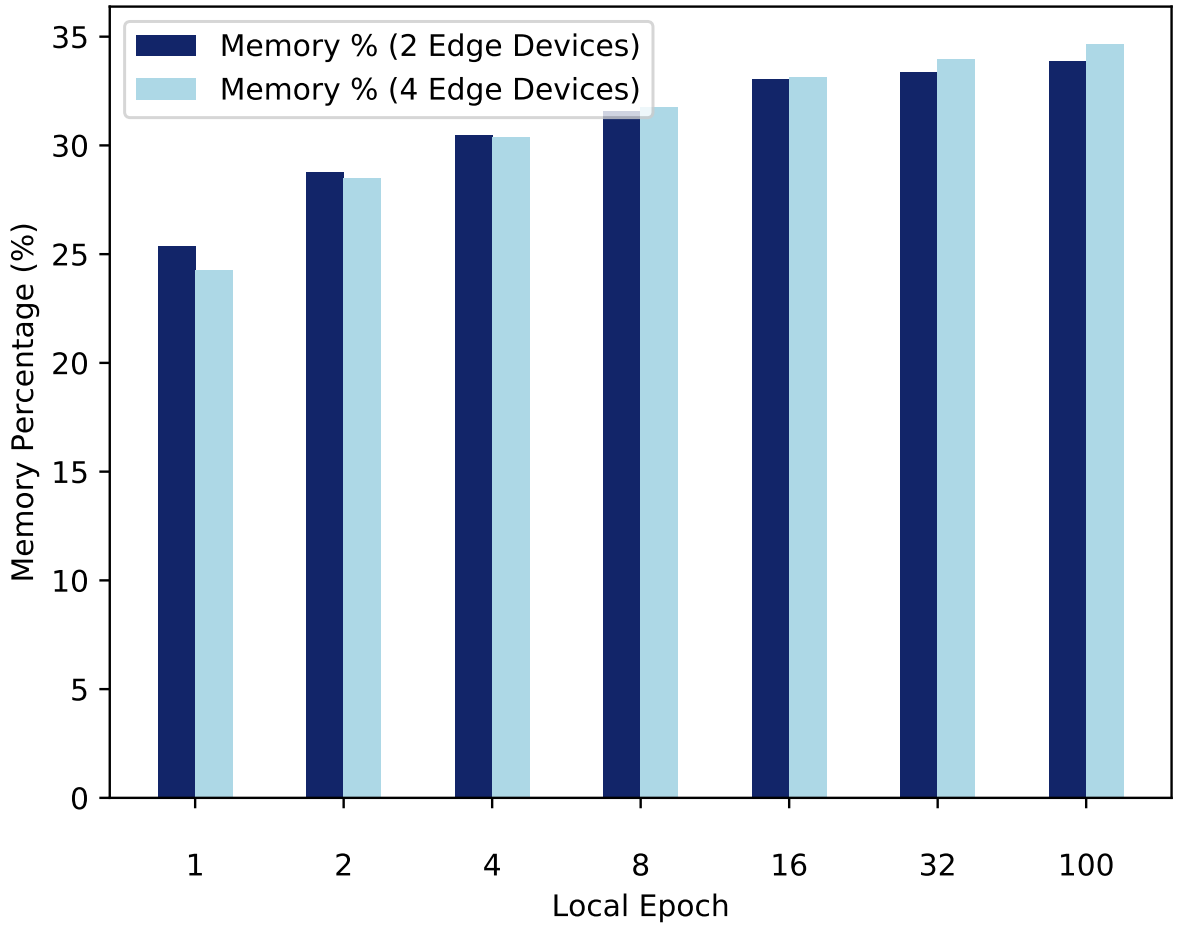

Figure 5.6: Memory (RAM) Usage in Relation to Local Epochs (Using 2 and 4 Edge Devices)

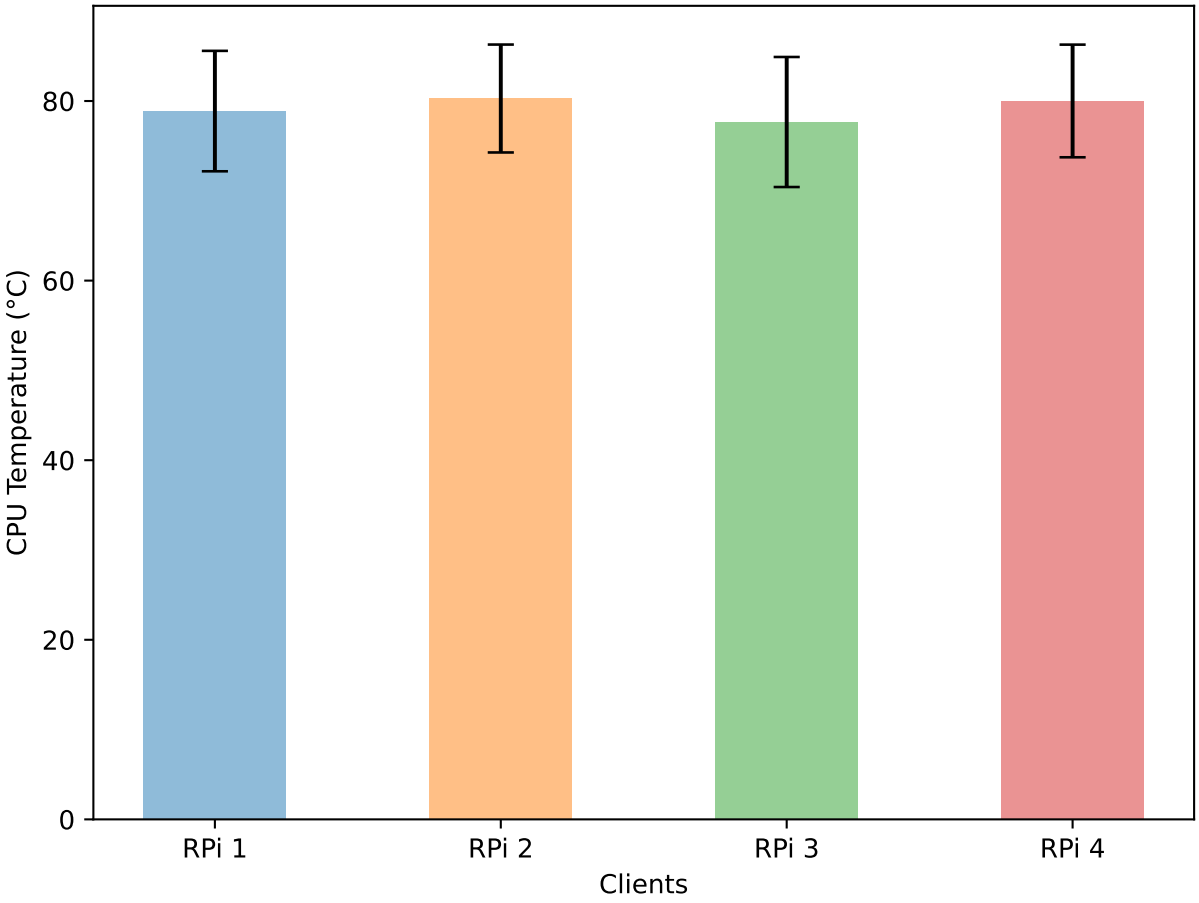

Figure 5.7: CPU Temperature (°C) of Raspberry Pi devices

5.3 Experimental Results - BACA

The goal of these experiments is to evaluate how the adaptive client design influences FL in a practical edge wireless environment. Initially, tests are conducted out to monitor CPU and memory usage, showcasing the system's adaptability. Next, accuracy and loss metrics are examined by adjusting the number of epochs and training images per device to illustrate the adaptive response. Furthermore, the impact of adaptive behavior is investigated in scenarios where clients exceed the maximum training time, experience network bandwidth limitations, or reach CPU capacity constraints.

5.3.1 Results

Figure 5.8 presents the CPU usage in four different clients. The results indicate that model training requires significant computational resources, with CPU utilization ranging between 80% and 90%. However, our algorithm's CPU adaptation optimizes this process. When CPU usage exceeds 80%, the algorithm reduces computational loads in subsequent rounds to prevent client overload (or overheating).

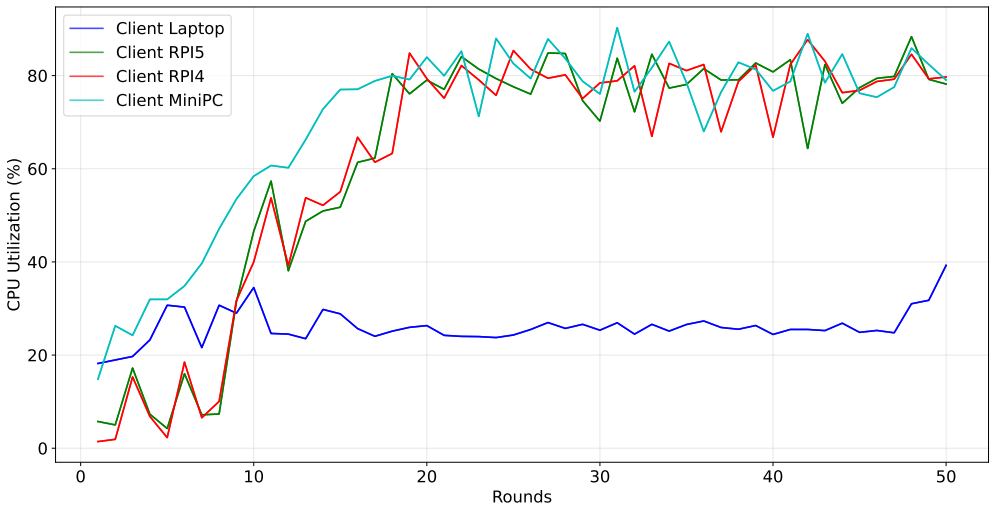

Figure 5.8: CPU Usage Across Various Training Rounds

Figure 5.9 shows the measurements of the network bandwidth up to round 50. In Figure 5.9, it can be seen that the bandwidth generally does not exceed 2 Mbps during communication with the server to transmit the model. However, in round 49, the bandwidth increased due to the initiating a file download to all clients at that time. This was done to test the algo-

rithm and observe the CPU, RAM, accuracy, number of images, and rounds. From the figures (i.e., Figure 5.8), it is evident that there is an increase in CPU usage. In addition, the algorithm decreased the number of images and epochs. In addition, we performed additional bandwidth measurements (ECDF graph, not shown in the paper), and the link bandwidth distribution reveals that most connections (98%) have a bandwidth within the range of 0 to 2 Mbps, along with (2%) connections that have 12-13.4 Mbps.

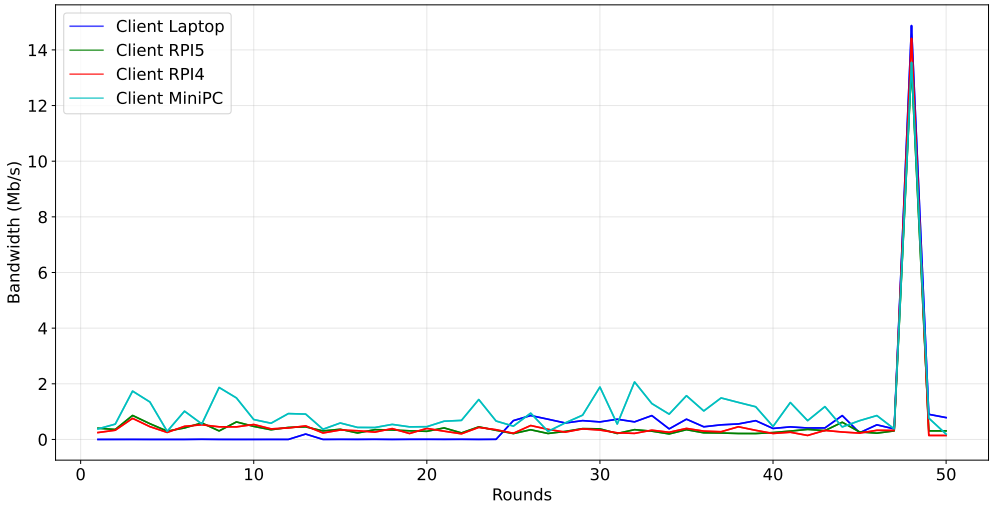

Figure 5.9: Bandwidth in different training rounds.

Figure 5.10 compares the training accuracies achieved in different rounds. Generally, accuracy increases with rounds until round 50, with the Laptop achieving 99%, MiniPC 78%, RPI4 85%, and RPI5 79%, respectively. Further, Figure 5.11 reveals a gradual decrease in losses over rounds. This indicates improvement in model training over time.

Regarding the trained images per client, Figure 5.12 shows a gradual increase with rounds. After round 18, lower performance clients fail to manage workload, leading to a decrease in resource availability until CPU usage is below 80%. When utilization falls below this threshold, they request an increase in the number of images and epochs. In contrast, the Laptop, benefiting from superior performance, continuously requests higher values for images and epochs.

Figure 5.13 shows the number of epochs performed by different clients in each round. Typically, epochs increase with rounds, but for clients who reach CPU thresholds (such as RPI4, RPI5, and MiniPC), epochs decrease due to adaptive behavior. Laptop with more CPU resources generally run more epochs during training. In round 50, the Laptop completes 44 epochs, MiniPC 16 epochs, RPI4 14 epochs and RPI5 20 epochs.

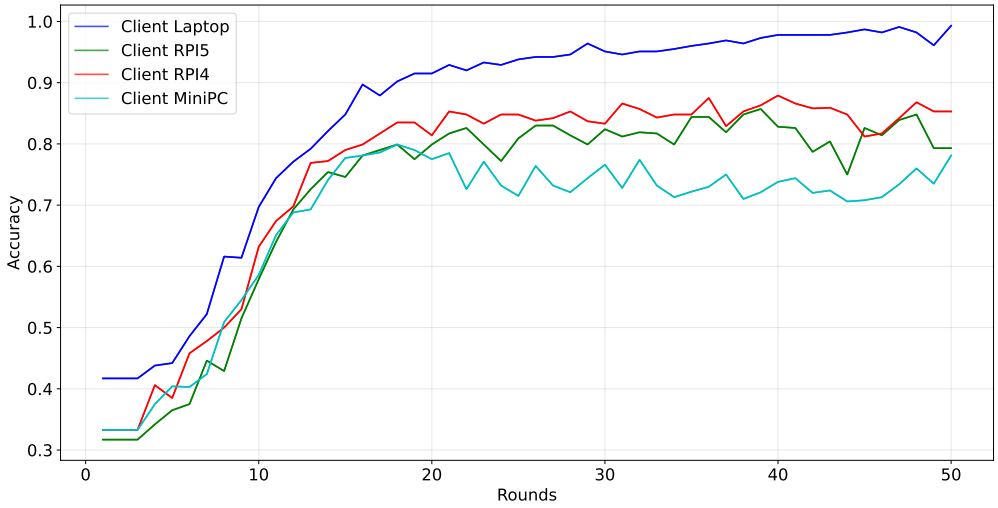

Figure 5.10: Accuracy in different training rounds.

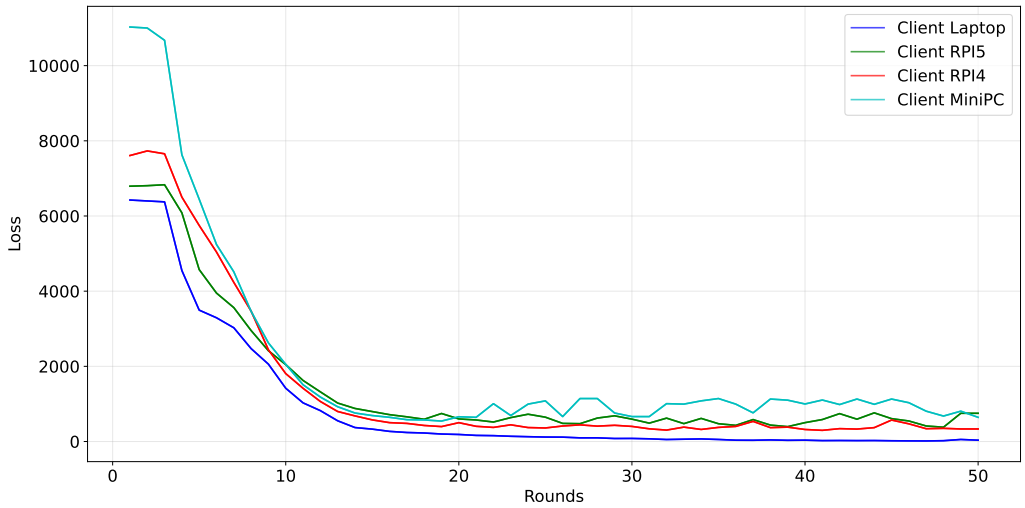

Figure 5.11: Loss in different training rounds.

In Figure 5.14, we can see that the time it takes for clients to complete their training increases after the third round. This happens because clients start adjusting based on the instructions of the algorithm, needing more training images and a number of epochs according to their workload capacity. In addition, training time changes over time. This is because it is affected by how much each client's CPU is being used. When CPU usage increases 80%, the clients reduce the number of images, which makes training time shorter. This helps prevent the devices from becoming too hot and causing problems.

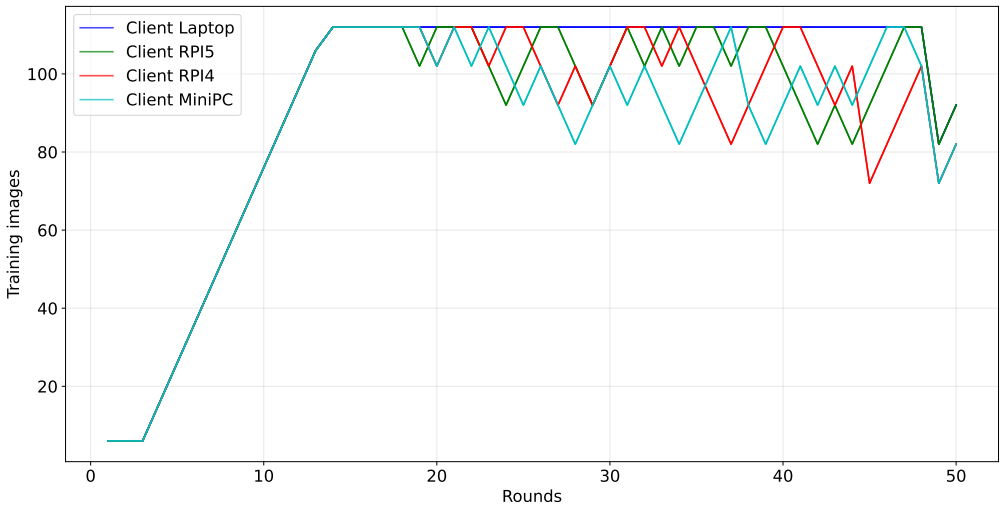

Figure 5.12: The number of training images achieved by different clients across various training rounds.

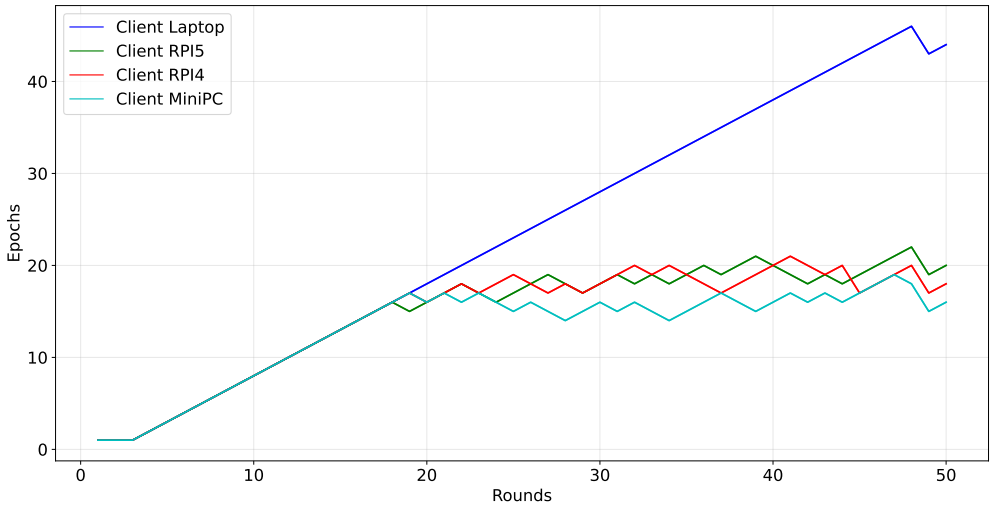

Figure 5.13: The number of and epochs achieved by different clients across various training rounds.

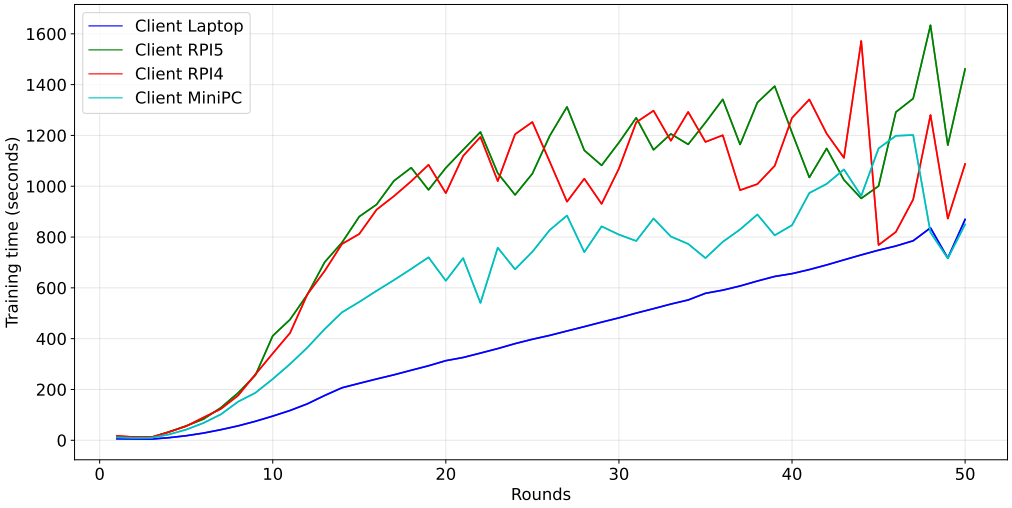

Figure 5.14: Training duration increases as adaptation begins, adjusting the number of image samples and epochs according to the workload.

5.4 Results AdaptiveMesh

Figure 5.15 illustrates the CPU utilization in four different clients, highlighting the significant computational demands of model training. However, the CPU adaptation mechanism in our algorithm improves the training efficiency. When CPU usage exceeds 80%, the algorithm adjusts the workload in subsequent rounds, reducing computational strain and preventing client overload. Figure 5.16 shows the utilization of RAM memory for different clients. Figure 5.16 indicates that clients managed the memory demands well, using them relatively steadily in different rounds.

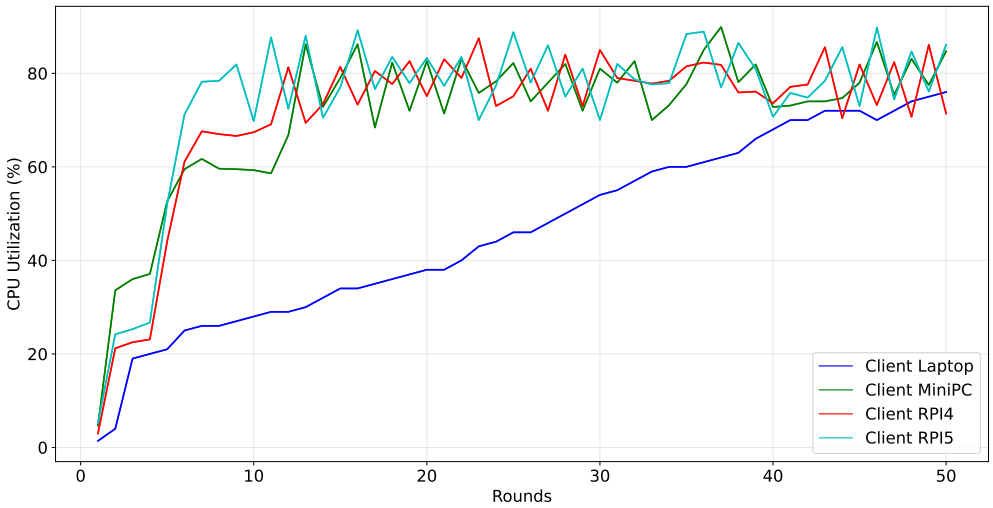

Figure 5.15: CPU usage across various training rounds.

Figure 5.17 illustrates the comparison of the accuracy achieved during training in different rounds. It is observed that as rounds increase, the accuracy also increases until the round 50. Specifically, the Laptop achieves 98% accuracy, MiniPC 83%, RPI4 82%, and RPI5 79%.

Figure 5.18 shows that clients initially experience high loss values, which progressively decrease as training rounds progress, indicating an improvement in model performance over time.

Similarly, Figure 5.19 presents the count of local epochs completed per round by different clients. The figure demonstrates a steady increase in epochs as rounds progress. However, after round 10, clients reaching their CPU limits (e.g., RPI4, RPI5) exhibit a decline in epochs due to the algorithm's adaptive behavior. Devices with higher CPU capacity (e.g., Laptop, MiniPC) continue to perform more epochs, whereas those with limited resources execute fewer epochs

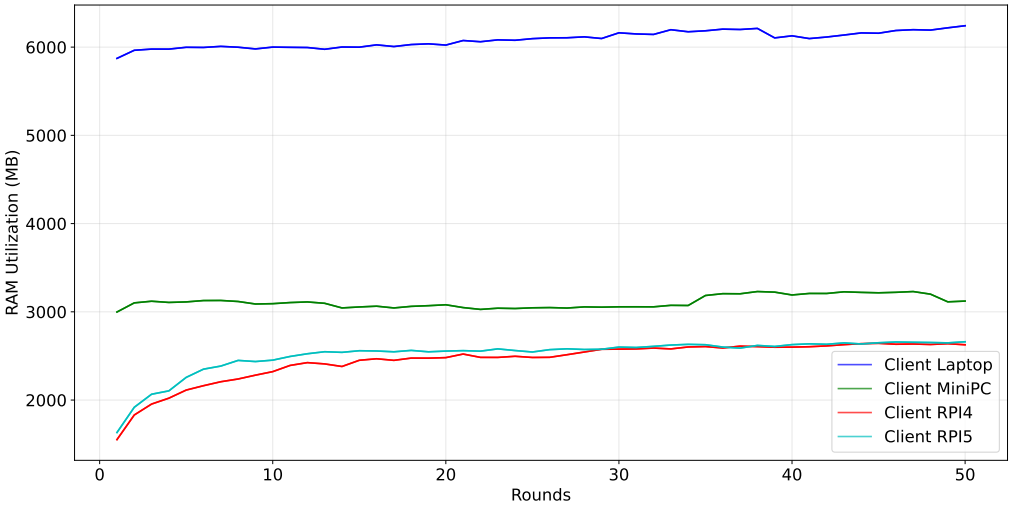

Figure 5.16: Memory usage across different training rounds.

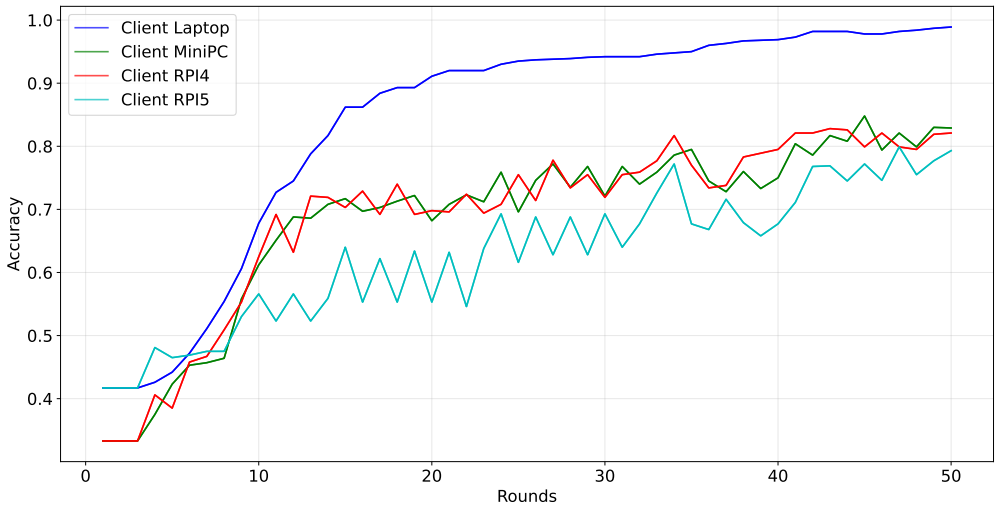

Figure 5.17: Accuracy in different training rounds.

accordingly. In round 50, the Laptop client reaches 48 epochs, the MiniPC reaches 18 epochs, the RPI4 reaches 16 epochs, and the RPI5 reaches 12 epochs.

Figure 5.20 illustrates the number of trained images per client, showing a steady increase as the training rounds progress. However, after round 10, lower performing clients struggle to handle the growing workload due to their CPU usage exceeding 80%. As a result, these clients reduce their workload to regulate CPU performance. When CPU usage drops below 80%, they request an increase in image samples. In contrast, the Laptop client, benefiting from higher computational capacity, continuously requests more images, eventually reaching

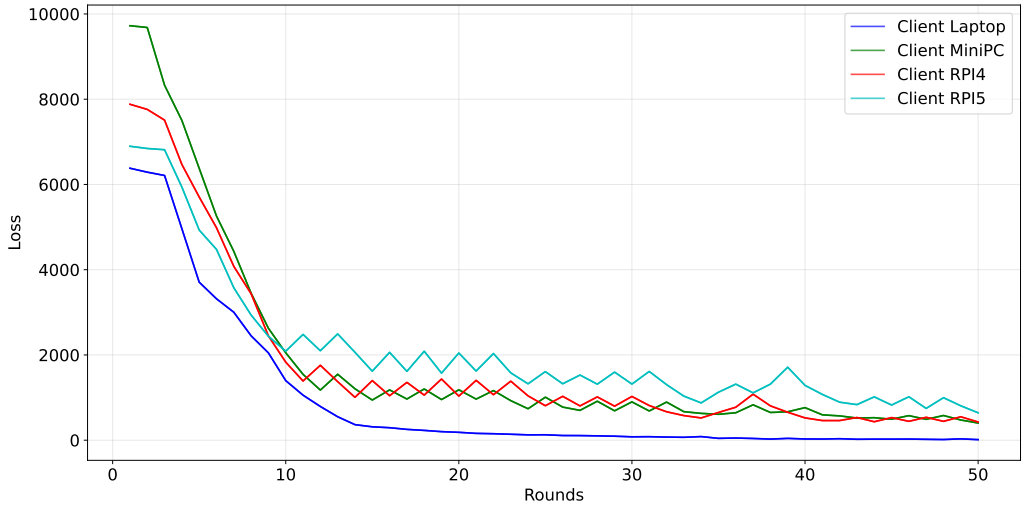

Figure 5.18: Losses in different training rounds.

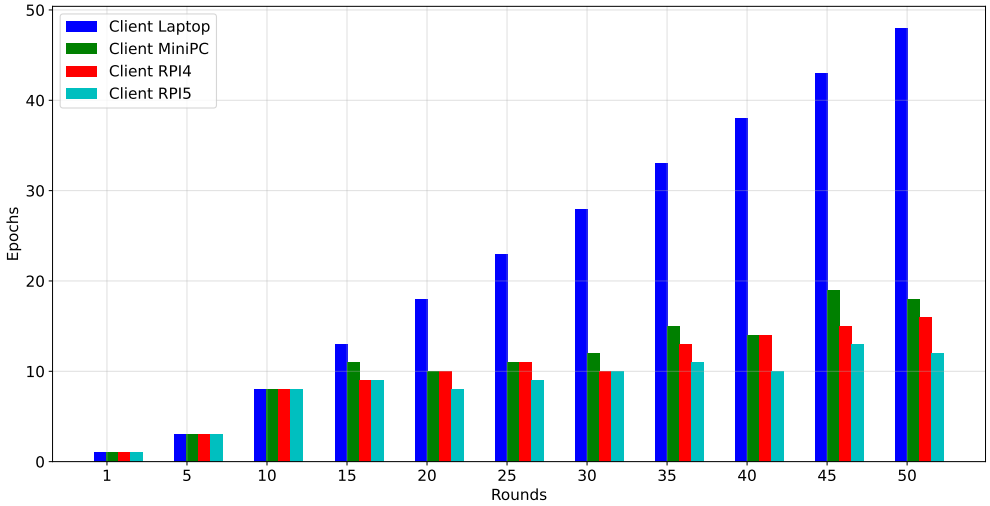

Figure 5.19: The number of epochs completed by different clients varied across training rounds

the maximum limit.

Figure 5.21 shows that the training time for clients increases after the third round as adaptation begins. During this phase, each client adjusts the number of training samples and epochs based on its available workload capacity. In addition, training time fluctuates, increasing or decreasing depending on the CPU usage of the client. When CPU utilization exceeds 80%, clients reduce the number of samples and epochs, which helps to lower training time and computational load, thus preventing device overheating.

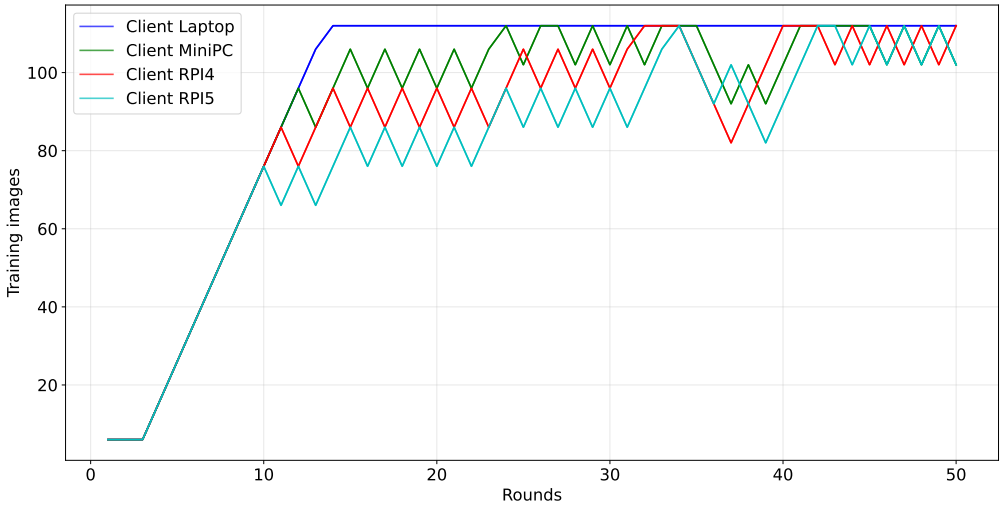

Figure 5.20: The number of trained images per client.

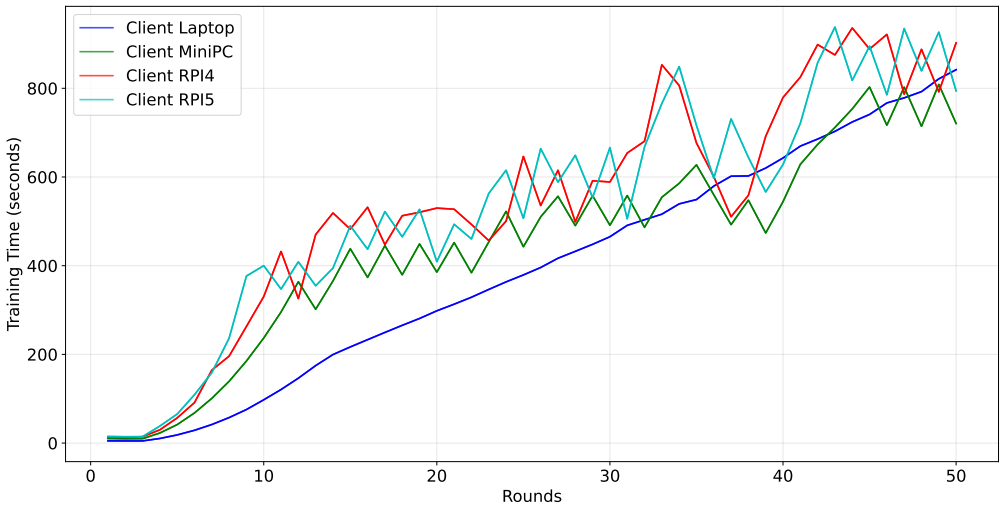

Figure 5.21: Training time grows as adaptation starts, adjusting the number of image samples and epochs depending on workload.

5.5 Statistical Results

In this study, the use of statistical techniques, such as ANOVA (Analysis of Variance), helps us determine whether there are significant variations in CPU usage, accuracy, the number of training images, and epochs among different FL client devices.

By examining these factors, we aim to understand how various device categories influence performance metrics, providing valuable insights into FL's effectiveness in diverse, heterogeneous environments.

Table 5.1 describes the performance of the CPU on multiple devices, such as laptops, MiniPCs, Raspberry Pi 4 (RPI4) and Raspberry Pi 5 (RPI5). The value of F derived from the One-Way ANOVA test ($F = 26.428$, $p < 0.01$) demonstrates notable differences in CPU performance between these devices.

Table 5.1: One-Way ANOVA Results on CPU Differences by Device

Device	N	Mean	Std. Deviation	F	Sig.
Laptop	50	46.528	19.6844	26.428	0.001
MiniPC	50	71.334	15.9116		
RPI4	50	71.224	17.8005		
RPI5	50	74.268	17.6564		

Table 5.2 presents the results of Tukey's Honestly Significant Difference (HSD) test. The findings highlight substantial variations in CPU utilization among different device types. Specifically, the Laptop exhibits lower CPU usage than the other three devices, with the MiniPC following, while RPI4 and RPI5 record higher utilization, in that order.

Table 5.2: Multiple Comparisons on CPU Differences by Device

Device	N	Mean	Std. Deviation	F	Sig.
Laptop	50	46.528	19.6844	26.428	0.001
MiniPC	50	71.334	15.9116		
RPI4	50	71.224	17.8005		
RPI5	50	74.268	17.6564		

Table 5.3 displays the One-Way ANOVA results regarding accuracy disparities among device types. The calculated F-value of 18.820, with a p-value below 0.01, confirms a statistically significant difference in precision between devices.

Table 5.4 presents Tukey's test outcomes related to accuracy variations across device types. Significant differences were identified between the Laptop and the other devices. The Laptop achieves superior accuracy compared to the remaining three devices, with the MiniPC showing the smallest accuracy difference relative to the Laptop, followed by RPI4 and RPI5, which demonstrate the lowest accuracy levels.

Table 5.3: One-Way ANOVA Results on Accuracy Differences by Device

Device	Mean	Std. Deviation	F	Sig.
Laptop	0.84032	0.188054	18.820	< 0.001
MiniPC	0.68696	0.140836		
RPI4	0.69128	0.141145		
RPI5	0.62826	0.108582		

Table 5.4: Multiple Comparisons on Accuracy Differences by Device

(I) device_id	(J) device_id	Mean Difference (I-J)	Sig.
Laptop	MiniPC	0.153360*	0.000
	RPI4	0.149040*	0.000
	RPI5	0.212060*	0.000

Table 5.5 presents the one-way ANOVA results that analyze epoch variations among device types. The calculated F-value of 35.648, with $p < 0.01$, confirms that the number of epochs differs significantly between devices.

Table 5.5: One-Way ANOVA Results on Epoch Differences by Device

Device	Mean	Std. Deviation	F	Sig.
Laptop	23.56	14.479	35.648	< 0.001
MiniPC	11.36	4.810		
RPI4	10.24	4.128		
RPI5	8.76	3.146		

Table 5.6 summarizes the results of the Tukey test that assess epoch variations by device type. Notable differences are observed between the Laptop and other devices. The Laptop achieves the highest number of epochs relative to the other devices. The MiniPC exhibits the smallest gap in epoch count compared to the Laptop, followed by RPI4, with RPI5 having the fewest epochs.

Table 5.7 outlines the results of the One-Way ANOVA test conducted to analyze the variations in training images between different types of devices. The calculated F value of 2.059,

Table 5.6: Findings from the multiple comparisons analysis highlighting variations in the number of epochs across different devices

(I) device_id	(J) device_id	Mean Difference (I-J)	Sig.
Laptop	MiniPC	12.200*	0.000
	RPI4	13.320*	0.000
	RPI5	14.800*	0.000

with $p > 0.05$, indicates that there are no statistically significant differences in the number of training images between devices. Among the tested devices, the Laptop records the highest average number of training images, followed by the MiniPC and RPI4, whereas RPI5 has the lowest average.

Table 5.7: Outcomes of the One-Way ANOVA analysis examining variations in the number of training images across different devices.

Device	Mean	Std. Deviation	F	Sig.
Laptop	95.44	33.276	2.059	0.107
MiniPC	90.04	31.219		
RPI4	86.08	30.028		
RPI5	80.64	28.584		

5.5.1 Regression Analysis

Table 5.8: Results of Regression Analysis

Dep. Variable	β	t	p
Accuracy	214.302	35.069	< 0.001
epoch	-1.223	-12.446	< 0.001
CPU	0.407	15.452	< 0.001
$R = 0.973$, $Adj.R^2 = 0.945$, $F = 1150.364$, $p < 0.001$			

Table 5.8 examines the impact of accuracy, epoch, and CPU on the number of training images. The results indicate that the three independent variables, accuracy, epoch, and CPU, significantly affect the number of training images. Specifically, higher accuracy (ß = 214.302, t = 35.069, p < 0.001) and increased CPU utilization (ß = 0.407, t = 15.452, p < 0.001) are both positively correlated with a higher number of training images. In conclusion, accuracy, epoch, and CPU utilization play an important role in determining the number of training images, with higher accuracy and CPU utilization leading to an increase in the number of images used for training.

5.6 Experimental Results - AdaptiveMESH

The goal of the experimentation in this study is to:

1. Evaluate the performance of the AdaptiveMesh adaptive algorithm: The experimentation aims to assess how well AdaptiveMesh optimizes resource usage (such as CPU utilization, temperature management, and bandwidth efficiency) in FL environments compared to a NAIVE non-adaptive FL algorithm.

2. Compare the effectiveness of AdaptiveMesh and NAIVE algorithms across different environments: The experimentation seeks to compare how these algorithms perform on physical devices (e.g., Raspberry Pis, Mini PCs, and laptops) versus virtual machine instances in Google Cloud. This comparison is intended to highlight the differences between the implementation of FL algorithms in physical versus virtual environments, providing insights into the practical implications and trade-offs involved in each scenario.

CPU Utilization: Figure 5.22 illustrates the percentage of CPU utilization during various training rounds for different devices. A horizontal red dashed line at 80% marks the CPU load limit to avoid overloading. The graph demonstrates that the Laptop device maintains a consistent and low CPU load throughout training rounds, unlike other devices that show significant increases in CPU load, sometimes exceeding 80%. This is managed by the AdaptiveMesh algorithm, which automatically optimizes the training process by reducing the resources when the CPU load exceeds 80%, ensuring stable performance and preventing device overloading. This analysis highlights the effectiveness of the AdaptiveMesh algorithm in *managing CPU load* during FL between different devices, optimizing performance, and avoiding conditions that could lead to reduced performance or cause the device to slow down.

Figure 5.23 shows the percentages of CPU utilization during various training rounds for the same devices without adaptive CPU load management, where resources increase with each round. Unlike the AdaptiveMesh approach, this method, referred to as NAIVE, does not adapt the CPU load. The graph reveals that, except for the Laptop device, all other devices exceed the 80% CPU utilization threshold as the number of images and epochs increases. The increase in tasks may require more energy, shortening battery life, or affecting the device's performance. The device may start to operate more slowly or may not be able to handle tasks. Additionally, increased CPU usage can lead to device overloading, indicating the importance of adaptive

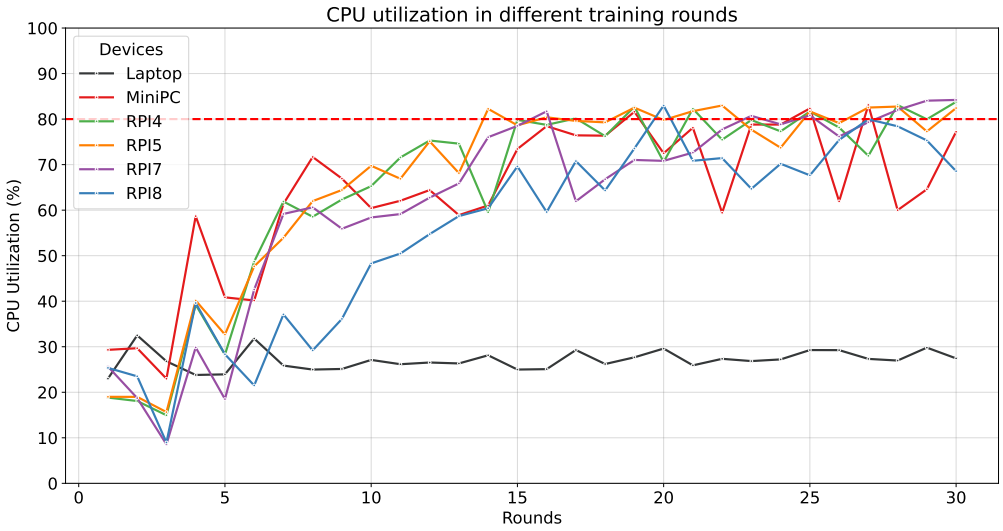

Figure 5.22: AdaptiveMesh's adaptive CPU usage across various training rounds.

management in maintaining optimal performance. Comparison of the two graphs highlights the effectiveness of the AdaptiveMesh algorithm in managing CPU load during FL. While *NAIVE approach leads to excessive CPU utilization and potential overheating*, AdaptiveMesh ensures a balanced load, maintaining the performance and longevity of the device.

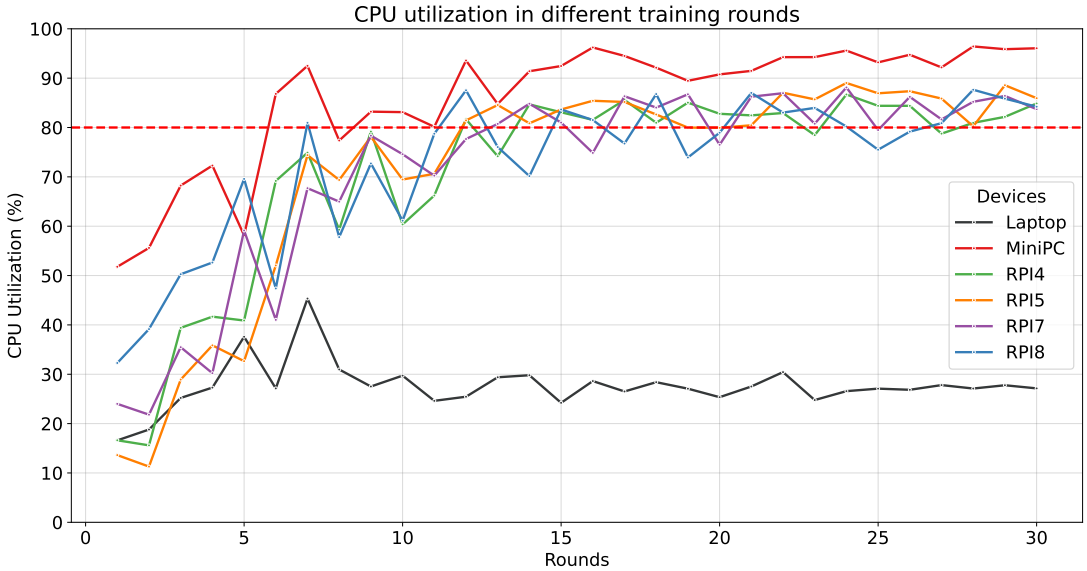

Figure 5.23: NAIVE CPU usage across various training rounds.

CPU Temperature: Figure 5.24 shows the CPU temperatures during various training rounds for the same devices using the AdaptiveMesh algorithm. With a red horizontal dashed line at 85°C marking the temperature limit. This graph highlights that when the CPU temperature exceeds 85°C, AdaptiveMesh reduces the workload by decreasing resources, effectively lowering

the CPU temperature. This adaptive resource management ensures that devices, particularly those with weaker hardware performance, are not overloaded, thereby avoiding performance degradation and conserving battery life. The *consistent temperature management* shown in the graph underscores AdaptiveMesh's ability to dynamically allocate resources based on temperature, ensuring optimal device performance and longevity.

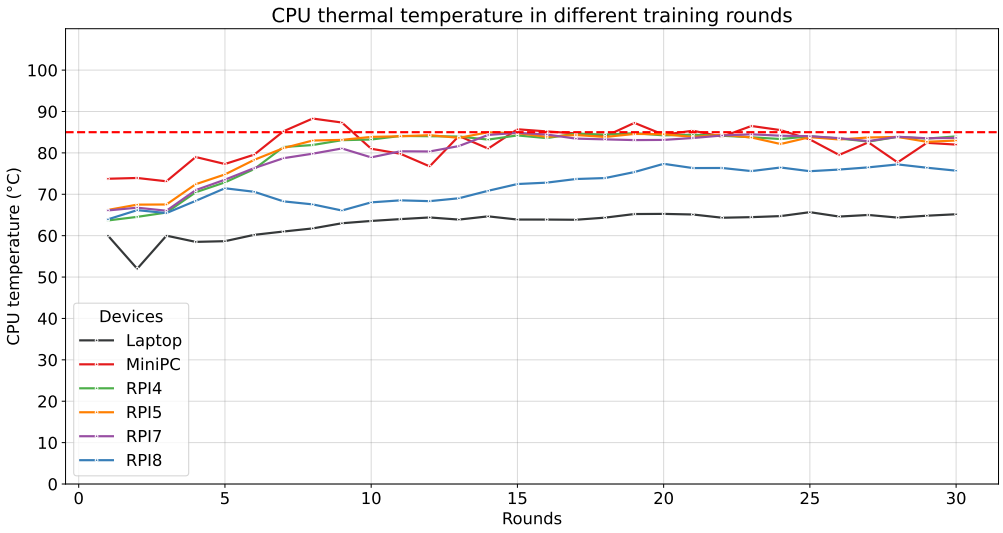

Figure 5.24: AdaptiveMesh's adaptive CPU temperature in different training rounds.

Figure 5.25 shows the CPU temperatures during various training rounds for different devices using a NAIVE approach, where the resources are increased with each round without adaptive management. Figure 5.25 shows that model training requires significant computational resources and highlights that, except for the Laptop, all other devices experience significant increases in CPU temperature, often exceeding the 85 ° C threshold as the number of images and epochs increases. The MiniPC, in particular, consistently exceeds this limit. The lack of adaptive management in the NAIVE approach results in *higher CPU loads and temperatures*, highlighting the necessity of adaptive algorithms like AdaptiveMesh to ensure optimal device performance.

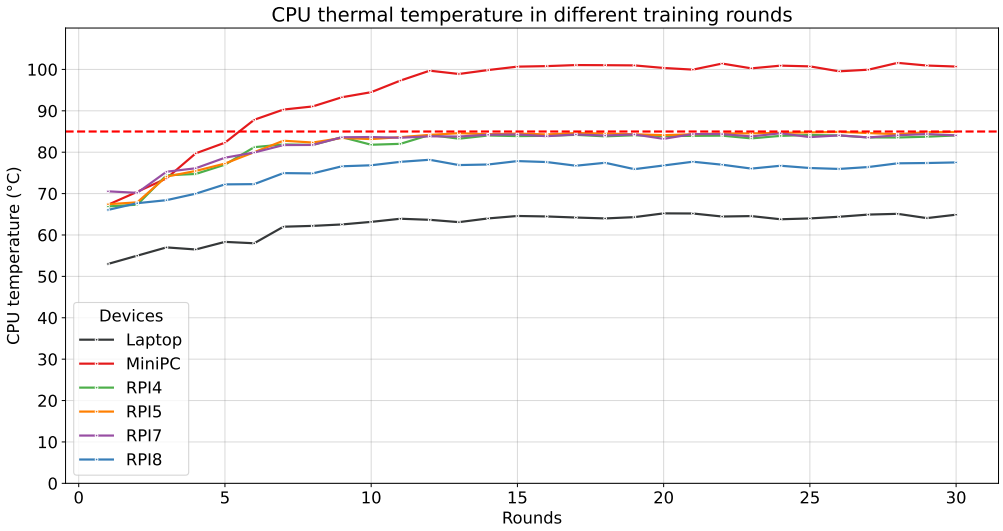

Figure 5.25: NAIVE CPU temperature in different training rounds.

Training Accuracy: Figure 5.26 shows the accuracy of different devices during training rounds when using an adaptive approach. It is noted that accuracy improves as the number of rounds increases. The Laptop achieved the highest accuracy of 95.1%. The RPI4, RPI5, RPI7, and RPI8 maintained accuracy around 80-83%. The MiniPC showed the lowest accuracy of 57.9%. Adaptive management improved accuracy and maintained device performance by adjusting training based on predefined thresholds. The arrows indicate that AdaptiveMesh intervened when the CPU load exceeded 80%, the CPU temperature exceeded 85°C, the bandwidth exceeded 2 Mbps, and the training time limit was exceeded. As a result, there were drops in accuracy during these instances because AdaptiveMesh reduced the number of resources allocated, leading to a decrease in accuracy. However, despite the slight loss in accuracy, the primary goal is to prevent excessive CPU load, reduce device slowdown, and minimize battery consumption. In general, the adaptive approach helps to maintain a balance between accuracy and system performance, preventing overloading, as shown by the annotations.

Figure 5.27 presents the progression of the accuracy in different training rounds using the NAIVE approach for the same devices. Across all devices, accuracy improves as the number of training rounds increases. This trend is expected since more rounds involve more training data and epochs, enhancing the model's learning capability. The Laptop device consistently achieves the highest accuracy at the end of the training rounds. In contrast, the MiniPC shows a lower peak accuracy of 84.4%, likely due to its lower hardware capacity. Devices like RPI4, RPI5, and RPI7 show similar convergence trends, stabilizing around 87-91%. This suggests that

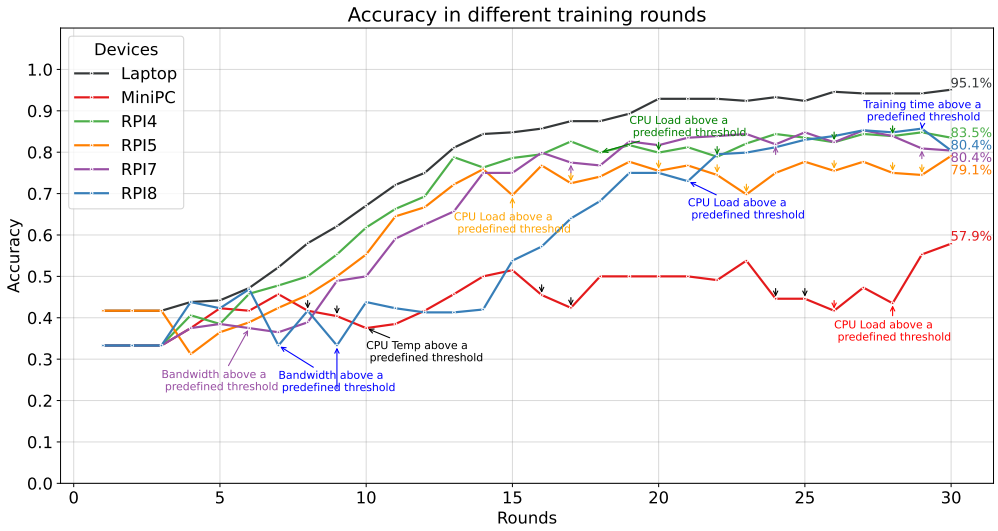

Figure 5.26: Accuracy in different training rounds (AdaptiveMesh).

these devices, while having different hardware capabilities, perform similarly under the NAIVE approach given enough rounds. The steady increase in accuracy comes at a cost. The NAIVE approach's continuous demand for more resources imposes a significant workload on devices, especially those with lower hardware capacities. This results in a higher CPU load, an increased training time, and a higher temperature, potentially causing a faster battery depletion and reduced device longevity. The increase in training time is a notable drawback, highlighting the need for a balanced approach to maintain accuracy without overburdening system resources.

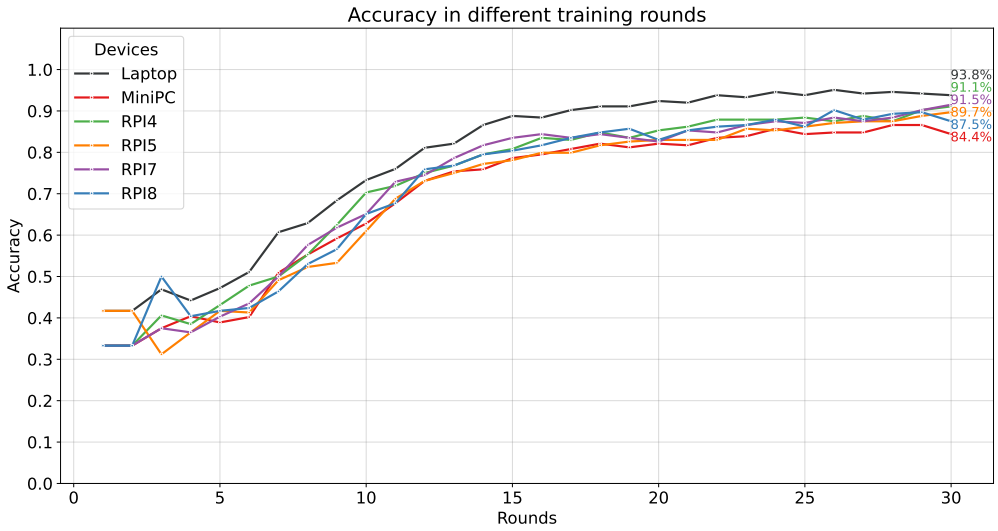

Figure 5.27: Accuracy in different training rounds (NAIVE).

Epochs per device: Figure 5.28 displays the number of local epochs per device in different

training rounds. It illustrates how the number of epochs changes between training rounds for each device used. It is crucial to highlight the role of the AdaptiveMesh algorithm in this variation. This algorithm automatically reacts not only when the CPU load exceeds 80%, but also when the CPU temperature is above 85%, the bandwidth is above 2 Mb/s, and the training time exceeds 30 minutes. The AdaptiveMesh algorithm reduces the resources to lower the load, ensuring stable performance. In the graph, the red arrows indicate moments when this reduction occurs, leading to an immediate decrease in the number of local epochs. The graph analysis demonstrates the effectiveness of the AdaptiveMesh algorithm in managing CPU load and optimizing federated model training. The automatic reduction in the number of epochs and training images ensures a good balance while optimizing bandwidth usage and training time.

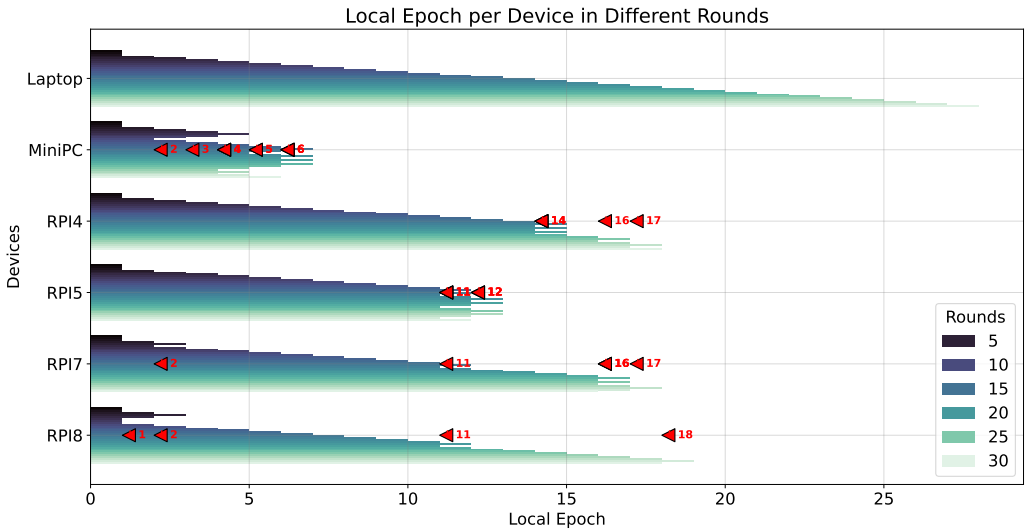

Figure 5.28: Number of epochs obtained by various clients throughout different training rounds (AdaptiveMesh).

Figure 5.29 displays the number of local epochs per device across different training rounds under a NAIVE approach. It illustrates how the number of epochs continuously increases without any reaction to hardware performance or device load. Unlike the AdaptiveMesh algorithm, which dynamically adjusts the number of epochs based on CPU load, temperature, bandwidth, and training time, the naive approach continuously increases the number of epochs regardless of the hardware performance or load of the device. This approach does not incorporate any mechanism that can affect the overall efficiency of the training process, especially on resource-constrained devices.

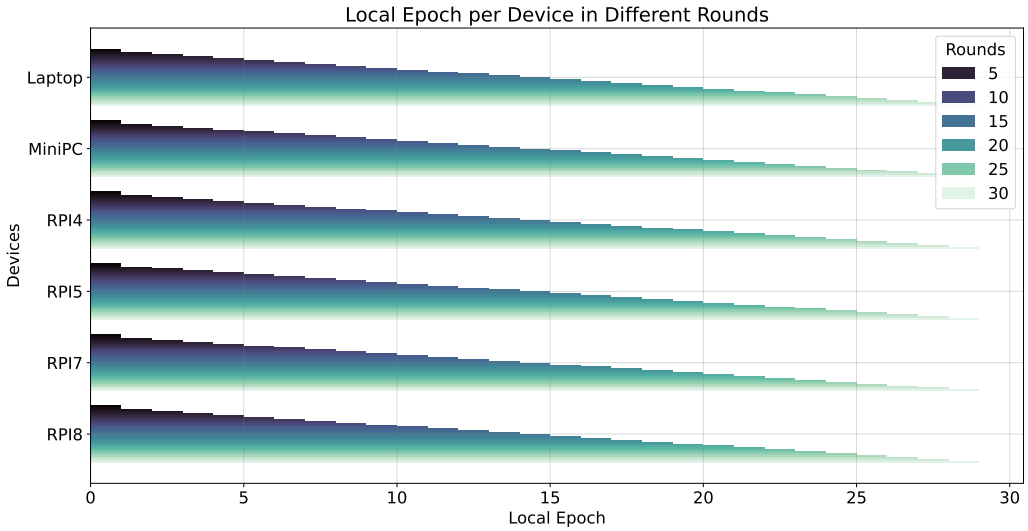

Figure 5.29: Number of epochs obtained by various clients throughout different training rounds (NAIVE).

Trained images per device: Figure 5.30 illustrates the distribution of training images between various devices over different rounds of training. The AdaptiveMesh algorithm dynamically adjusts the allocation of training resources based on the predetermined threshold, the algorithm reduces the number of training images assigned to devices with lower hardware performance to ensure efficient resource utilization and prevent overloading. The red arrows indicate the points where the AdaptiveMesh algorithm has intervened to reduce the number of training images. The MiniPC for example, experiences reductions in training images at multiple stages, indicating that these devices frequently encounter threshold exceeding, prompting the AdaptiveMesh algorithm to adjust their workloads. Device such as the Laptop can handle a higher number of training images across more rounds without as frequent interventions, reflecting their relatively better hardware performance.

Regarding the quantity of images processed per client, Figure 5.31 shows a steady increase as the training rounds advance. However, after the 10th round, certain lower capacity clients struggle to manage the increasing image and epoch processing load, as their CPU utilization exceeds 80%. Consequently, these devices decrease their computational load to regulate CPU usage. When CPU consumption falls below 80%, they request a higher allocation of the image. On the other hand, the Laptop client, which has greater processing power, continuously requests more images, eventually reaching the highest permissible limit.

This Figure 5.31 demonstrates the distribution of training images between various devices

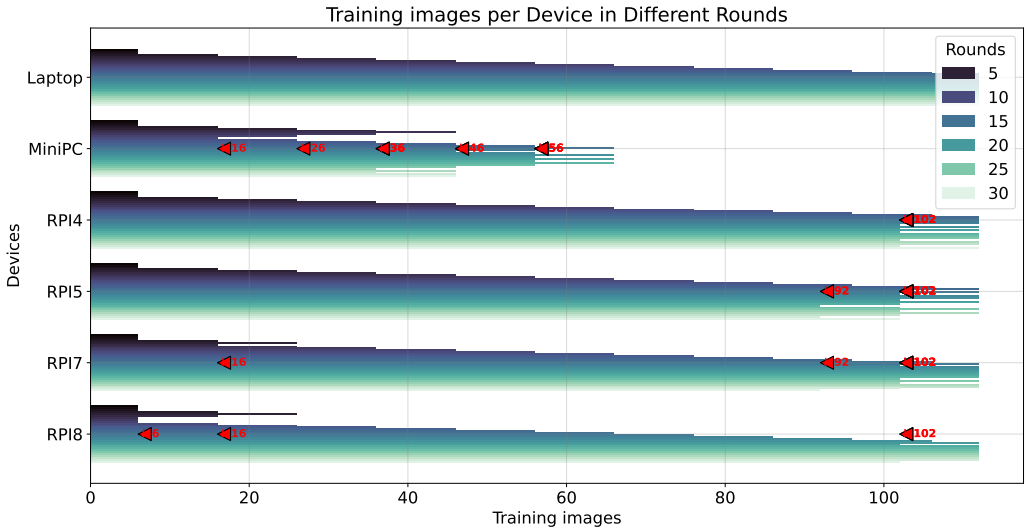

Figure 5.30: Each client's number of training images (AdaptiveMesh).

during different training rounds using the NAIVE approach. Unlike the AdaptiveMesh algorithm, the NAIVE approach does not adapt or react based on the performance metrics of the devices. As a result, all devices are assigned the same number of training images regardless of their individual hardware capabilities or performance constraints. In the NAIVE approach, each device handles the same workload, which could lead to inefficiencies and potential overloads on devices with lower hardware performance. Unlike the AdaptiveMesh algorithm, there are no interventions or adjustments in the number of training images based on device performance, which results in suboptimal training times and resource utilization. The NAIVE approach can lead to overloading less capable devices, whereas AdaptiveMesh aims to optimize resource allocation by reducing the workload on such devices when necessary.

Training time: Figure 5.32 shows the training time for various devices in different training rounds. Figure shows the adaptive behavior of the AdaptiveMesh algorithm in response to specific thresholds of CPU load, CPU temperature, bandwidth, and training time. Arrows are placed at points where the CPU load threshold, CPU temperature, bandwidth, and training time exceed. The Laptop maintains a relatively stable training time. MiniPC shows several instances where the CPU temperature and load exceed thresholds, triggering adjustments to reduce training time. RPI4, RPI5, RPI7, and RPI8 experience small variations in training time, with adaptive measures and other metrics reaching critical levels. This adaptive approach ensures efficient resource usage and optimizing training performance in real-time across different rounds and devices, enhancing the overall efficiency of the FL process. The ability of the

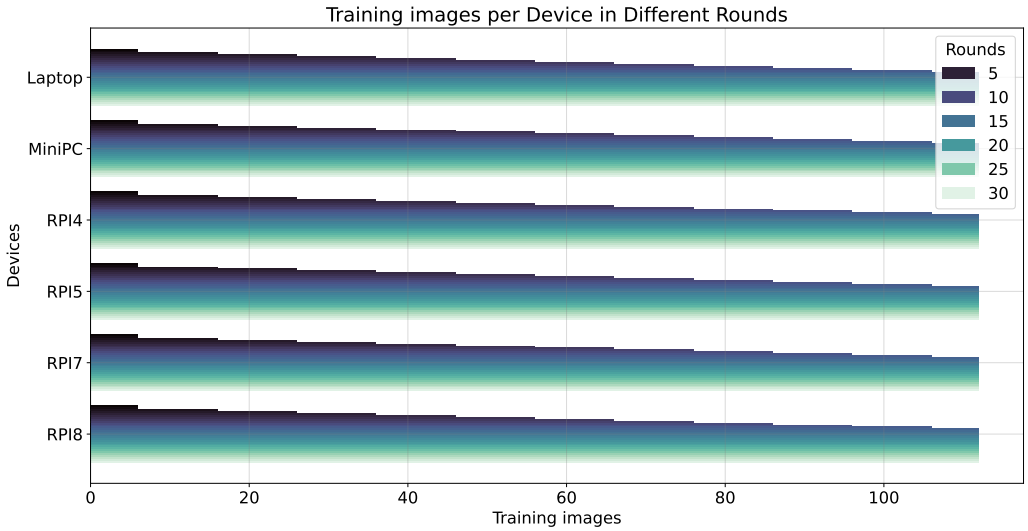

Figure 5.31: Each client's number of training images (NAIVE).

AdaptiveMesh algorithm to dynamically adjust training parameters in response to hardware performance metrics is crucial to maintaining optimal training times and preventing device overload.

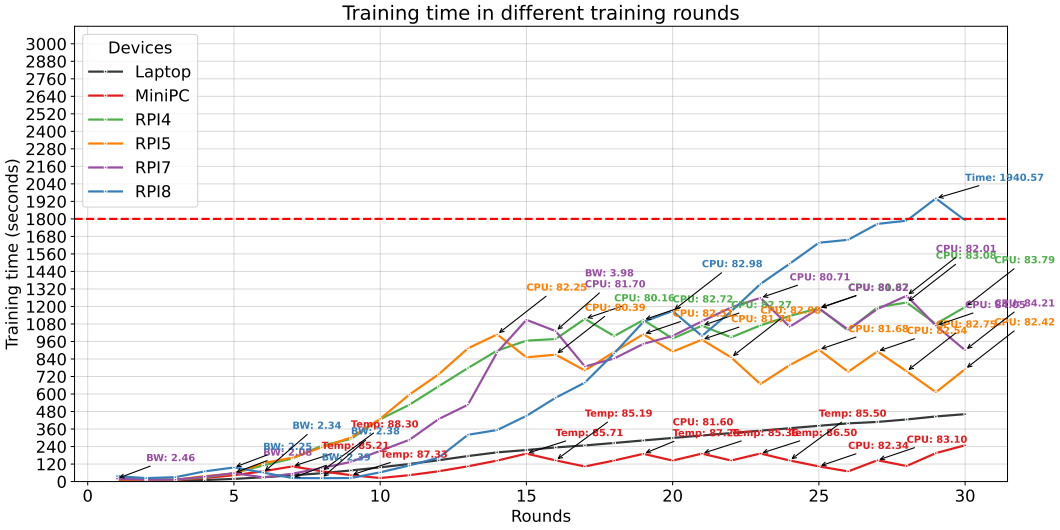

Figure 5.32: Training duration increases as adaptation starts, with adjustments to image samples and epochs based on the workload (AdaptiveMesh).

Figure 5.33 shows the training time for each device during the 30 training rounds under a naive, non-adaptive approach. The training time for each device increases steadily as the number of training rounds progresses. The lines represent the cumulative training time for each round, reflecting the constant allocation of resources without any adaptive measures. The

absence of adaptive adjustments leads to continuously increasing training times, potentially leading to inefficiencies and device overloads in resource-constrained environments.

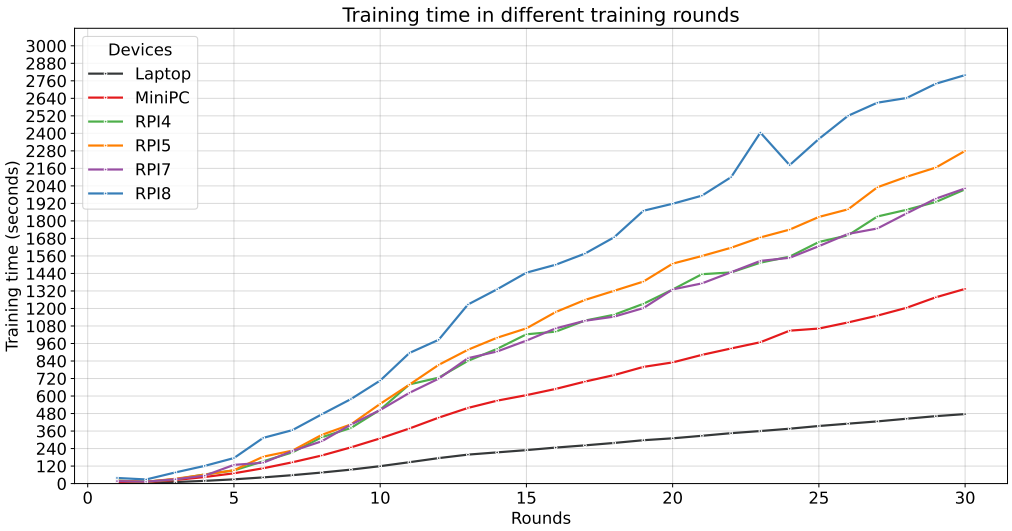

Figure 5.33: Training time grows as adaptation begins, with changes to image samples and epochs according to workload (NAIVE).

Bandwidth usage: In Figure 5.34, the utilization of the bandwidth between different rounds and devices is visualized using the adaptive adaptive AdaptiveMesh algorithm. Figure 5.34 (top left plot) shows the density distribution of rounds. Most rounds are concentrated around the midpoint. In the same figure (Top Right Plot) the distribution of devices across different rounds is shown. Each dot represents a device in a specific round. The bottom left plot illustrates the relationship between bandwidth utilization and the number of rounds. Different colors indicate different devices. Finally, the lower right plot represents the density distribution of bandwidth utilization for each device. This helps identify bandwidth usage patterns across different devices.

The pair plot effectively visualizes the adaptive nature of the AdaptiveMesh algorithm by showing how bandwidth utilization changes between different rounds and devices. The adaptive approach allows efficient bandwidth management, preventing overload, and optimizing performance across all devices.

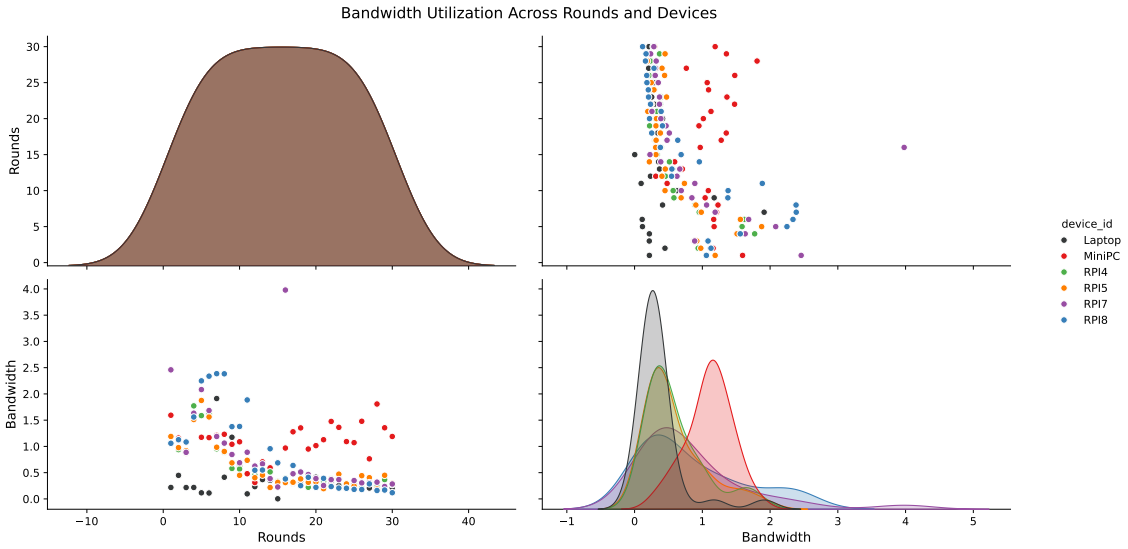

Figure 5.34: Bandwidth Utilization Across Rounds and Devices Using the Adaptive Algorithm (AdaptiveMesh).

In Figure 5.35, it can be observed that the non-adaptive version shows less uniformity in the distribution of the bandwidth between rounds and devices. The bandwidth utilization is concentrated more in the earlier rounds, suggesting that the non-adaptive algorithm does not regulate bandwidth usage as effectively over time. This could potentially lead to inefficiencies and higher loads on the network early in the process.

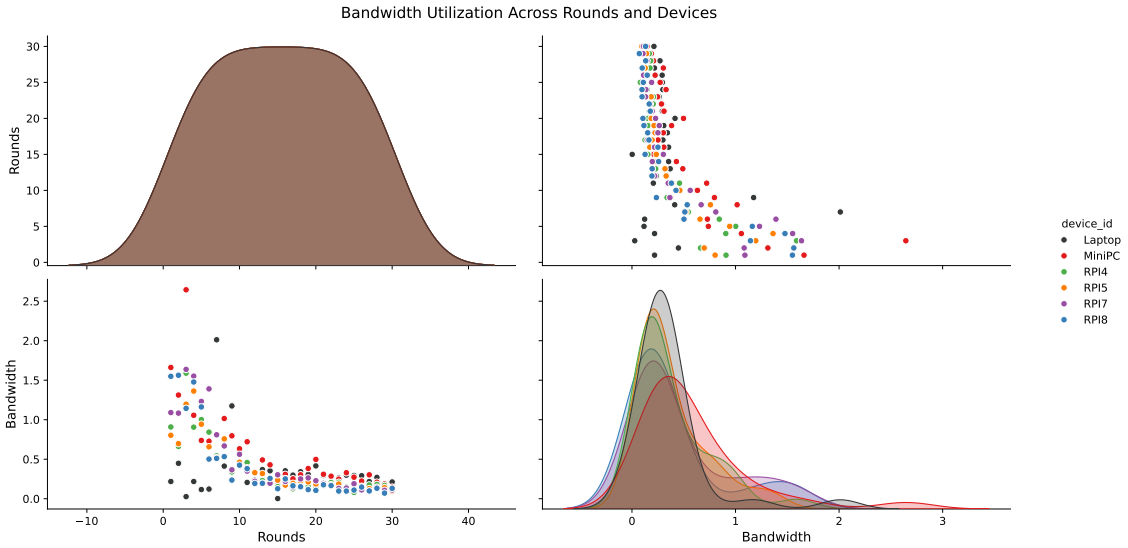

Figure 5.35: Bandwidth Utilization Across Rounds and Devices (NAIVE Algorithm).

5.7 Google Cloud Virtual Environment

In order to measure the impact of the wireless environment when training FL models on resource-constrained devices, we also tested the AdaptiveMesh in a Google Cloud environment. Google Cloud is a set of cloud computing services that use the same infrastructure as Google's end-user applications, including Google Search, Gmail, and YouTube.

In Google Cloud, we have created instances that closely resemble our physical hardware, such as a Raspberry Pi 4, a MiniPC NEO Z83-4, or a laptop with specific CPU and RAM requirements. For a laptop with an Intel(R) Core(TM) i5-8250U CPU and 8 GB of RAM, we configured a Google Cloud instance using the N2 Series with 4 core 8 logical processors, 8 GB of RAM and an SSD Persistent Disk. For the MiniPC NEO Z83-4, we selected an instance from the E2 Custom Series with a 4 logical core logical processor, 4 GB of RAM and a persistent standard disk in Google Cloud.

Google Cloud offers instances with ARM processors under its Tau T2A series. These instances use Ampere Altra processors, which are based on ARM architecture and are similar to the Cortex-A72 in terms of architecture. To create instances comparable to our physical devices, such as Raspberry Pi 4 Model B units with a Quad Core Cortex-A72 CPU, we chose a T2A instance with 4 cores and 8 logical processors, along with a Standard Persistent Disk to meet basic storage needs, similar to how an SD card is used in a Raspberry Pi.

Figure 5.36 depicts CPU utilization (%) during different training rounds across various devices simulated as virtual machines in Google Cloud. The figure depicts CPU utilization (%) during different training rounds on various devices used in the experiment. The dashed red line marks the 80% CPU threshold, which is a limit where AdaptiveMesh starts adapting if this limit is exceeded. As seen in the figure, the Laptop exhibits the lowest CPU utilization, starting around 20% and gradually increasing to about 40%, staying well below the 80% threshold. The RPI4, RPI5, RPI7 and RPI8 devices show a similar behavior, with varying levels of CPU utilization fluctuating between 60-80%. The MiniPC experiences more significant fluctuations, often exceeding the threshold 80%. However, when the CPU exceeds this limit, the algorithm allocates fewer resources to the device, resulting in a subsequent decrease in CPU utilization. This illustrates the algorithm's effectiveness in managing CPU resources dynamically to prevent overloading during intensive training processes. From this, we observe that CPUs with 4 cores and 8 logical processors handle training well, while processors with 2 cores and 4 logical pro-

cessors experience CPU utilization exceeding 80%, which could lead to reduced performance to avoid overheating.

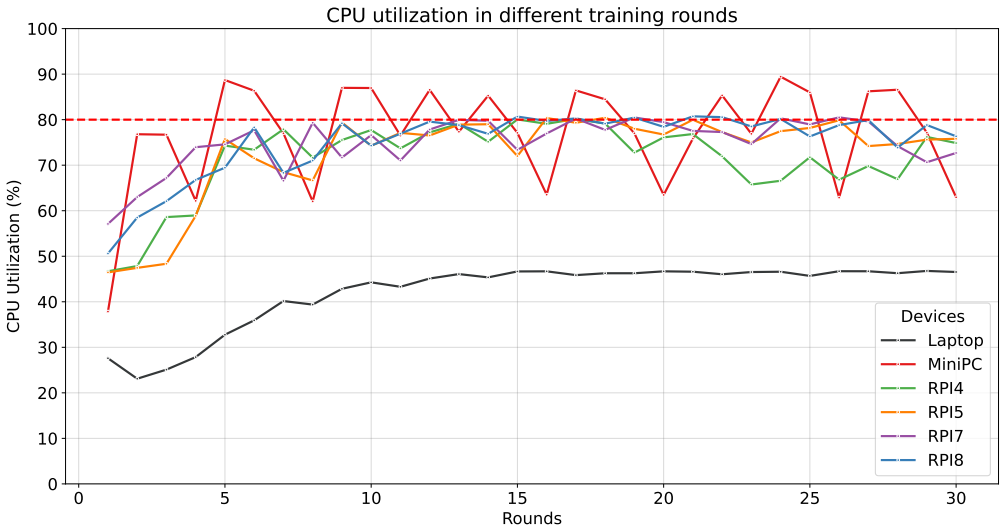

Figure 5.36: AdaptiveMesh's adaptive CPU usage across training rounds.

Figure 5.37 shows the percentages of CPU utilization during various training rounds, where resources are inserted with each round without adaptive techniques to prevent CPU overload. In this scenario, CPUs with fewer cores and logical processors (e.g., 2 cores, 4 threads) are likely showing higher CPU utilization more consistently above the 80% threshold, which could indicate increased risk of overloading. This lack of adaptation would cause more strain on devices, particularly those with limited resources. AdaptiveMesh adjusts the CPU load during training, ensuring stable performance and extending the lifespan of devices. Unlike the NAIVE approach, which leads to excessive CPU usage and increases the risk of overloading, AdaptiveMesh dynamically manages the workload to prevent such issues.

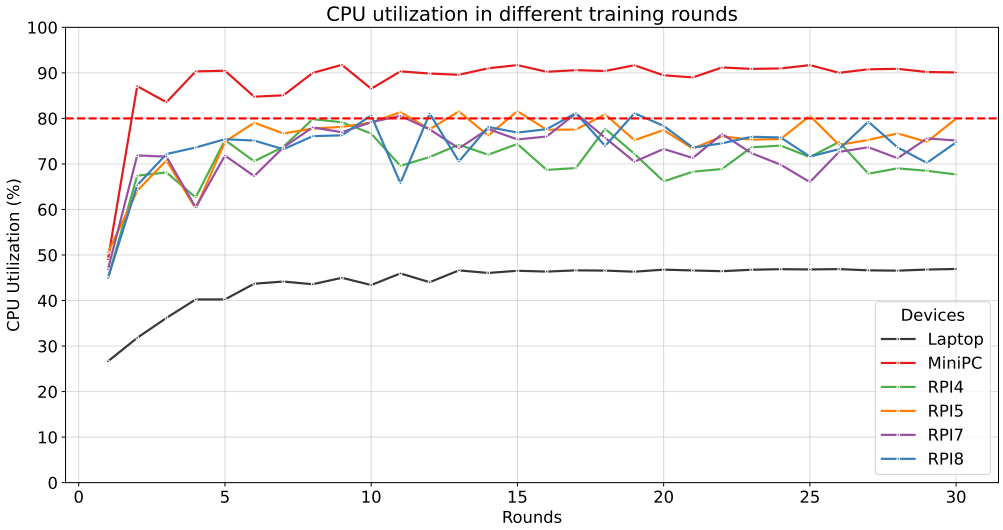

Figure 5.37: NAIVE CPU usage across training rounds.

Figure 5.38 illustrates the accuracy with threshold violations (CPU and bandwidth) over 30 rounds using the adaptive method. The initial accuracy is lower compared to the non-adaptive graph, with the lowest accuracy reaching 51.0%. The red arrows indicate where the CPU exceeded its threshold, while the blue arrows show bandwidth threshold violations. These threshold breaches contributed to the reduction in accuracy. Exceeding predefined CPU or bandwidth limits negatively affected the training process, but these interventions are crucial to avoid performance throttling and prevent system overload.

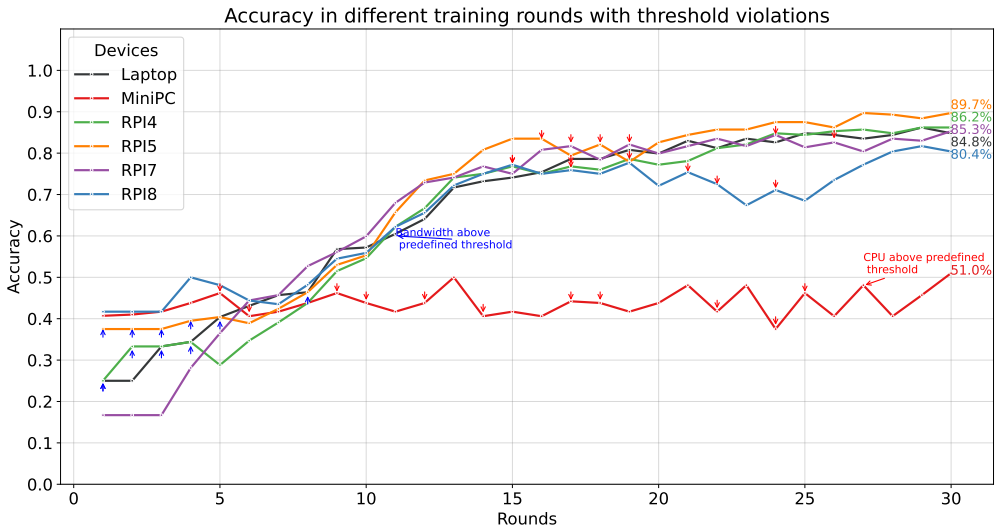

Figure 5.38: Accuracy in different training rounds (AdaptiveMesh).

Figure 5.39 shows the accuracy during 30 rounds of training for different devices (Laptop, MiniPC, RPI4, RPI5, RPI7, RPI8) without any adaptive mechanisms. The trend indicates that as

training continues, the accuracy increases slightly across all devices. The improvement in accuracy occurs because resources are increased in each round, allowing the model to learn more data and perform better over time. However, continuously increasing rounds, epochs, images, and other resources without adaptation can lead to several issues. First, the CPU load may become excessively high, potentially resulting in performance throttling as the devices attempt to manage the heat generated by the increased processing demands. This can slow down devices, particularly those with limited hardware resources. Furthermore, prolonged training time due to the increase in training data can lead to inefficiencies, particularly on lower-performance devices. High CPU utilization also increases power consumption, which can accelerate battery life in portable or IoT devices. Ultimately, without adaptive mechanisms, constant increase in computational load can cause performance degradation and potentially lead to system instability or restarts.

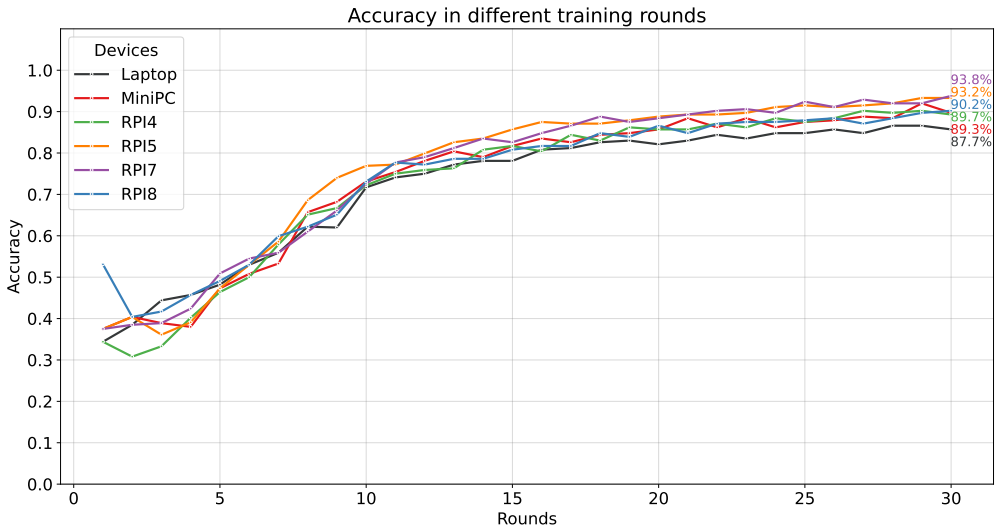

Figure 5.39: Accuracy in different training rounds (NAIVE).

5.8 Comparison of AdaptiveMesh, FedAda, and FedALA

We compared AdaptiveMesh with two existing algorithms, FedAda and FedALA, which are used in different FL scenarios for resource-constrained devices. FedAda focuses on distributing training workloads based on device computing power, but lacks a specific mechanism to manage CPU overload and temperature. On the other hand, FedALA adapts the aggregation process for heterogeneous clients but does not optimize network bandwidth or training time. In contrast, AdaptiveMesh offers better resource adaptation, dynamically adjusting the number of epochs and the size of the training dataset based on CPU usage, temperature, and bandwidth constraints. This makes it ideal for environments with limited resources, where maintaining stability and effectively managing resources are essential.

Table 5.9: Comparison of AdaptiveMesh, FedAda, and FedALA

Algorithm	**Adaptability to Device Load**	**CPU Usage Optimization**	**Dynamic Bandwidth Adjustment**	**Model Accuracy**	**Training Time**
AdaptiveMesh	Yes (based on CPU usage and temperature)	Yes (reduces overload)	Yes (adjusts network usage in real-time)	Slightly lower to maintain device performance	Shorter, as it prevents excessive workload
FedAda	Yes (uses an adaptive workload distribution method)	Not always (focuses on speed of convergence, no mechanism for temperature control)	No (does not adjust bandwidth dynamically)	Higher, but may increase device workload	Improves training speed, but may increase load on devices with limited resources, as it prioritizes accuracy
FedALA	Yes (adapts local aggregation for heterogeneous clients)	Partially (Reduces communication overhead, but does not monitor CPU load in real time)	No (It uses adaptive aggregation to reduce communication overhead, but does not adjust bandwidth dynamically)	Higher on powerful devices but not always for IoT	Improves training speed by reducing communication overhead

Part IV. Discussion, Conclusions and Future Work

Chapter 6

Discussion & Conclusion

6.1 Discussion

In this thesis, we focus on the implementation of FL in resource-constrained environments, particularly in wireless networks and IoT devices. FL is a machine learning paradigm that allows models to be trained in a decentralized manner, preserving data privacy. However, implementing FL in such environments presents significant challenges, including computational resource limitations, network bandwidth constraints, and device heterogeneity.

This thesis focuses on optimizing FL performance in resource-constrained environments. The proposed algorithms use adaptive mechanisms to adjust training parameters based on device performance and network conditions, improving training efficiency and stability.

FederatedMesh: This research emphasized that increasing the number of client devices within a mesh network enhances the accuracy of the model but simultaneously increases the usage of CPU and RAM. The FederatedMesh framework demonstrated that FL can effectively facilitate medical model training on distributed devices; however, achieving an optimal balance between accuracy and the use of computational resources is essential. This aligns with BACA and AdaptiveMesh, both of which demonstrate how adaptive algorithms dynamically manage this trade-off by adjusting training parameters in real time, taking into account factors such as CPU utilization, network bandwidth, and device temperature.

BACA: This algorithm focused on optimizing CPU and bandwidth usage in wireless environments. The results showed that BACA could adapt to network conditions and dynamically change training parameters, improving the accuracy of the model and the speed of convergence.

AdaptiveMesh: This algorithm demonstrated the ability to dynamically manage CPU load, temperature, and bandwidth usage, preventing device overload and improving training efficiency. Compared to non-adaptive algorithms, AdaptiveMesh maintained optimal device performance, especially in resource-constrained devices like Raspberry Pi and MiniPC. The AdaptiveMesh algorithm, evaluated across physical and virtual environments (e.g., Google Cloud), emerged as a robust solution for heterogeneous settings. By dynamically reducing workloads when thresholds are exceeded, it prevents device overheating and network congestion.

Compared to other studies, such as FedAda and FedALA, the algorithms proposed in these papers showed significant improvements in resource management and training performance. For example, while FedAda optimizes convergence speed and FedALA reduces communication overhead, neither explicitly manages CPU load or thermal conditions. AdaptiveMesh and BACA offer more advanced mechanisms for managing temperature and bandwidth, making them more suitable for resource-constrained environments.

FL has broad uses, especially in the medical field, where protecting data is critical. For example, in remote patient monitoring, FL enables the processing of data from wearable devices without exchanging it with a central server. In industrial settings, FL facilitates the examination of sensor information for predictive maintenance, safeguarding data security, and optimizing resource usage.

6.1.1 Strengths and Weaknesses

FederatedMesh:

- **Strengths:** Demonstrated the feasibility of FL in real-world IoT devices with limited resources. Indicated that enhancing the number of epochs and devices contributes to improved model accuracy, particularly in the initial epochs.
- **Weaknesses:** The rise in epochs also leads to increased CPU and RAM utilization, which could pose a limitation in environments with limited resources.

BACA:

- **Strengths:** Effectively managed CPU and bandwidth usage in wireless environments. Showed adaptive behavior in resource allocation, leading to improved model accuracy and convergence.
- **Weaknesses:** The algorithm's performance is highly dependent on network conditions, which can vary significantly in real-world scenarios.

AdaptiveMesh:

- **Strengths:** Dynamically adjusted training parameters based on real-time device metrics, preventing device overload and overheating. Demonstrated significant improvements in CPU and temperature management. AdaptiveMesh was compared with a non-adaptive algorithm (NAIVE), showing clear advantages in resource management and training efficiency. Demonstrated the effectiveness of the algorithm in both physical and virtual environments.
- **Weaknesses:** The algorithm sometimes sacrifices accuracy to maintain system stability, which may not be acceptable in all applications.

6.2 Conclusion

In conclusion, these studies demonstrate that adaptive algorithms such as AdaptiveMesh and BACA are highly effective in optimizing FL performance in resource-constrained environments. They provide dynamic mechanisms for resource management, improving training efficiency and stability, especially in devices with limited computational power and in networks with limited bandwidth.

The results of these studies highlight the importance of adaptive mechanisms in FL, particularly in heterogeneous and resource-constrained environments. They provide a solid foundation for the development of more advanced algorithms in the future, which can further enhance FL performance while minimizing resource strain on participating devices.

The comparative analysis between physical devices and virtual machines highlights the flexibility of AdaptiveMesh. Ensure system stability on low-power devices while leveraging the flexibility of cloud environments. In addition, its applicability in mobile and industrial IoT settings demonstrates its potential for real-world use cases, including personalized learning, healthcare monitoring, and predictive maintenance.

For future research, it is recommended to explore more advanced resource load prediction mechanisms using machine learning techniques, such as neural networks or reinforcement learning. This would enable a higher degree of adaptation and further expand the use of FL in more advanced and larger-scale applications. Furthermore, future work could focus on integrating these adaptive algorithms into real-world applications, such as healthcare monitoring systems, industrial IoT, and mobile applications, to further validate their effectiveness and scalability.

Building on these findings, future research should also address several key challenges and opportunities in FL, such as enhancing privacy-preserving techniques, including secure multi-party computation and federated differential privacy, to ensure robust data protection. Integrating reinforcement learning to dynamically adjust training parameters based on real-time device and network conditions could significantly improve FL efficiency. Blockchain technology can improve the security and transparency of FL by providing a decentralized and tamper-proof record of model updates. Exploring the potential of quantum computing to accelerate FL training processes and handle complex computations more efficiently is another promising direction. Finally, leveraging edge and fog computing infrastructures can reduce latency and

improve the responsiveness of FL systems in real-time applications.

In summary, the results of these studies underscore the significant potential of adaptive FL algorithms in decentralized machine learning, particularly in environments where data privacy and resource efficiency are crucial. By addressing the challenges and exploring the future directions outlined above, FL can be further advanced to meet the demands of increasingly complex and resource-constrained applications.

Bibliography

[1] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] M. Selimi, A. Lertsinsrubtavee, A. Sathiaseelan, L. Cerdà-Alabern, and L. Navarro, “Picasso: Enabling information-centric multi-tenancy at the edge of community mesh networks,” *Computer Networks*, vol. 164, p. 106897, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618312787

[3] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria, “Machine learning on mainstream micro-controllers,” *Sensors*, vol. 20, no. 9, 2020.

[4] K. S. Arikumar, S. B. Prathiba, M. Alazab, T. R. Gadekallu, S. Pandya, J. M. Khan, and R. S. Moorthy, “Fl-pmi: Federated learning-based person movement identification through wearable devices in smart healthcare systems,” *Sensors*, vol. 22, no. 4, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/4/1377

[5] A. Farhad, S. Woolley, and P. Andras, “ Federated learning for AI to improve patient care using wearable and IoMT sensors ,” in *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*. Los Alamitos, CA, USA: IEEE Computer Society, Aug. 2021, pp. 434–434. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICHI52183.2021.00071

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[7] K. Yang, T. Jiang, Y. Shi, and Z. Ding, “Federated learning via over-the-air computation,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[8] P. Pinyoanuntapong, P. Janakaraj, P. Wang, M. Lee, and C. Chen, “Fedair: Towards multi-hop federated learning over-the-air,” in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.

[9] F. Freitag, P. Vilchez, L. Wei, C.-H. Liu, and M. Selimi, “Performance evaluation of federated learning over wireless mesh networks with low-capacity devices,” in *Information Technology and Systems*, Á. Rocha, C. Ferrás, A. Méndez Porras, and E. Jimenez Delgado, Eds. Cham: Springer International Publishing, 2022, pp. 635–645.

[10] Z. Qin, G. Y. Li, and H. Ye, “Federated learning and wireless communications,” *IEEE Wireless Communications*, vol. 28, no. 5, pp. 134–140, 2021.

[11] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, “Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data,” *Scientific Reports*, vol. 10, no. 1, p. 12598, Jul 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-69250-1

[12] A. Imteaj, K. Mamun Ahmed, U. Thakker, S. Wang, J. Li, and M. H. Amini, *Federated Learning for Resource-Constrained IoT Devices: Panoramas and State of the Art*. Cham: Springer International Publishing, 2023, pp. 7–27. [Online]. Available: https://doi.org/10.1007/978-3-031-11748-0_2

[13] M. H. Alsharif, A. H. Kelechi, A. Jahid, R. Kannadasan, M. K. Singla, J. Gupta, and Z. W. Geem, “A comprehensive survey of energy-efficient computing to enable sustainable massive iot networks,” *Alexandria Engineering Journal*, vol. 91, pp. 12–29, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016824001091

[14] G. Rjoub, O. A. Wahab, J. Bentahar, R. Cohen, and A. S. Bataineh, “Trust-augmented deep reinforcement learning for federated learning client selection,” *Information Systems Frontiers*, vol. 26, no. 4, pp. 1261–1278, Aug 2024. [Online]. Available: https://doi.org/10.1007/s10796-022-10307-z

[15] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, “Distributed federated learning for ultra-reliable low-latency vehicular communications,” *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2020.

[16] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, “Federated learning for resource-constrained iot devices: Panoramas and state-of-the-art,” 2020. [Online]. Available: https://arxiv.org/abs/2002.10610

[17] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. B. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, S. Y. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, O. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. X. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, pp. 1–210, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:209202606

[18] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, “Federated learning over wireless networks: Convergence analysis and resource allocation,” *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021.

[19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[20] L. Shkurti and M. Selimi, “Baca: Bandwidth and cpu-aware adaptive federated learning for wireless environments,” in *2024 13th Mediterranean Conference on Embedded Computing (MECO)*, 2024, pp. 1–5.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[22] L. Shkurti, M. Selimi, and A. Besimi, “Federatedmesh: Collaborative federated learning for medical data sharing in mesh networks,” in *Collaborative Computing: Networking, Applications and Worksharing*, H. Gao, X. Wang, and N. Voros, Eds. Cham: Springer Nature Switzerland, 2024, pp. 154–169.

[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[24] A. Giuseppi, L. Della Torre, D. Menegatti, and A. Pietrabissa, "Adafed: Performance-based adaptive federated learning," in *Proceedings of the 5th International Conference on Advances in Artificial Intelligence*, ser. ICAAI '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 38–43. [Online]. Available: https://doi.org/10.1145/3505711.3505717

[25] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawit, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. Theertha Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, 2021.

[26] R. Kapoor and D. Sachdeva, *Machine Learning Techniques*, 01 2023.

[27] G. T. Dionisios Sotiropoulos, *Machine Learning Paradigms*. Springer, 10 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-47194-5

[28] J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery, 2016. [Online]. Available: https://books.google.com/books?id=n--oDwAAQBAJ

[29] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, p. 160, Mar 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[31] M. Ekman, *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow*. Addison-Wesley Professional, 2021.

[32] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer Cham, 2023, https://doi.org/10.1007/978-3-031-29642-0.

[33] A. Rahman, M. S. Hossain, G. Muhammad, D. Kundu, T. Debnath, M. Rahman, M. S. I. Khan, P. Tiwari, and S. S. Band, "Federated learning-based ai approaches in smart healthcare: concepts, taxonomies, challenges and open issues," *Cluster Computing*, vol. 26, no. 4, pp. 2271–2311, Aug 2023. [Online]. Available: https://doi.org/10.1007/s10586-022-03658-4

[34] S. Pati, S. Kumar, A. Varma, B. Edwards, C. Lu, L. Qu, J. J. Wang, A. Lakshminarayanan, S.-h. Wang, M. J. Sheller, K. Chang, P. Singh, D. L. Rubin, J. Kalpathy-Cramer, and S. Bakas, "Privacy preservation for federated learning in health care," *Patterns*, vol. 5, no. 7, Jul 2024. [Online]. Available: https://doi.org/10.1016/j.patter.2024.100974

[35] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, p. 119, Sep 2020. [Online]. Available: https://doi.org/10.1038/s41746-020-00323-1

[36] Y. Xu, J. Zhang, and Y. Gu, "Privacy-preserving heterogeneous federated learning for sensitive healthcare data," in *2024 IEEE Conference on Artificial Intelligence (CAI)*, 2024, pp. 1142–1147.

[37] S. Pandya, G. Srivastava, R. Jhaveri, M. R. Babu, S. Bhattacharya, P. K. R. Maddikunta, S. Mastorakis, M. J. Piran, and T. R. Gadekallu, "Federated learning for smart cities: A comprehensive survey," *Sustainable Energy Technologies and Assessments*, vol. 55, p. 102987, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213138822010359

[38] E. X. Neo, K. Hasikin, M. I. Mokhtar, K. W. Lai, M. M. Azizan, S. A. Razak, and H. F. Hizaddin, "Towards integrated air pollution monitoring and health impact assessment

using federated learning: A systematic review," *Frontiers in Public Health*, vol. 10, 2022. [Online]. Available: https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2022.851553

[39] D.-D. Le, A.-K. Tran, M.-S. Dao, K.-C. Nguyen-Ly, H.-S. Le, X.-D. Nguyen-Thi, T.-Q. Pham, V.-L. Nguyen, and B.-Y. Nguyen-Thi, "Insights into multi-model federated learning: An advanced approach for air quality index forecasting," *Algorithms*, vol. 15, no. 11, 2022. [Online]. Available: https://www.mdpi.com/1999-4893/15/11/434

[40] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 91–100.

[41] H. G. Abreha, M. Hayajneh, and M. A. Serhani, "Federated learning in edge computing: A systematic survey," *Sensors*, vol. 22, no. 2, 2022. [Online]. Available: https://doi.org/10.3390/s22020450

[42] A. Mathur, D. J. Beutel, P. P. B. de Gusmão, J. Fernandez-Marques, T. Topal, X. Qiu, T. Parcollet, Y. Gao, and N. D. Lane, "On-device federated learning with flower," 2021. [Online]. Available: https://arxiv.org/abs/2104.03042

[43] A. E. Cetinkaya, M. Akin, and S. Sagiroglu, "A communication efficient federated learning approach to multi chest diseases classification," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 2021, pp. 429–434.

[44] S. Hakak, S. Ray, W. Z. Khan, and E. Scheme, "A framework for edge-assisted healthcare data analytics using federated learning," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 3423–3427.

[45] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2022.

[46] L. Shkurti, M. Selimi, and A. Besimi, "Federatedmesh: Collaborative federated learning for medical data sharing in mesh networks," in *Collaborative Computing: Networking, Applications and Worksharing*, H. Gao, X. Wang, and N. Voros, Eds. Cham: Springer Nature Switzerland, 2024, pp. 154–169.

[47] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 98 423–98 432, 2021.

[48] N. Tabassum, M. Ahmed, N. J. Shorna, M. M. U. R. Sowad, and H. M. Z. Haque, "Depression detection through smartphone sensing: A federated learning approach," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 17, no. 01, p. pp. 40–56, Jan. 2023. [Online]. Available: https://online-journals.org/index.php/i-jim/article/view/35131

[49] B. G. Mohammed and D. S. Hasan, "Smart healthcare monitoring system using iot," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 17, no. 01, p. pp. 141–152, Jan. 2023. [Online]. Available: https://online-journals.org/index.php/i-jim/article/view/34675

[50] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gündüz, "Dopamine: Differentially private federated learning on medical data," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2101.11693

[51] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[52] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1808.03949

[53] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, "A blockchain-orchestrated federated learning architecture for healthcare consortia," 2019. [Online]. Available: https://arxiv.org/abs/1910.12603

[54] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls: A framework for decentralized federated learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[55] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2020.

[56] J. Luo and S. Wu, "Fedsld: Federated learning with shared label distribution for medical image classification," in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, 2022, pp. 1–5.

[57] B. Ankayarkanni, N. K. Pani, M. Anand, V. Malathy, and Bhupati, “P2flf: Privacy-preserving federated learning framework based on mobile fog computing,” *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 17, no. 17, p. pp. 72–81, Sep. 2023. [Online]. Available: https://online-journals.org/index.php/i-jim/article/view/42835

[58] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, “Secure, privacy-preserving and federated machine learning in medical imaging,” *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, Jun 2020. [Online]. Available: https://doi.org/10.1038/s42256-020-0186-1

[59] F. Nikolaidis, M. Symeonides, and D. Trihinas, “Towards efficient resource allocation for federated learning in virtualized managed environments,” *Future Internet*, vol. 15, no. 8, 2023. [Online]. Available: https://doi.org/10.3390/fi15080261

[60] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, “Joint device scheduling and resource allocation for latency constrained wireless federated learning,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2021.

[61] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[62] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, “Adaptive federated optimization,” 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2003.00295

[63] F. Freitag, P. Vilchez, L. Wei, C.-H. Liu, and M. Selimi, “Performance evaluation of federated learning over wireless mesh networks with low-capacity devices,” in *Information Technology and Systems*, Á. Rocha, C. Ferrás, A. Méndez Porras, and E. Jimenez Delgado, Eds. Cham: Springer International Publishing, 2022, pp. 635–645.

[64] K. Li, H. Wang, and Q. Zhang, “Fedtcr: communication-efficient federated learning via taming computing resources,” *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5199–5219, Oct 2023. [Online]. Available: https://doi.org/10.1007/s40747-023-01006-6

[65] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedadapt: Adaptive offloading for iot devices in federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20 889–20 901, 2022.

[66] R. Mishra, H. P. Gupta, G. Banga, and S. K. Das, "Fed-rac: Resource-aware clustering for tackling heterogeneity of participants in federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 7, pp. 1207–1220, 2024.

[67] Y. Gu, Q. Hu, X. Wang, Z. Zhou, and S. Lu, "Fedacs: an efficient federated learning method among multiple medical institutions with adaptive client sampling," in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2021, pp. 1–6.

[68] P. D. Lorenzo, C. Battiloro, M. Merluzzi, and S. Barbarossa, "Dynamic resource optimization for adaptive federated learning at the wireless network edge," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 4910–4914.

[69] M. Kundroo and T. Kim, "Efficient federated learning with adaptive client-side hyperparameter optimization," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 973–974.

[70] J. Shen, N. Cheng, Z. Yin, Y. Fu, W. Xu, and S. Guo, "Ringsfl: An adaptive federated learning system for heterogeneous clients," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 2023, pp. 1–1.

[71] M. N. Hossen, K. Ahmed, F. M. Bui, and L. Chen, "Fedrsmax: An effective aggregation technique for federated learning with medical images," in *2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2023, pp. 229–234.

[72] F. Freitag, P. Vilchez, L. Wei, C.-H. Liu, and M. Selimi, "Performance evaluation of federated learning over wireless mesh networks with low-capacity devices," in *Information Technology and Systems*, Á. Rocha, C. Ferrás, A. Méndez Porras, and E. Jimenez Delgado, Eds. Cham: Springer International Publishing, 2022, pp. 635–645.

[73] J. Zhang, X. Cheng, C. Wang, Y. Wang, Z. Shi, J. Jin, A. Song, W. Zhao, L. Wen, and T. Zhang, "Fedada: Fast-convergent adaptive federated learning in heterogeneous mobile edge

computing environment," *World Wide Web*, vol. 25, no. 5, pp. 1971–1998, Sep 2022. [Online]. Available: https://doi.org/10.1007/s11280-021-00989-x

[74] J. Zhang, Y. Hua, H. Wang, T. Song, Z. Xue, R. Ma, and H. Guan, "Fedala: Adaptive local aggregation for personalized federated learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 11 237–11 244, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26330

[75] L. Luo, C. Zhang, H. Yu, G. Sun, S. Luo, and S. Dustdar, "Communication-efficient federated learning with adaptive aggregation for heterogeneous client-edge-cloud network," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3241–3255, 2024.

[76] Y. Xu, Z. Ma, H. Xu, S. Chen, J. Liu, and Y. Xue, "Fedlc: Accelerating asynchronous federated learning in edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5327–5343, 2024.

[77] A. Imteaj and M. H. Amini, "Fedparl: Client activity and resource-oriented lightweight federated learning model for resource-constrained heterogeneous iot environment," *Frontiers in Communications and Networks*, vol. 2, 2021. [Online]. Available: https://www.frontiersin.org/journals/communications-and-networks/articles/10.3389/frcmn.2021.657653

[78] L. Zhou, Y. He, K. Zhai, X. Liu, S. Liu, X. Ma, G. Ye, and H. Chai, "Fedcada: Adaptive client-side optimization for accelerated and stable federated learning," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.

[79] J. Li, X. Liu, and T. Mahmoodi, "Federated learning in heterogeneous wireless networks with adaptive mixing aggregation and computation reduction," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2164–2182, 2024.

[80] A. Xiong, H. Zhou, Y. Song, D. Wang, X. Wei, D. Li, and B. Gao, "A multi-task based clustering personalized federated learning method," *Big Data Mining and Analytics*, vol. 7, no. 4, pp. 1017–1030, 2024.

[81] L. Ju, T. Zhang, S. Toor, and A. Hellander, "Accelerating fair federated learning: Adaptive federated adam," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1017–1032, 2024.

[82] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 98 423–98 432, 2021.

[83] P. Tam, S. Math, C. Nam, and S. Kim, "Adaptive resource optimized edge federated learning in real-time image sensing classifications," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 10 929–10 940, 2021.

[84] M. A. Hidayat, Y. Nakamura, and Y. Arakawa, "Enhancing efficiency in privacy-preserving federated learning for healthcare: Adaptive gaussian clipping with dft aggregator," *IEEE Access*, vol. 12, pp. 88 445–88 457, 2024.

[85] J. Song, M.-H. Oh, and H.-S. Kim, "Personalized federated learning with server-side information," *IEEE Access*, vol. 10, pp. 120 245–120 255, 2022.

[86] B. Alhalabi, S. Basurra, and M. M. Gaber, "Fednets: Federated learning on edge devices using ensembles of pruned deep neural networks," *IEEE Access*, vol. 11, pp. 30 726–30 738, 2023.

[87] O. Marnissi, H. El Hammouti, and E. H. Bergou, "Adaptive sparsification and quantization for enhanced energy efficiency in federated learning," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4307–4321, 2024.

[88] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[89] L. Ibraimi, M. Selimi, and F. Freitag, "Bepoch: Improving federated learning performance in resource-constrained computing devices," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[90] L. Shkurti and M. Selimi, "Adaptivemesh: Adaptive federate learning for resource-constrained wireless environments," *International Journal of Online and Biomedical*

Engineering (iJOE), vol. 20, no. 14, p. pp. 22–37, Nov. 2024. [Online]. Available: https://online-journals.org/index.php/i-joe/article/view/50559

[91] G. Women and C. M. Center, “Chest x-ray images (pneumonia),” https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia, accessed: 2021-04-28.

[92] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang, “Identifying medical diagnoses and treatable diseases by image-based deep learning,” *Cell*, vol. 172, no. 5, pp. 1122–1131.e9, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0092867418301545