Advanced Decentralized Techniques for Privacy-Preserving Continuous Authentication and Resource Efficiency

by

Mohamad WAZZEH

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, "MARCH 09, 2025"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

FOREWORD

This dissertation examines the topic of continuous user authentication with a focus on privacy-preserving decentralization and resource optimization—an increasingly significant area in cybersecurity and distributed systems. The research outcomes include one magazine article, two published journal papers, and a paper currently under review. The first two chapters provide a thorough introduction and a comprehensive review of the current state of research on continuous authentication and privacy-preserving decentralized methodologies. The subsequent chapters are based on the journal articles, and presented without modification. While each article addresses distinct aspects of the subject, they are intricately linked, collectively forming a cohesive and unified body of work. This dissertation makes a valuable contribution to understanding continuous user authentication, with particular emphasis on privacy preservation and the efficient utilization of resources in decentralized systems.

ACKNOWLEDGEMENTS

I am deeply grateful for the support, guidance, and encouragement I have received throughout this journey, without which this work would not have been possible.

I want to start by sincerely thanking my supervisor, Professor Chamseddine Talhi, for their incredible guidance, expertise, and constant support throughout this research. Their thoughtful feedback, patience, and encouragement have been instrumental in shaping this thesis and helping me bring my work to its best.

I would also like to sincerely thank my co-supervisor, Professor Azzam Mourad, whose technical expertise and constructive criticism have been essential in overcoming many challenges during this research. Their mentorship and valuable advice have greatly enriched my academic experience.

My deep appreciation goes to my thesis committee members, Professor Georges Kaddoum, Professor Kaiwen Zhang, and Professor Benjamin Fung, for their efforts in evaluating my work and providing insightful comments and suggestions. Their contributions have been invaluable in improving the quality of this dissertation.

I'm deeply thankful to my family for always being there for me and supporting me every step of the way. My parents' love and faith in me have been a constant source of strength and motivation. I'm also truly grateful to my friends, whose encouragement and company have made this journey more enjoyable and meaningful.

Lastly, I would like to thank everyone who has contributed to this work, directly or indirectly, and supported me in achieving this academic milestone. Your support means the world to me. Thank you.

Techniques avancées décentralisées pour l'authentification continue préservant la vie privée et l'efficacité des resources.

Mohamad WAZZEH

RÉSUMÉ

Les préoccupations relatives à la confidentialité et la demande croissante pour une authentification continue, sécurisée et efficace ont souligné la nécessité de solutions décentralisées. Cette thèse vise à développer des techniques avancées de préservation de la confidentialité pour l'authentification continue des utilisateurs tout en répondant aux défis liés à l'optimisation des ressources et à la sélection des clients dans les systèmes décentralisés. En abordant des problématiques telles que l'efficacité des ressources, l'évolutivité et la participation sécurisée des clients, cette recherche améliore la praticité et les performances des mécanismes d'authentification décentralisés dans des scénarios réels.

Les contributions de cette recherche se déclinent en trois points principaux :

1. **Développement d'un cadre décentralisé de préservation de la confidentialité :** Ce cadre garantit la confidentialité des données des utilisateurs tout en atteignant une haute précision d'authentification.
2. **Mécanisme de sélection adaptative des clients :** Le mécanisme proposé optimise l'utilisation des ressources et priorise la participation des clients fiables.
3. **Stratégies d'optimisation des ressources :** Ces stratégies améliorent l'efficacité et l'évolutivité du système décentralisé, permettant un fonctionnement fluide dans des environnements et sur des dispositifs variés.

Les approches proposées sont validées par des expériences approfondies utilisant des ensembles de données réels, démontrant des avancées significatives en matière d'utilisation des ressources, de précision d'authentification et d'évolutivité. Ces contributions établissent une base solide pour les systèmes décentralisés de préservation de la confidentialité, ouvrant la voie à des applications pratiques dans les environnements de l'Internet des objets (IoT) et au-delà.

Mots-clés: Apprentissage fédéré, apprentissage scindé, authentification continue, Internet des objets, optimisation des ressources

Advanced Decentralized Techniques for Privacy-Preserving Continuous Authentication and Resource Efficiency

Mohamad WAZZEH

ABSTRACT

Privacy concerns and the increasing demand for secure and efficient continuous authentication have underscored the necessity of decentralized solutions. This thesis aims to develop advanced privacy-preserving techniques for continuous user authentication while addressing resource optimization and client selection challenges in decentralized systems. By tackling issues such as resource efficiency, scalability, and secure client participation, this research enhances the practicality and performance of decentralized authentication mechanisms in real-world scenarios.

The contributions of this research are threefold:

1. **Development of a decentralized privacy-preserving framework:** This framework ensures user data confidentiality while achieving high authentication accuracy.
2. **Adaptive client selection mechanism:** The proposed mechanism optimizes resource usage and prioritizes the participation of reliable clients.
3. **Resource optimization strategies:** These strategies improve the efficiency and scalability of the decentralized system, facilitating seamless operations across diverse environments and devices.

The proposed approaches are validated through extensive experimentation using real-world datasets, demonstrating notable advancements in resource utilization, authentication accuracy, and system scalability. These contributions provide a strong foundation for privacy-preserving decentralized systems, enabling their practical application in Internet of Things (IoT) environments and other domains.

Keywords: federated learning, split learning, continuous authentication, Internet of Things, resource optimization

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BS	Base Station
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
DL	Deep Learning
DP	Differential Privacy
ETS	École de Technologie Supérieure
FL	Federated Learning
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
HE	Homomorphic Encryption
IDS	Intrusion Detection System
IID	Independent and Identically Distributed
IoT	Internet of Things
LR	Learning Rate
MEC	Mobile Edge Computing
ML	Machine Learning
MLP	Multi-Layer Perceptron

MNIST	Modified National Institute of Standards and Technology
NN	Neural Network
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent
SL	Split Learning
SMC	Secure Multi-party Computation
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
UMDAA	University of Maryland Data Acquisition and Analysis

INTRODUCTION

0.1 Motivation

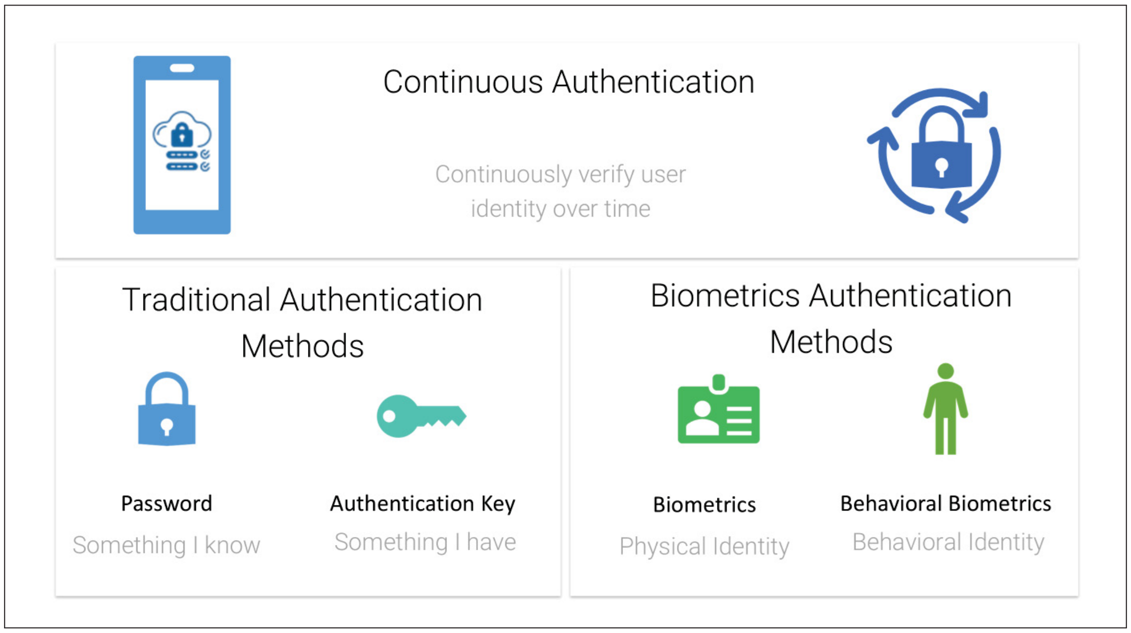
Continuous Authentication
Continuously verify user identity over time
Traditional Authentication Methods
Password
Something I know
Authentication Key
Something I have
Biometrics Authentication Methods
Biometrics
Physical Identity
Behavioral Biometrics
Behavioral Identity

Figure 0.1 Overview of Continuous Authentication and its Comparison with Traditional and Biometric Methods

The rapid evolution of connected devices, including smartphones, IoT devices, and wearables, has created a wealth of opportunities for continuous data-driven applications, ranging from healthcare to smart cities. However, this surge in connected technologies has also heightened the demand for robust and privacy-preserving authentication systems.

Continuous authentication is a promising solution that provides ongoing user identity verification based on behavioral and biometric data, addressing limitations of traditional static methods such as passwords or tokens (Yang, Zhang, Ren & Shen, 2015; Liang, Samtani, Guo & Yu, 2020). These static methods are prone to vulnerabilities like credential theft, session hijacking, and shoulder surfing, especially in resource-constrained IoT environments (Mourad & Jebbaoui, 2015; Jebbaoui, Mourad, Otrok & Haraty, 2015).

Figure 0.1 provides an overview of continuous authentication and its comparison with traditional and biometric authentication methods.

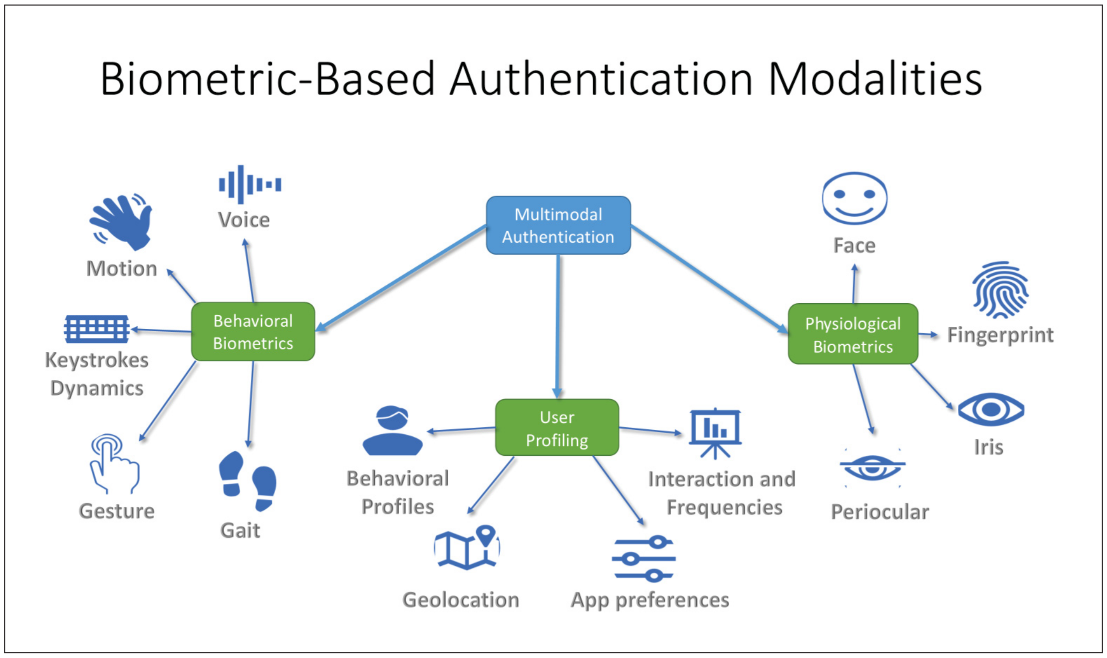

Figure 0.2 Behavioral and Physiological Biometrics in Authentication

Continuous authentication leverages device sensors and behavioral biometrics, such as typing patterns, motion, and interaction habits, to build adaptive user profiles. As shown in Figure 0.2, authentication modalities are categorized into behavioral biometrics (e.g., keystroke dynamics, gait, and voice) and physiological biometrics (e.g., face, fingerprints, and iris). The figure also highlights multi-modal authentication, which combines these approaches for greater accuracy and security, along with user profiling to integrate behavioral patterns and preferences. However, traditional centralized approaches for processing such data introduce critical privacy concerns. Sending raw user data to a central server increases the risk of security breaches and exposure to vulnerabilities, as documented in prior studies on IoT and wearable device security (Arias, Wurm, Hoang & Jin, 2015; Ching & Singh, 2016). In this context, decentralized learning frameworks such as Federated Learning (FL) and Split Learning (SL) have emerged as viable alternatives, enabling collaborative training while maintaining data on local devices to enhance privacy (McMahan, Moore, Ramage, Hampson & y Arcas, 2017b; Vepakomma, Gupta, Swedish & Raskar, 2018).

Despite their advantages, both FL and SL face significant challenges in implementing continuous authentication. The non-IID nature of data, resource heterogeneity across devices, and inefficiencies in client selection can impact model convergence and accuracy. Furthermore, traditional FL approaches often struggle to address the computational constraints of IoT devices, while SL's sequential training methods can introduce bottlenecks, limiting scalability (Wahab, Mourad, Otrok & Taleb, 2021; Gao *et al.*, 2020; Wazzeh *et al.*, 2023).

To address these challenges, this research introduces a novel framework that combines the strengths of FL and SL, enabling efficient and secure continuous authentication. By leveraging behavioral biometrics and client-side training, the proposed framework ensures data privacy while achieving high model accuracy. Additionally, clustering techniques and heuristic client selection algorithms are employed to optimize resource allocation and enhance scalability (Tu, Ouyang, Zhou, He & Xing, 2021; Wazzeh, Ould-Slimane, Talhi, Mourad & Guizani, 2022a). The integration of parallel SL methods further reduces communication overhead, enabling resource-constrained devices to participate effectively in the learning process (Duan, Hu, Deng & Lu, 2022; Gupta & Raskar, 2018).

The motivation behind this work stems from the need to develop robust, scalable, and privacy-preserving solutions for continuous authentication in real-world scenarios. Applications such as healthcare, smart cities, and autonomous systems depend on the secure and efficient operation of IoT devices. By addressing current limitations in decentralized learning and authentication techniques, this research aims to contribute significantly to the state-of-the-art in this domain (Wazzeh, Ould-Slimane, Talhi, Mourad & Guizani, 2022b; Yazdinejad, Parizi, Dehghantanha & Karimipour, 2021).

0.2 Continuous Authentication in Decentralized Systems

Continuous authentication ensures persistent user verification using behavioral and physiological data. Decentralized frameworks such as FL and SL offer privacy-preserving solutions by training models locally without sharing sensitive data. This section explores these methods and their challenges in resource optimization and client management.

0.2.1 Centralized Methods

Centralized authentication methods have been a fundamental aspect of traditional security frameworks, where raw data collected from user devices is transmitted to a centralized server for processing and analysis. These methods enable machine learning (ML) models to be trained on high-performance servers, leveraging the large volumes of data gathered from various users. Cloud service providers, such as Amazon Web Services, Google Cloud, and Microsoft Azure, offer robust infrastructure to deploy and manage ML services and scale computing resources as needed. While effective in terms of computational power and storage capabilities, these approaches inherently raise privacy concerns, as sensitive user data must be directly transferred to external servers for training and inference.

However, this centralized approach poses significant challenges in terms of privacy and security. Data transmitted to a central server is vulnerable to breaches, eavesdropping, and cyberattacks, potentially exposing sensitive user information such as behavioral patterns, payment details, or personal identification data (Feng *et al.*, 2017). Moreover, centralized systems introduce latency due to data transmission over long distances and incur high costs for data transfer (Mourad & Jebbaoui, 2015). These limitations highlight the need for alternative approaches that prioritize user privacy and reduce reliance on centralized infrastructure.

0.2.2 FL Approaches

FL emerged in 2016 as a paradigm shift in distributed machine learning, addressing centralized methods' privacy and scalability issues. FL enables collaborative model training across multiple devices while keeping raw data localized, thus preserving user privacy and reducing communication overhead (McMahan *et al.*, 2017b). In FL, each device trains a local model using its data and transmits only the model updates to a central server for aggregation.

This decentralized approach is particularly beneficial for continuous authentication systems, where behavioral and biometric data must remain confidential. FL has demonstrated its effectiveness in various domains, including IoT security (Rahman, Tout, Talhi & Mourad, 2020), healthcare (Elayan, Aloqaily & Guizani, 2021), and autonomous vehicular/drone networks (Yazdinejad *et al.*, 2021). Despite its advantages, FL faces challenges such as non-IID data distributions, computational constraints of client devices, and communication efficiency, all of which can affect model performance and convergence (Wahab *et al.*, 2021).

0.2.3 SL Approaches

SL offers another decentralized framework, wherein the neural network model is partitioned between clients and a central server. This approach reduces the computational burden on client devices by offloading a portion of the model's training to the server while ensuring that raw data remains on the client side (Vepakomma *et al.*, 2018). SL is particularly advantageous for resource-constrained IoT devices, as it allows clients to train only specific layers of the neural network and share intermediate results with the server.

SL can be employed in parallel or sequential modes, as illustrated in Figure 3.3. In the parallel mode, multiple clients train simultaneously, reducing overall training time and mitigating the inefficiencies of sequential processes. While SL enhances training efficiency and privacy preservation, it faces challenges such as weight divergence and the need for synchronization across clients (Gao *et al.*, 2020; Duan *et al.*, 2022).

Integrating SL with FL presents an opportunity to leverage the strengths of both frameworks, enabling robust and scalable continuous authentication systems. By combining FL's ability to aggregate models with SL's efficient resource utilization, this hybrid approach addresses the limitations of each method and enhances their applicability to real-world scenarios (Wazzeh *et al.*, 2023).

0.2.4 Challenges in Resource and Client Management

The deployment of decentralized learning frameworks for continuous authentication introduces unique challenges in resource allocation and client management. IoT devices vary significantly in their computational capabilities, sensor modalities, and activity patterns, leading to heterogeneity that complicates model training (Tu *et al.*, 2021). Additionally, traditional client selection techniques, which often rely on random or predetermined strategies, fail to account for the dynamic resource constraints of devices, resulting in inefficient learning processes (Wazzeh *et al.*, 2022a).

Addressing these challenges requires advanced clustering and heuristic-based algorithms for client selection. By grouping devices with similar capabilities and optimizing resource allocation based on dynamic constraints, decentralized systems can achieve better scalability and performance (Gupta & Raskar, 2018). These strategies not only enhance training efficiency but also ensure fair participation among clients, enabling the development of robust continuous authentication models for diverse environments.

0.3 Problem Statement

Continuous authentication systems play a vital role in ensuring secure and seamless user verification across IoT and mobile devices. However, implementing such systems in decentralized frameworks like FL and SL faces several challenges that hinder their practical application.

- **Heterogeneous Client Resources:** IoT and mobile devices differ significantly in their computational power, network connectivity, and available resources. Clients with limited capabilities may fail to complete training tasks within the required time, causing delays or forcing the abandonment of training rounds. This inefficiency wastes the resources of other participating clients and increases the number of communication rounds needed to reach the desired model accuracy (Tu *et al.*, 2021).
- **Non-IID Data Distribution:** The data collected across devices is often non-Independent and Identically Distributed (non-IID), which impacts the convergence and accuracy of the global model. Variability in data volume, quality, and activity patterns further complicates the training process (Wahab *et al.*, 2021; Wazzeh *et al.*, 2022a).
- **Client Selection Challenges:** Current client selection methods frequently employ random or static strategies, ignoring resource constraints and data quality. This approach may result in the selection of unreliable clients or underutilization of high-quality participants, negatively affecting model performance and resource allocation efficiency (Wazzeh *et al.*, 2023).
- **Model Training Bottlenecks:** While FL and SL are promising for privacy preservation, their implementation introduces technical limitations. SL's sequential training process increases latency, whereas FL's weight divergence and communication overhead issues create scalability challenges (Vepakomma *et al.*, 2018; Duan *et al.*, 2022).
- **Resource Optimization and Scalability:** Efficient resource allocation is crucial for decentralized learning systems to operate effectively. Current approaches lack advanced strategies to address the dynamic nature of client resource availability, leading to suboptimal system performance and scalability issues (Gupta & Raskar, 2018).

This thesis addresses these challenges by proposing novel approaches to optimize client selection, resource allocation, and decentralized model training, thereby enhancing the efficiency and scalability of continuous authentication systems in real-world scenarios.

0.4 Research Objectives

The primary objective of this thesis is to enhance the efficiency, scalability, and security of continuous authentication systems in decentralized environments. To achieve this overarching goal, we focus on the following sub-objectives:

1. **Developing a Decentralized Framework for Continuous Authentication:** Establishing a robust framework that integrates FL and SL to address the limitations of traditional centralized authentication methods while ensuring user data privacy and security.
2. **Optimizing Client Selection:** Designing advanced client selection mechanisms that account for resource heterogeneity, data quality, and dynamic constraints. These mechanisms aim to improve the training process by prioritizing clients with sufficient computational resources and reliable data contributions.
3. **Enhancing Resource Allocation Strategies:** Proposing heuristic and clustering-based algorithms for efficient resource management. These strategies focus on minimizing communication overhead and ensuring fair resource distribution among participating clients to achieve scalability and performance optimization.
4. **Addressing Non-IID Data Challenges:** Investigating the impact of non-Independent and Identically Distributed (non-IID) data on model accuracy and convergence. This includes developing techniques to mitigate weight divergence and ensure consistent model performance across heterogeneous devices.
5. **Integrating Privacy-Preserving Mechanisms:** Incorporating secure training methodologies, such as FL and SL, to safeguard user data while maintaining high authentication accuracy.

These methods aim to address privacy concerns associated with raw data transmission and centralized storage.

By achieving these objectives, this thesis aims to advance the state-of-the-art in decentralized continuous authentication systems, paving the way for practical applications in IoT, mobile, and resource-constrained environments.

0.5 Contributions and Novelty

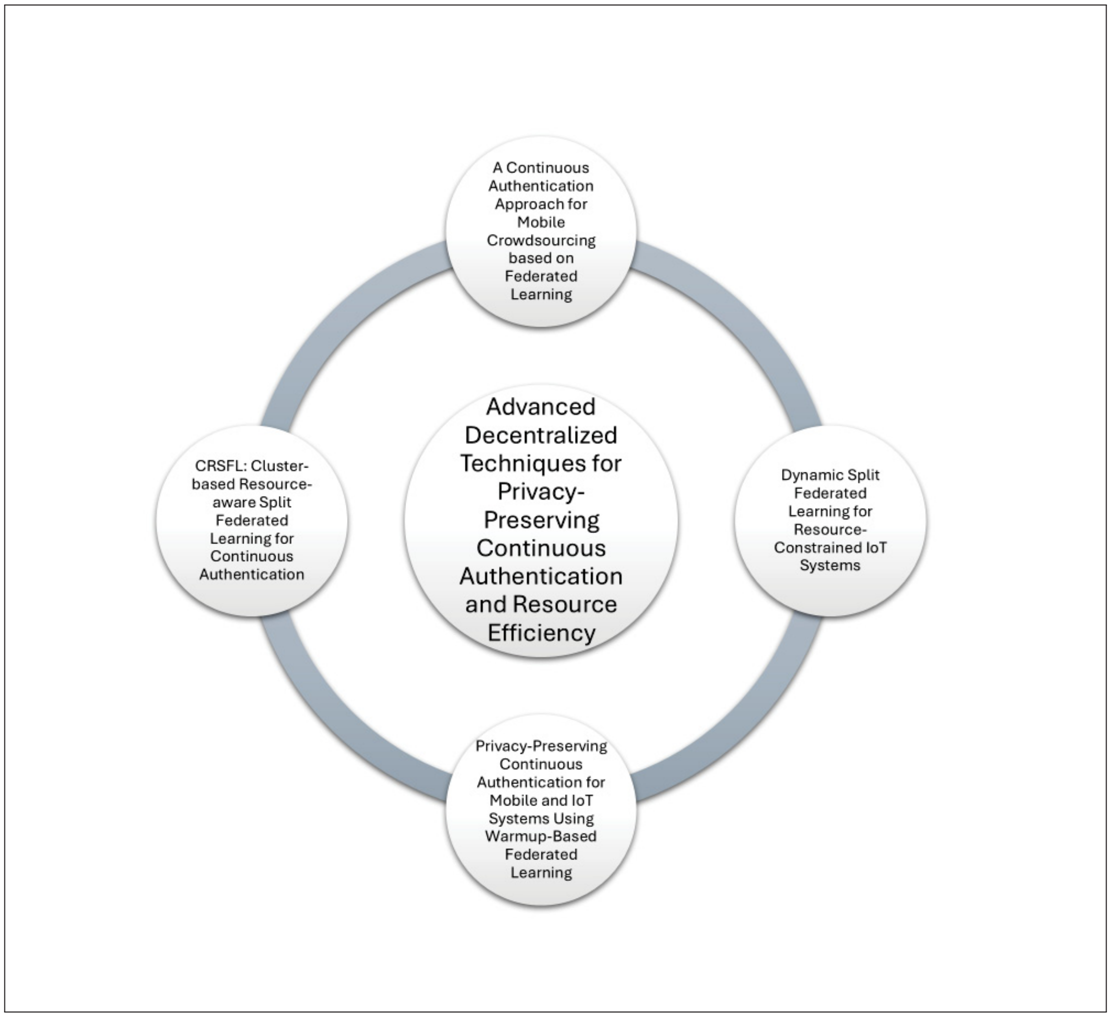
A Continuous Authentication Approach for Mobile Crowdsourcing based on Federated Learning
CRSFL: Cluster-based Resource-aware Split Federated Learning for Continuous Authentication
Advanced Decentralized Techniques for Privacy-Preserving Continuous Authentication and Resource Efficiency
Dynamic Split Federated Learning for Resource-Constrained IoT Systems
Privacy-Preserving Continuous Authentication for Mobile and IoT Systems Using Warmup-Based Federated Learning

Figure 0.3 Overview of the thesis contributions paradigm, showcasing the key research papers and their alignment with the thesis structure

In this section, we enumerate the contributions of this thesis to the field of decentralized continuous authentication while highlighting the novelty. Building on existing decentralized learning paradigms, we address key challenges in privacy preservation, resource optimization, and scalability. Our contributions are summarized below and further illustrated in Figure 0.3 for a visual representation of how the contributions align with the thesis structure.

1. We propose a novel FL framework for continuous authentication that embeds a warmup-based strategy to address the challenges of privacy preservation and non-IID data distributions. Our approach enables mobile and IoT devices to collaboratively train authentication models without sharing raw user data, ensuring user privacy. The initial model is formed using a warmup strategy, where the server generates a baseline model from a small subset of data. This model is then distributed to clients to enhance convergence and boost the performance of local models. We further extend the framework by addressing the non-IID nature of behavioral data, which often causes divergence in federated systems. By integrating a warmup initialization, we achieve more stable and accurate global models. Experimental evaluations, conducted on a composite non-IID dataset derived from MNIST, CIFAR-10, and FEMNIST, demonstrate that the proposed framework significantly improves authentication accuracy, with the warmup model contributing to a 60% performance boost.
Novelty: We are the first to propose a privacy-preserving FL approach tailored for continuous authentication, incorporating a warmup strategy to mitigate the challenges of non-IID data and improve the convergence of federated models. This work advances the state of the art by ensuring user privacy while achieving superior authentication accuracy and stability.

2. We propose a Cluster-Based Resource-aware Split Federated Learning (CRSFL) framework to enhance continuous authentication for IoT and mobile devices. By integrating FL and SL, our approach addresses the challenges of privacy, resource heterogeneity, and efficiency in decentralized machine learning. The CRSFL framework ensures that users' raw data

remains on their devices, preserving privacy while enabling robust model training. To address device heterogeneity, we introduce a clustering methodology that groups devices with similar capabilities, reducing communication overhead and ensuring effective collaboration within clusters. Additionally, our framework employs a heuristic client selection algorithm, leveraging a Genetic Algorithm (GA) to optimize resource allocation based on multiple objectives such as resource availability, event rate, and device history. By combining these features, CRSFL facilitates scalable, efficient, and secure continuous authentication.

Novelty: This study uniquely integrates FL and SL into a unified framework tailored for continuous authentication, introducing a cluster-based approach for managing device heterogeneity and a heuristic client selection algorithm to optimize resource allocation. Our methodology preserves user privacy by retaining data on devices while addressing the challenges of non-IID data distributions, slow training due to resource limitations, and inefficiencies in client participation. Through a novel combination of clustering, resource prediction, and GA-based client selection, CRSFL achieves superior performance, scalability, and efficiency, setting a new benchmark for continuous authentication frameworks in IoT and mobile environments.

3. We propose a novel framework for continuous authentication in Mobile Crowdsourcing (MCS) that leverages FL in combination with transfer learning and warmup techniques. Our approach enables secure and privacy-preserving continuous identity verification by keeping user data localized while collaboratively training a decentralized authentication model. To address the challenges of non-IID data distributions and weight divergence in FL, we incorporate transfer learning to initialize model training with pre-trained knowledge and a warmup strategy to improve convergence. Extensive experiments conducted on real-world datasets validate the effectiveness of our framework, demonstrating significant improvements in authentication accuracy and privacy protection compared to state-of-the-art methods.

Novelty: This study uniquely integrates transfer learning and warmup techniques into an FL framework tailored for continuous authentication in MCS. By addressing the challenges of privacy preservation, non-IID data, and model performance, our approach establishes a robust mechanism for secure and efficient identity verification in real-world applications, outperforming traditional and existing FL-based methods.

4. We propose a novel Dynamic Split Federated Learning (DSFL) architecture that integrates FL and SL to address resource limitations and inefficiencies in IoT environments. DSFL dynamically adapts to the real-time resource availability of client devices, allowing each client to handle a variable number of model layers based on its computational capacity. This flexibility ensures optimal resource utilization, reduces training overhead, and minimizes dropout rates. A Genetic Algorithm (GA) further optimizes client selection, aligning with multiple objectives to enhance system performance. By combining FL's decentralization with SL's model partitioning, DSFL facilitates efficient, privacy-preserving collaborative training across heterogeneous devices.
Novelty: This work uniquely integrates dynamic layer adjustment into an FL framework, enabling IoT devices with varying resource constraints to collaborate effectively. By optimizing computational task allocation with machine learning and employing GA-based client selection, DSFL achieves efficient resource utilization, reduces training time, and enhances system scalability. Comprehensive evaluations using real-world datasets demonstrate that DSFL outperforms existing methods in accuracy, training efficiency, and stability, setting a new benchmark for resource-constrained IoT environments.

These contributions collectively advance the state-of-the-art in decentralized continuous authentication, addressing critical challenges and paving the way for practical implementations in IoT and mobile environments.

0.6 Author's Publications

0.6.1 Journal Publications

In terms of journals, the contribution to the state-of-the-art resulted in a total 3 journal articles. We list them below:

1. **Wazzeh, M.**, Arafeh, M., Sami, H., Ould-Slimane, H., Talhi, C., Mourad, A., & Otrok, H. (2024). CRSFL: Cluster-based Resource-aware Split Federated Learning for Continuous Authentication. *Journal of Network and Computer Applications, 231*, 103987.
2. **Wazzeh, M.**, Ould-Slimane, H., Talhi, C., Mourad, A., & Guizani, M. (2024). A Continuous Authentication Approach for Mobile Crowdsourcing based on Federated Learning. *IEEE Access*.
3. **Wazzeh, M**., Hammoud, A., Mourad, A., Otrok, H., Talhi, C., Dziong, Z., Wang, C.-D., & Guizani, M. (2024). Dynamic Split Federated Learning for Resource-Constrained IoT Systems. Submitted to *IEEE Internet of Things Journal*.

0.6.2 Magazine Publications

The contribution to the state-of-the-art in magazines amounts to a single publication, outlined below:

1. **Wazzeh, M.**, Ould-Slimane, H., Talhi, C., Mourad, A., & Guizani, M. (2022). Privacy-preserving continuous authentication for mobile and IoT systems using warmup-based federated learning. *IEEE Network*, 37(3), 224-230.

0.6.3 Conference Publications

The thesis also resulted in two conference publications that contribute to advancing continuous authentication and resource-aware FL in decentralized systems:

1. **Wazzeh, M.**, Arafeh, M., Ould-Slimane, H., Talhi, C., Mourad, A., & Otrok, H. (2023, October). Towards cluster-based split federated learning approach for continuous user authentication. In *2023 7th Cyber Security in Networking Conference (CSNet)* (pp. 114-118). IEEE.
2. **Wazzeh, M.**, Hammoud, A., Guizani, M., Mourad, A., Otrok, H., Talhi, C., & Wang, C. D. (2024, October). Resource-aware split federated learning for fall detection in the metaverse. In *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 626-631). IEEE.

0.6.4 Collaborative Publications

We also contributed with other scholars to extend and enhance certain components relevant to the scope of this thesis. Our primary role involved writing, supervising, and contributing to the conceptual aspects of these works. The result of these efforts is summarized in the following publications:

1. Arafeh, M., **Wazzeh, M**., Ould-Slimane, H., Talhi, C., Mourad, A., & Otrok, H. (2023, October). Efficient privacy-preserving ML for IoT: Cluster-based split federated learning scheme for non-IID data. In *2023 7th Cyber Security in Networking Conference (CSNet)* (pp. 143-147). IEEE.
2. Sami, H., Hammoud, A., Arafeh, M., **Wazzeh, M.**, Arisdakessian, S., Chahoud, M., ... & Guizani, M. (2024). The metaverse: Survey, trends, novel pipeline ecosystem & future directions. *IEEE Communications Surveys & Tutorials*.
3. Arafeh, M., **Wazzeh, M.**, Sami, H., Ould-Slimane, H., Talhi, C., Mourad, A., & Otrok, H. (2024). Efficient privacy-preserving ML for IoT: Cluster-based split federated learning scheme for non-IID data. Submitted to *Journal of Network and Computer Applications*.

4. Bouldjedri, O., **Wazzeh, M.**, Sami, H., Talhi, C., & Ould-Slimane, H. (2024). Privacy-preserving continuous user authentication using federated learning: A case study. Submitted to *IEEE Network Magazine*.

0.7 Thesis Outline

- **Chapter 1** introduces the foundational concepts relevant to this thesis. It discusses the challenges of continuous authentication, resource utilization, and privacy preservation in IoT and mobile environments. The chapter also provides an overview of FL, SL, and their integration in addressing these challenges.
- **Chapter 2** presents the work of *Privacy-preserving continuous authentication for mobile and IoT systems using warmup-based FL*. This chapter focuses on a novel FL framework that ensures privacy preservation during continuous authentication for mobile and IoT devices. It introduces a warmup-based strategy to address non-IID data issues, significantly improving model accuracy and convergence.
- **Chapter 3** presents the work of *CRSFL: Cluster-based Resource-aware Split FL for Continuous Authentication*. This chapter introduces the CRSFL framework, which combines FL and SL with transfer learning to address resource heterogeneity and privacy concerns. It incorporates clustering and heuristic client selection to enhance resource utilization and scalability in IoT environments.
- **Chapter 4** presents the work of *A Continuous Authentication Approach for Mobile Crowdsourcing based on FL*. This chapter focuses on enabling continuous authentication in Mobile Crowdsourcing (MCS) using FL. It employs transfer learning to enhance model initialization and addresses privacy and non-IID data challenges, demonstrating improved accuracy and efficiency in real-world MCS applications.
- **Chapter 5** presents the work of *Dynamic Split FL for Resource-Constrained IoT Systems*. This chapter introduces the DSFL architecture, which dynamically adjusts client model

layers based on real-time resource availability. The chapter highlights how DSFL integrates GA-based client selection to optimize resource utilization, minimize training overhead, and enhance scalability and system performance in resource-constrained IoT environments.

CHAPTER 1

PRELIMINARIES

1.1 Background

This chapter provides a comprehensive overview of the foundational concepts, methodologies, and technical challenges that underpin the research in this thesis. Inspired by well-established work in FL and related distributed machine learning paradigms, we begin by introducing the fundamentals of FL and SL. We detail their core architectures, standard protocols, and essential problem formulations relevant to continuous authentication scenarios. Following this, we examine how the unique characteristics of IoT and Mobile Crowdsourcing (MCS) environments influence the design and deployment of FL/SL-based continuous authentication frameworks. In particular, we highlight critical technical challenges such as handling non-Independent and Identically Distributed (non-IID) data distributions, coping with heterogeneous resource capacities, and ensuring stringent privacy and security requirements. Additionally, we discuss the role of advanced optimization, aggregation, and transfer learning techniques to improve model performance and convergence. The concepts and insights introduced here serve as the foundational layer for the subsequent chapters, wherein we propose, analyze, and evaluate novel solutions and frameworks for continuous authentication in modern distributed systems.

1.2 Literature Review

This chapter consolidates the existing literature on continuous authentication, FL, split learning SL, transfer learning, and distributed learning techniques in resource-constrained IoT environments. The review highlights the evolution of continuous authentication systems, the integration of FL and SL to preserve privacy while addressing non-IID data challenges, and strategies to optimize client selection and resource allocation.

1.2.1 Continuous Authentication

Continuous authentication verifies users based on behavioral data collected from their interactions with devices. Early studies focused on centralized approaches where sensor data is transmitted to an external server to create unique behavioral profiles. For example,(Lee & Lee, 2017) combined smartphone and wearable sensor readings to achieve 98.1% accuracy, while (Zhu *et al.*, 2019) proposed the *RiskCog* model using accelerometer, gyroscope, and gravity sensor data to reach accuracies of 93.8% and 95.6% for steady and moving users, respectively. The work in (Sánchez, Maimó, Celdrán & Pérez, 2021) extended these methods to multi-device scenarios by leveraging a hybrid Mobile Edge Computing (MEC) and Cloud Computing framework, achieving 99.02% accuracy. However, centralizing raw data raises significant privacy concerns (Rahman *et al.*, 2020).

1.2.2 Federated Learning in Continuous Authentication

Federated Learning offers an alternative by training models locally on user devices and transmitting only model updates rather than raw data. This approach preserves privacy and mitigates the risks associated with centralized data storage. Recent works (Wazzeh *et al.*, 2022a,b) have demonstrated that FL, augmented by warm-up strategies and transfer learning, can effectively handle the unique challenges of continuous authentication. However, continuous authentication data are inherently non-IID, as each client typically possesses a single, unique label. This divergence in local data distributions often leads to weight divergence during model aggregation (Monschein *et al.*, 2021; Yazdinejad *et al.*, 2021; Feng, Wang, Liu, Yang & Wang, 2024), necessitating additional strategies to maintain global model stability.

1.2.3 Integration of Split Learning with Federated Learning

To further alleviate the challenges of non-IID data and limited client resources, hybrid approaches that integrate Split Learning with Federated Learning have been proposed. In Split Learning, the model is partitioned between the client and the server, thereby reducing the computational burden on the client side. Works such as (Thapa, Arachchige, Camtepe & Sun, 2022) and (Arafeh *et al.*, 2023b) utilize split federated learning (SFL) to enable parallel client-side training. Other studies (Samikwa, Di Maio & Braun, 2022; Wu *et al.*, 2023; Han, Bhatti, Lee & Moon, 2021) focus on optimizing client selection and resource allocation within SL frameworks. Notably, (Oza & Patel, 2021b) and (Wazzeh *et al.*, 2023) explore FL and SL integration for continuous authentication, though most approaches do not fully address the dynamic nature of client resources.

1.2.4 Transfer Learning in Authentication

Transfer learning has emerged as a valuable tool in continuous authentication, particularly when training data are limited. By transferring knowledge from models pre-trained on related tasks, systems can achieve higher accuracy and robustness with reduced data requirements (Weiss, Khoshgoftaar & Wang, 2016). Recent studies (Singh, Dhanaraj & Sharma, 2024; Kong, Lu, Yu, Chen & Tang, 2020; He, Ma, Tu & Zhang, 2022) have successfully applied transfer learning to enhance authentication models, demonstrating improved performance even in cross-domain scenarios.

1.2.5 Distributed Learning in Resource-Constrained Environments

With the rapid evolution of IoT devices, addressing the computational and resource constraints inherent to such environments has become critical. Distributed learning approaches have been extensively studied to optimize training processes and reduce overhead. For instance, Dynamic Federated Learning (DFL) (Samikwa, Di Maio & Braun, 2024) and cluster-based resource-aware split federated learning (Wazzeh *et al.*, 2024b,a) employ filtering and clustering strategies based on client resource capacities. These approaches aim to minimize training latency and ensure efficient resource allocation, though challenges such as fixed split points and client idle times remain. In related domains like fall detection, various machine learning models (e.g., LSTM-RNN, CNN, Decision Trees, k-NN) have been tested to optimize sensor usage and detection algorithms (Bharathkumar, Paolini & Sarkar, 2020; Hussain *et al.*, 2019;

Chelli & Pätzold, 2019), further underscoring the need for tailored solutions in resource-limited settings (Hemmatpour, Ferrero, Montrucchio & Rebaudengo, 2019).

1.2.6 Summary

In summary, while centralized continuous authentication systems have demonstrated high accuracy, they compromise user privacy by transferring raw sensor data to external servers. Federated Learning and its hybrid integration with Split Learning provide promising avenues to address these privacy concerns. Nevertheless, the non-IID nature of continuous authentication data and the resource limitations of IoT devices present ongoing challenges. Transfer learning and dynamic, resource-aware client selection emerge as critical components to enhance model robustness and convergence. Our work builds on these foundations by developing advanced client selection and resource optimization strategies within a hybrid FL and SL framework tailored for continuous authentication.

1.3 Preliminary: Architecture and Design

1.3.1 FL and SL Overview

FL adopts a decentralized model training paradigm wherein each participating client (e.g., a smartphone, IoT sensor, or wearable device) trains a local model on its private data and communicates only the resulting updates back to a central server for aggregation. This architecture alleviates privacy concerns inherent in centralized data collection by ensuring that raw user data remains on-device. In contrast, SL divides a deep neural network across the client and the server. The client trains the initial layers and forwards intermediate representations (smashed data) to the server, which trains the remaining layers. By never transmitting raw data, SL enhances privacy, and by offloading complex computations to the server, it reduces the client's resource burden.

FL and SL aim to address data privacy, communication constraints, and scalability challenges, making them naturally suitable for continuous authentication tasks in distributed IoT ecosystems. Figure 1.1 illustrates the Federated Split Learning architecture, which combines the principles of FL and SL. The figure demonstrates how multiple clients collaborate with a central server by training the initial layers of their respective models locally and sending smashed data to the server. The server completes the model training and computes the loss, enabling privacy preservation and reducing computational overhead on resource-constrained devices.

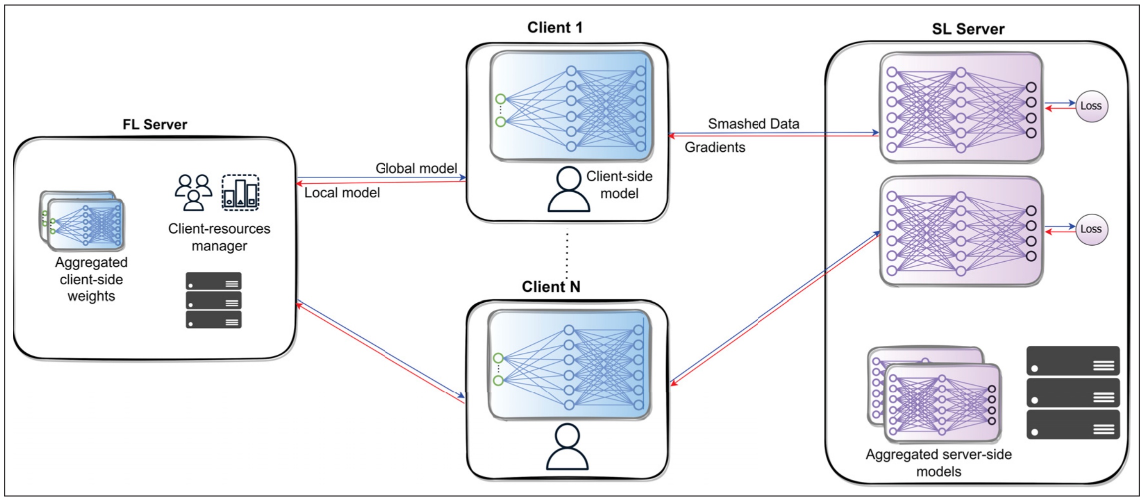

Figure 1.1 Federated Split Learning architecture showcasing the collaborative training process between multiple clients and a server

1.3.2 FL Lifecycle

The standard FL lifecycle involves multiple communication rounds:

1. **Client Selection:** The central server chooses a subset of clients based on availability, connectivity, or resource conditions.
2. **Model Distribution:** The selected clients download the current global model parameters from the server.
3. **Local Training:** Each client refines the global model using its on-device data for a set number of epochs or mini-batch updates.

4. **Update Upload:** The clients send their updated model parameters (gradients or weights) to the server.
5. **Model Aggregation:** The server aggregates received updates (e.g., via weighted averaging) to form a new global model.

This iterative process continues until the global model converges or meets predetermined performance criteria. However, the inherent non-IID nature of user data, varying device capacities, and communication bottlenecks often complicate convergence and degrade model quality.

1.3.3 SL Lifecycle

The standard SL lifecycle involves splitting the model between the client and the server, with the following key steps:

1. **Model Partitioning:** The neural network model is divided into two segments—one part is deployed on the client device, and the other resides on the central server.
2. **Forward Propagation:** The client performs forward propagation using its portion of the model and sends the intermediate activations (cut-layer outputs) to the server.
3. **Server Computation:** The server continues forward propagation with its segment of the model and computes the loss based on the received activations and ground truth labels.
4. **Backward Propagation:** The server calculates the gradients for its model segment and sends the gradients of the cut-layer or smashed data back to the client.
5. **Client Updates:** The client uses the received gradients to update its portion of the model via backpropagation, completing the training cycle.

This process is repeated until the model converges or achieves a desired performance level. SL is particularly advantageous for resource-constrained devices since clients only train a fraction of the model. However, challenges such as synchronization delays, communication overhead, and weight divergence between client and server segments can impact scalability and efficiency.

1.3.4 Problem Formulation for Continuous Authentication

Continuous authentication relies on persistent user verification, ensuring only the rightful owner accesses the device over extended sessions. Integrating FL or SL into this setting involves treating continuous authentication as a classification or verification task distributed across many devices. Each device's local data (e.g., sensor readings, behavioral biometrics, or application usage patterns) forms a unique, often skewed dataset. Instead of relying on isolated models per device, a global authentication model is collaboratively trained across devices to leverage shared patterns while preserving personalized adaptations. This global model benefits from diverse user interactions, making it more resilient to unseen variations and better generalized across different conditions. The key challenge is to achieve high accuracy and low latency under strict resource constraints while preserving user privacy by preventing the centralization of sensitive biometric or behavioral data.

1.4 Technical Challenges and Research Fields

The decentralized, privacy-preserving nature of FL and SL addresses key limitations of centralized continuous authentication mechanisms. While centralized systems often face privacy risks, scalability issues, and resource constraints, FL and SL offer innovative solutions by decentralizing data processing and enhancing privacy through on-device learning or split architecture designs. However, implementing these decentralized approaches introduces its own set of challenges that must be carefully managed. These challenges can be categorized into five critical areas, as illustrated in Figure 1.2:

- **Privacy:** FL and SL address privacy concerns in centralized systems by keeping user data on-device or splitting models. However, risks such as private data leakage through model updates or adversarial inference still require robust mitigation strategies.
- **Accuracy:** Decentralized approaches reduce inaccuracies caused by centralized data biases but face challenges from non-IID data distributions, requiring methods to ensure consistent model performance across diverse environments.

- **Adaptability:** FL and SL improve adaptability by continuously updating models without centralized retraining. However, managing behavioral drift, where user patterns evolve, remains critical.
- **Security:** While decentralized architectures reduce some attack surfaces of centralized systems, other threats like model poisoning, backdoor triggers, and inference-based data reconstruction demand advanced defensive measures.
- **Resources:** By distributing computation across devices, FL and SL alleviate some resource constraints of centralized systems. Yet, managing computation, storage, and communication overheads for resource-limited devices is an ongoing challenge.

These challenges underscore the need for innovative strategies that balance privacy, security, adaptability, and resource efficiency in FL and SL frameworks, enabling their effective deployment in continuous authentication systems.

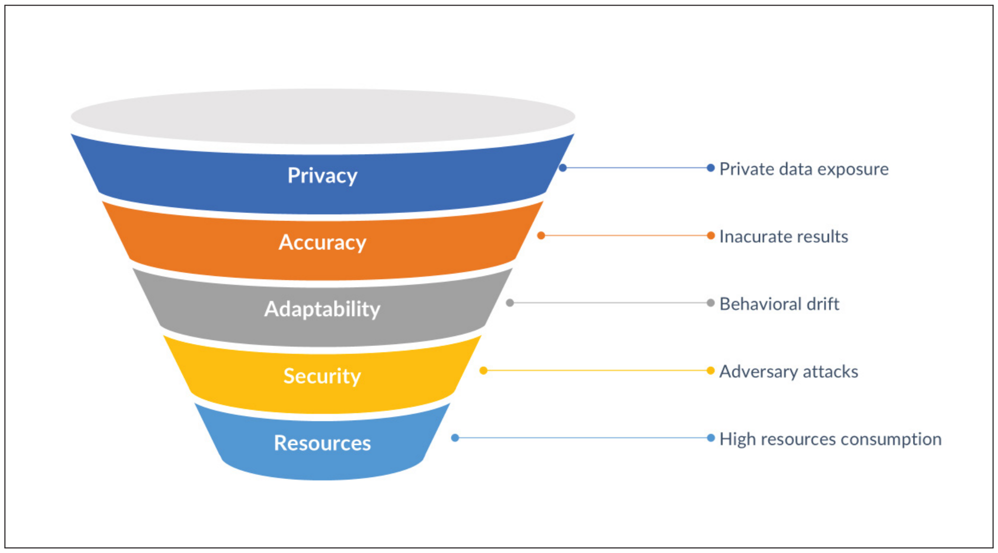

Figure 1.2 Key challenges in continuous authentication systems using FL and SL, categorized by privacy, accuracy, adaptability, security, and resource constraints

1.5 FL and SL System Model and Design Aspects

1.5.1 Communication Overhead

Communication overhead remains a critical bottleneck in FL and SL. Large models and frequent parameter exchanges can overwhelm network resources, leading to latency and energy drain. Strategies to mitigate this include model compression (pruning, quantization), sparsification of gradients, and selective parameter updates where only the most informative or salient weights are transmitted. Additionally, hierarchical FL architectures (e.g., involving edge servers) can reduce round-trip times, and advanced coding schemes can enhance communication efficiency.

1.5.2 Client Selection and Resource Heterogeneity

Clients differ significantly in computation power, memory capacity, bandwidth availability, and battery life. Naive random selection can produce stragglers—devices that slow down training due to limited resources—and lead to inefficient resource usage. Heuristic or optimization-based client selection frameworks, such as those employing Genetic Algorithms (GAs) or reinforcement learning, can ensure that only capable and reliable clients contribute to the global model. This leads to balanced workloads, reduced idle times, and improved model quality and convergence speed.

1.5.3 Optimization, Aggregation, and Transfer Learning Techniques

Robust optimization methods and efficient aggregation rules are necessary to combine heterogeneous local updates into a cohesive global model. Beyond simple averaging (FedAvg), more sophisticated methods weigh updates according to data quality, client reliability, or model divergence. Transfer learning and warmup strategies offer additional advantages by initializing client models with pre-trained weights or representative data subsets. Such initializations reduce variance in local updates, speed up convergence, and improve accuracy in challenging continuous authentication scenarios.

1.5.4 Handling Non-IID Data and Warmup Strategies

In continuous authentication, data distributions are inherently non-IID due to differing user behaviors, device usage patterns, and environmental conditions. Warmup phases use a small, representative dataset or early training rounds to align model weights, mitigating the effects of non-IID distributions. Coupled with partial data sharing, synthetic data generation, or knowledge distillation techniques, warmup strategies can significantly boost performance and stability.

1.6 Application Areas of Continuous Authentication with FL and SL

Mobile crowdsourcing frameworks aggregate data from numerous personal and IoT devices for tasks like human activity recognition, environmental sensing, or user authentication. FL and SL enable the building of robust global authentication models without centralizing sensitive data. This is crucial for compliance with data protection regulations and user trust, as well as for maintaining system performance under network variability and mobility. Furthermore, continuous authentication supported by FL/SL can benefit applications like smart cities, where users interact with public kiosks and IoT services, or autonomous vehicles authenticating drivers continuously. Industrial IoT and edge computing scenarios, such as secure industrial control systems, also demand robust, privacy-preserving authentication solutions at scale.

1.7 Privacy and Security Considerations

While FL and SL substantially reduce data exposure, model updates can still inadvertently leak information about the underlying data. Privacy-enhancing methods, including differential privacy (adding noise to updates), secure aggregation (cryptographic protocols ensuring no single party sees raw updates), and homomorphic encryption (enabling computations on encrypted data), serve as critical safeguards.

Adversaries may attempt model poisoning (injecting malicious updates to degrade model performance), backdoor attacks (embedding triggers that produce incorrect authentication decisions), or inference attacks (reconstructing sensitive data from model parameters). Defenses range from robust aggregation rules that minimize the impact of outliers to anomaly detection systems that identify suspicious updates, and reputation-based schemes that penalize consistently unreliable clients.

Balancing strong privacy guarantees, robust security, and high model performance is a delicate task. Overly aggressive privacy measures may degrade model accuracy, while lenient security can risk data breaches. This interplay influences the algorithmic designs, parameter choices, and deployment strategies, guiding the frameworks and solutions proposed in this thesis.

1.8 Resource Management in FL and SL

Resource management ensures that continuous authentication models can be trained efficiently and effectively in realistic, constrained environments. Techniques include adaptive layer splitting, where the cut layer between client and server changes dynamically based on resource availability; and dynamic client assignment, where the server intelligently selects clients to minimize training time and energy consumption. Hierarchical processing approaches—e.g., training partial models at the network edge and aggregating them at the cloud—further improve scalability and responsiveness.

1.9 Summary

This chapter established the theoretical and conceptual foundation for continuous authentication using FL and SL in diverse IoT and MCS environments. We discussed the FL/SL architectures, lifecycle protocols, and problem formulations, emphasizing continuous authentication's unique requirements. We explored technical challenges—communication overhead, client heterogeneity, non-IID data, privacy, and security—and surveyed application domains ranging from personal IoT devices to healthcare and beyond. With these insights in hand, the subsequent chapters detail the proposed frameworks, algorithms, and empirical evaluations that push the boundaries of continuous authentication capabilities in real-world decentralized settings.

CHAPTER 2

PRIVACY-PRESERVING CONTINUOUS AUTHENTICATION FOR MOBILE AND IOT SYSTEMS USING WARMUP-BASED FEDERATED LEARNING

Mohamad Wazzeh[1] , Hakima Ould-Slimane[2] , Chamseddine Talhi[1] , Azzam Mourad[3,4] , Mohsen Guizani[5]

[1] Department of Software and IT Engineering, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Mathematics and Computer Science, Université du Québec à Trois-Rivières, 3351 boulevard des Forges, Trois-Rivières, Québec, Canada G8Z 4M3
[3] Department of Computer Science and Mathematics, Lebanese American University, Chouran, Beirut 1102 2801, Lebanon
[4] Science Division, New York University Abu Dhabi, Saadiyat Island, Abu Dhabi, United Arab Emirates
[5] Mohamed Bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates

Paper published in IEEE Network, August 2022

2.1 Abstract

Continuous authentication for mobile devices acknowledges users by studying their behavioural interactions with their devices. It provides an extended protection mechanism that supplies an additional layer of security for smartphones and Internet of Things (IoT) devices and locks out intruders in cases of stolen credentials or hijacked sessions. Most of the continuous authentication efforts in the literature consist of collecting behavioural, and sensory data from users and extracting statistical patterns through adopting various Machine Learning (ML) techniques. The main drawback of these approaches is their heavy reliance on acquiring the users' personal data, which exposes the latter's privacy. To address this limitation, we introduce a novel Federated Learning (FL) based continuous authentication mechanism for mobile and IoT devices. Our approach preserves the users' privacy by allowing each individual to locally train an ML model that captures his/her behaviour and then shares the model weights with the server for global aggregation. An extended scheme with a warmup FL approach for continuous

authentication is proposed. Performance evaluation is done with a unique non-IID dataset built from three well-known datasets MNIST, CIFAR-10, and FEMNIST. The extensive experimental results show a major accuracy increase in user authentication.

2.2 Introduction

Existing mobile and IoT authentication methods depend on the user to input a password, pin, match a pattern or use Two-Factor authentication methods to access the device. However, these techniques provide undefined access over the system once the session starts. In parallel, the traditional methods cannot implicitly preserve active sessions without repeatedly interrupting the user's workflow to grant him/her access again. Therefore, this approach is not practical in scenarios where user identity needs to be continuously verified. In this context, continuous authentication has been proposed to authenticate the individual's identity who is trying to gain access over a system or entry point on an ongoing, real-time basis. Such a process begins with collecting user behavioural data from device sensors and sending it to an external server, which is processed using ML techniques. The ML trained from the user data can continuously evaluate the interaction behaviour with the device and predict the user's identity as a background service without interrupting the user's activities.

Continuous authentication is achievable using built-in smartphone and IoT hardware (e.g., accelerometer, camera, microphone, etc.) and software (e.g., application preferences, interaction and frequencies, etc.). Similar to smartphones, IoT devices ranging from smart homes, smartwatches, heart monitors, security cameras, automobiles and others are equipped with hardware and software capability making collecting and processing user behavioural data feasible. Such devices are subject to malicious attacks that could compromise the privacy and security of users' data or may even lead to physical harm. Therefore, providing a secure and private environment for the IoT devices is vital to maintaining the privacy and integrity of the device's legitimate user. Traditional continuous authentication approaches (Lee & Lee, 2017; Zhu *et al.*, 2019) collect user behavioural data and transmit it to an external server. Such data is subjected to attacks and data leakage, implicating privacy and security concerns over the transmitted data.

Moreover, our objective is to deliver a privacy-preserving continuous authentication framework that allows users to verify user behavioural data continuously. Therefore, we studied the literature related to protecting private user data, such as the work of Google in (McMahan, Moore, Ramage, Hampson & y Arcas, 2017a), where the authors introduced a distributed machine learning framework known as federated learning. They showed how a collection of models could be used instead of raw data to generate a global predictive model. Therefore, users' ML models are sent to a central server, where an aggregation method produces a robust global model for user authentication. However, adopting a FL technique for continuous authentication imposes additional problems that need to be addressed.

The behavioural biometric data collected for continuous authentication are unique to each user. Each user collects data representing his distinctive behavioural identity while interacting with a mobile or IoT device and has a unique class or label. Such collected data might also have different sample sizes and vary between users. Thus, the data are not independent nor identically distributed (non-IID). In parallel, FL shows performance degradation when exposed to non-IID data (Wahab *et al.*, 2021; AbdulRahman *et al.*, 2020b; AbdulRahman, Tout, Mourad & Talhi, 2020a). The accuracy degradation is due to local models' weight divergence. The users' local models have the same initial weights at the initialization phase on the server-side and converge to different models due to the unique non-IID data problem in continuous authentication.

To the best of our knowledge, only two research attempts in (Oza & Patel, 2021a; Monschein *et al.*, 2021) have discussed the feasibility of a FL approach for continuous authentication. However, both approaches did not take advantage of the full potential of FL since private user data are still shared to an external server on both approaches. In this context, this paper investigates a novel FL approach for continuous authentication aiming to preserve the raw user data on mobile and IoT devices. Moreover, we tackle the problem of non-IID by introducing a warmup architecture that is inspired by the work of (Zhao *et al.*, 2018). We devise a scheme that generates a warmup model from a small portion of users' data. The server uses this data to create an initializer model shared with the clients. Later in the experiment, we show that an initial warmup model created on the server-side can help boost the accuracy of the models by 60%. We summarize the contributions of this paper as follows:

- Develop a novel Federated Learning approach for continuous authentication on mobile and IoT devices that allows preserving the user's private data, including the initial set.
- Devise a warmup strategy to boost the model weights of the clients while increasing the performance for authentication models and user verification.

The rest of the paper is organized as follows. Section II shows the related work done in continuous authentication and federated learning. Section III explains the framework we are using and introduces our warmup strategy. Section IV shows the details of the experiments we performed alongside the datasets we used and presents the experimental results. Finally, Section V concludes our work and discusses possible future enhancements to our framework.

2.3 Literature Review

This section reviews the literature on continuous authentication and its prospects in a federated learning context.

Continuous authentication is authenticating users by leveraging their behavioural data based on their interactions with their devices. This data is used to create a unique behavioural profile for the user. Recent literature has studied the usage of continuous authentication on mobile and IoT devices and revealed the potential of such an approach.

Lee et al. (Lee & Lee, 2017) suggested using a combination of sensors' readings from the smartphone and other wearable devices of the user to improve the authentication accuracy. The average accuracy achieved is 98.1%, FRR of 0.9%, and FAR of 2.8% when combining smartphone and smart-watches users' data on a central server. Zhu et al. (Zhu *et al.*, 2019) proposed a support vector machine-based model, "RiskCog," which can authenticate users by collecting sensor data from the smartphone and wearable devices. The data is collected from accelerometer, gyroscope, and gravity sensors. The authors claimed that such sensor data is not sensitive to identifying individuals. They were able to get an average system accuracy of 93.8% and 95.6% for steady and moving users, respectively. In a different study by Sanchez et al. (Sánchez *et al.*, 2021), the authors applied a study to multi-devices rather than one device. They used a hybrid approach that combines the Mobile Edge Computing (MEC) and Cloud Computing paradigms. Model training and evaluation were done in the cloud, and they sent the authentication results to the users. They achieved 99.02% accuracy for the authentication of the users.

The research conducted on mobile devices has the main target of continuously protecting the device owner's privacy and security, keeping in mind the importance of security and privacy of the data owned by this user. Since sacrificing transmitting sensor data to the cloud has cons, it is stored in a third-party server subjected to attacks and data leakage. Mobile device resources have been increasing in processing, storage, and memory. Therefore, keeping the device's data to process and evaluate the continuous authentication model has been made possible (Rahman *et al.*, 2020). Limited studies have been done on the usage of FL in the context of continuous authentication. The authors in (Oza & Patel, 2021a) claimed to use FL and split learning. They used a VGG16 pre-trained model on the VGGFace dataset and sent the mean and variance of the data from the users to server, where they applied their algorithm in one round between the clients and server. However, FL works by sending model weight instead of user data. Sending mean and variance of users' data should be studied to determine if malicious attackers could compromise the data. Another recent study in (Monschein *et al.*, 2021) discusses a peer-to-peer FL environment for continuous authentication. The authors discuss FL usage between two or more servers to generate a collaborative ML model. However, the users already share data with each server/entity. Therefore the user's privacy is not preserved on his device. To the best of our knowledge, none of the other works in the literature above have investigated the feasibility of

FL for continuous authentication to address the privacy and security concerns or addressed the Non-IID limitations imposed by adopting such an approach.

2.4 Proposed Framework

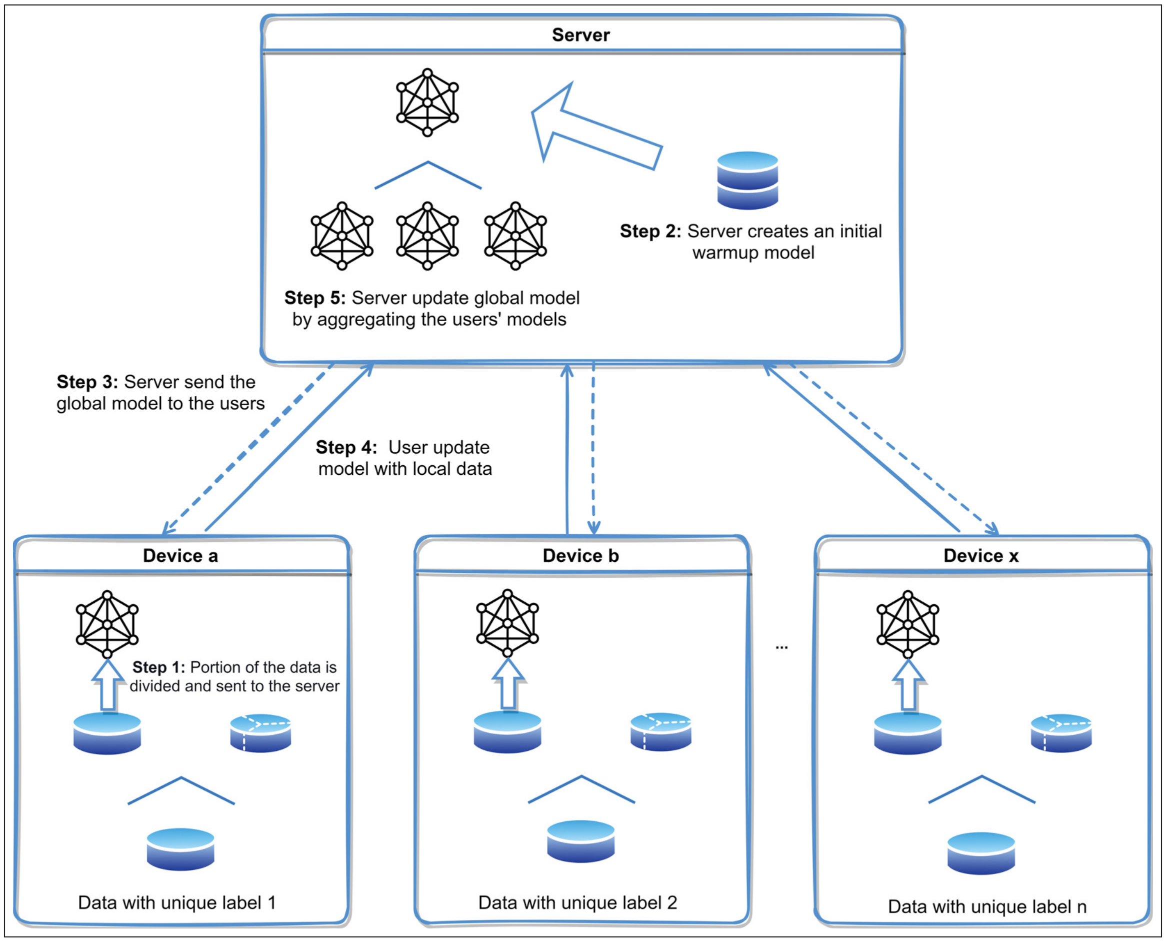
Server
Step 2: Server creates an initial warmup model
Step 5: Server update global model by aggregating the users' models
Step 3: Server send the global model to the users
Step 4: User update model with local data
Device a
Device b
Device x
Step 1: Portion of the data is divided and sent to the server
...
Data with unique label 1
Data with unique label 2
Data with unique label n

Figure 2.1 Federated Learning Warmup-Based Architecture for Continuous Authentication

This section illustrates our continuous authentication approach using warmup-based federated learning. We call this technique a warmup learning approach, as we use pre-trained weights to initiate the federated learning rounds.

Security and privacy are essential factors in the continuous authentication framework. Moreover, the European Union enforced General Data Protection Regulation (GDPR) (AbdulRahman *et al.*, 2020b), which posed strict regulations regarding data sharing and user privacy-preserving techniques, creating new challenges for the data collection of sensor data for the machine learning techniques.

Federated learning is a decentralized collaborative machine learning that can be explored as a potential solution for privacy-preserving individuals' data. Adopting a FL technique can maintain the privacy of the user's data from being transmitted to an external server by sharing model weights instead of raw data. The shared models are further aggregated to increase the learning process between the clients and produce a global model that is shared back with the clients within several rounds until model convergence.

The classical Federated Averaging (FedAvg) algorithm created by Google (McMahan *et al.*, 2017a) relies on the data to be in an IID form. Therefore, directly applying the algorithm to our continuous authentication problem will not yield good results. The FL approach for such unique non-IID data is explored in a few studies (Oza & Patel, 2021a; Zhao *et al.*, 2018; Wahab *et al.*, 2021), in which users have only their labels, and the models generated from each device lack the negative class data of other users. The unique non-IID problem in continuous authentication seems unrealistic in other fields. However, as reported in (Wahab *et al.*, 2021), such non-IID data distribution is highly challenging since the classical FedAvg failed to achieve an optimal model convergence due to the data distribution and the complexity of the problem itself. Moreover, in (Zhao *et al.*, 2018), the authors showed that the accuracy reduction of models is attributed to model weight divergence, as there is an association between weight divergence and the skewness of the data. The authors proposed a data-sharing strategy based on a warmup model that could help improve the accuracy of the FedAvg algorithm with non-IID data. The proposed architecture is by creating a small subset of the data, which is shared globally between the user devices and the server. Their results showed that only 5% of the data globally was shared, resulting in an increase of 30% for the CIFAR-10 dataset using FedAvg. However, such improvement is at the cost of sharing the users' private data with other users.

We looked further for other research work on FL. As in a different study in (Wahab *et al.*, 2021), the authors compared different FL algorithms with the classic FedAvg algorithm. The authors also emphasize the severity of the problem of non-IID, as it brings significant challenges to the learning accuracy of FL algorithms. Furthermore, they showed that the label distribution skew setting, where each client has a unique label, is the most challenging distribution against the other ones. We have a similar distribution in the continuous authentication problem. In conclusion, applying FL in the context of continuous authentication is doable. It has promising potential to preserve the privacy of users' data and offload huge loads from central servers.

In this context, we proposed a novel Federated Learning Scheme for continuous authentication embedding a warmup model addressing the challenges of Non-IID. Our federated learning warmup strategy is inspired by the work of (Zhao *et al.*, 2018) where we used a similar approach of an initializer model to boost the performance of the federated learning. The architecture of our proposed framework, illustrated in Figure 2.1, is described as follows:

- **Step 1:** On the client-side, we divide the clients' captured data into train and test sets. We send a portion of the training set to the server. It is noted that Step 1 takes place on the user device only once, during the first participation in a federated round.
- **Step 2:** The server collects the shared data portions from each participant of the FL round and trains a machine learning model centrally using the collected data. This model will act as a warmup model.
- **Step 3:** The server sends this trained model to C the clients participating in a federated round.
- **Step 4:** The client receives the global model sent from the server, loads this model and trains a new local model with their training dataset. Once the training is done, the user model is sent back to the server.
- **Step 5:** The server aggregates the received models by averaging them to create a global model that will be sent back to the participants.

Our architecture has addressed the non-IID challenge that brings significant challenges to FL when working with the FedAvg algorithm. As each user has a unique class label and different data sample sizes, we had to deal with the non-IID data problem in FL. Therefore the warmup model helped adjust the model weights divergence by starting with a better-trained model.

Algorithm 2.1 Warmup-based federated learning algorithm

```
1:  **Server executes:**
2:     initialize $w_0$
3:     $w_0 \leftarrow \textbf{Train}(-1, w_0, \zeta)$
4:     **for** each round $t = 1, 2, \ldots$ **do**
5:        $m \leftarrow \max(C \cdot K, 1)$
6:        $S_t \leftarrow$(random set of $m$ clients)
7:        **for** each client $k \in S_t$ **in parallel do**
8:           $w_{t+1}^{k} \leftarrow \textbf{TrainClient}(k, w_t)$
9:        $w_{t+1} \leftarrow avg(w_{t+1})$
10:
11: $\textbf{Train}(k, w, D)$ :
12:    $\beta \leftarrow$ (split $D$ into batches of size B)
13:    **for** each local epoch $i$ from 1 to $E$ **do**
14:       **for** batch b $\in \beta$ **do**
15:          calculate $w$ based on $\ell$
16:    return $w$
17:
18: $\textbf{TrainClient}(k, w)$ : // Run on client k
19:     return Train $(k, w, P_k)$
```

The algorithm we use in our approach is built upon the FedAvg algorithm (McMahan *et al.*, 2017a) by an initial warmup model. We define our system components with a central server and clients with mobile devices. We present the pseudo-code in Algorithm 4.1.

The server initializes a random model weight w_0 and updates such weights by training the model with ζ portion of users' data provided from user devices. Next, for each FL round, the server requests S_t, a random set of m available clients from the total number of clients K. Afterwards, and for each client indexed by k, the $\textbf{TrainClient}(k, w_t)$ trains a machine learning model for the client data P_k in FL round w^k_{t+1}. For the model's training, we split the client data D into mini-batch size B. We update the model weights w using a fixed learning rate η and a cross-entropy loss function ℓ. Finally, after collecting the users' models in a central server, the models are aggregated by averaging their weights.

The aggregation of the models results in one superior global model that can differentiate between the clients' labels prediction. The model created is later sent to the clients for another federated round. We repeat the process until a model convergence is reached or after a certain number of rounds that can be tuned.

2.5 Experiments

2.5.1 Implementation and Setup

In this section, we present the performed experiments on three different public datasets: MNIST (Deng, 2012), CIFAR-10 (Krizhevsky, Nair & Hinton, 2014) and Federated Extended MNIST (FEMNIST) (Caldas *et al.*, 2018). In order to adapt the dataset to fit and somehow emulate the continuous authentication use cases and environments, we managed to partition data distributions for the clients to match the unique non-IID problem presented in continuous authentication data. A continuous authentication dataset typically contains different samples of data for several clients. Each client has a unique label, and none of the clients share any label similarity between them. Therefore, we preprocessed each dataset as follows:

1. Samples of data for all clients are sorted by class/label.
2. Same-label samples are assigned to one client, resulting in ten unique classes for the CIFAR-10, 62 classes for the FEMNIST, and ten classes for MNIST.
3. Different sample size distributions are used for each dataset. We have used a random client distributor that shuffles the client data and assigns a random number of samples per client in a set of ranges that we can specify.

We used the datasets to test our hypotheses, containing numerous data samples. These publicly accessible datasets are widely used in the research community, and we can leverage them to compare our work with the related work in the FL context. We performed three types of experiments on each dataset: Centralized, FedAvg and Warmup. The centralized experiment combines all the clients' data on a server and trains a machine learning model. Our next experiment is the classical FL with the FedAvg algorithm. Initial model weights are created on the server-side, and the global model is sent to each selected client in a FL round. If the client is selected, it loads the global model and trains a new local model with the clients' data. After the training of the local model is completed, the client sends the model back to the server, keeping the raw data on the user device intact. In order to simulate realistic case scenarios where a portion of clients could not participate in a federated round, we used a random client selector that could simulate such a selection procedure in each of the federated rounds. Several criteria could be used for client selection to address this matter further, which is the subject of our future work. For our experiments, most of the participants are performing well. However, in real-life scenarios, many approaches exist in the literature regarding client selection techniques, such as resource management or Heuristic-based client selection AbdulRahman *et al.* (2020a). If most participants cannot participate in the federated round, the global model will be missing new information from these local clients; therefore, the model will take a longer processing duration to reach convergence. In addition, the model will be biased to correctly identify the positive samples of the clients who have provided the models. Finally, we performed the Warmup experiment approach as explained in the proposed framework section.

To perform our experiments, we used a windows machine with 16 GB RAM to run the experiments, having a CPU core i7-10700 @2.90GHz with 17 CPUs and GPU processing unit GTX 1660 Super of 6GB VRAM. We used Wandb (Biewald, 2020) to visualize the results of the experiments. The used learning rate in the experiments is based on a grid search, where we selected the best value while performing the experiments. Learning rates are the most sensitive hyper-parameter that we must carefully choose as they can severely increase or decrease the learning process of the machine learning models. Therefore, we performed extensive experiments to maintain a suitable learning rate for each of the datasets in each of the three different experiments we performed. As for the batch size, we demonstrated the impact of data size on model convergence. The model type used for the experiment is the convolution neural network (CNN), which deals with image classification problems. The convolutional neural network model consists of several layers that need a matrix shape of data as an input. Our data samples suit the input needed for models since the information shape of samples in MNIST and FEMNIST is 28*28 pixels and 32*32 for the CIFAR-10 dataset, where each pixel in the image is considered a feature. We have used the same Convolutional Neural Network model that was used in (McMahan *et al.*, 2017a) which consists of two 5*5 convolution layers, with 32 and 64 channels for the first and second layers. Each layer follows a 2*2 max pooling, a 512 units fully connected layer, a Relu activation function, and a final softmax output layer. In the following sub-section, we describe each used dataset alongside the details of the experiments.

2.5.2 Experimental Results

MNIST. The MNIST dataset (Deng, 2012) contains images of handwritten digits with 28*28 pixels per image. It contains 60,000 images for training and 10,000 for testing. This dataset is widely used for training image processing systems using various machine learning algorithms. The data partitioning yields ten unique class representations, from zero to nine. We distributed random samples between 800 and 1000 for each client. we have used a CNN model alongside a fixed learning rate of 0.1 for federated and 0.001 for the centralized dataset. The MNIST plots presented in Figure 2.2 show that we were able to achieve excellent results in terms of accuracy for the Centralized, Warmup and FedAVg. We used only 5% of the data in the warmup experiment to create an initialized global model on the server-side. We also assumed that only 20% of clients would be available to participate in a FL round. The results show that the centralized training of the model yields 98.53%. However, we note that the warmup model slightly improves the convergence of the MNIST model, achieving similar results as the FedAvg algorithm of 97%.

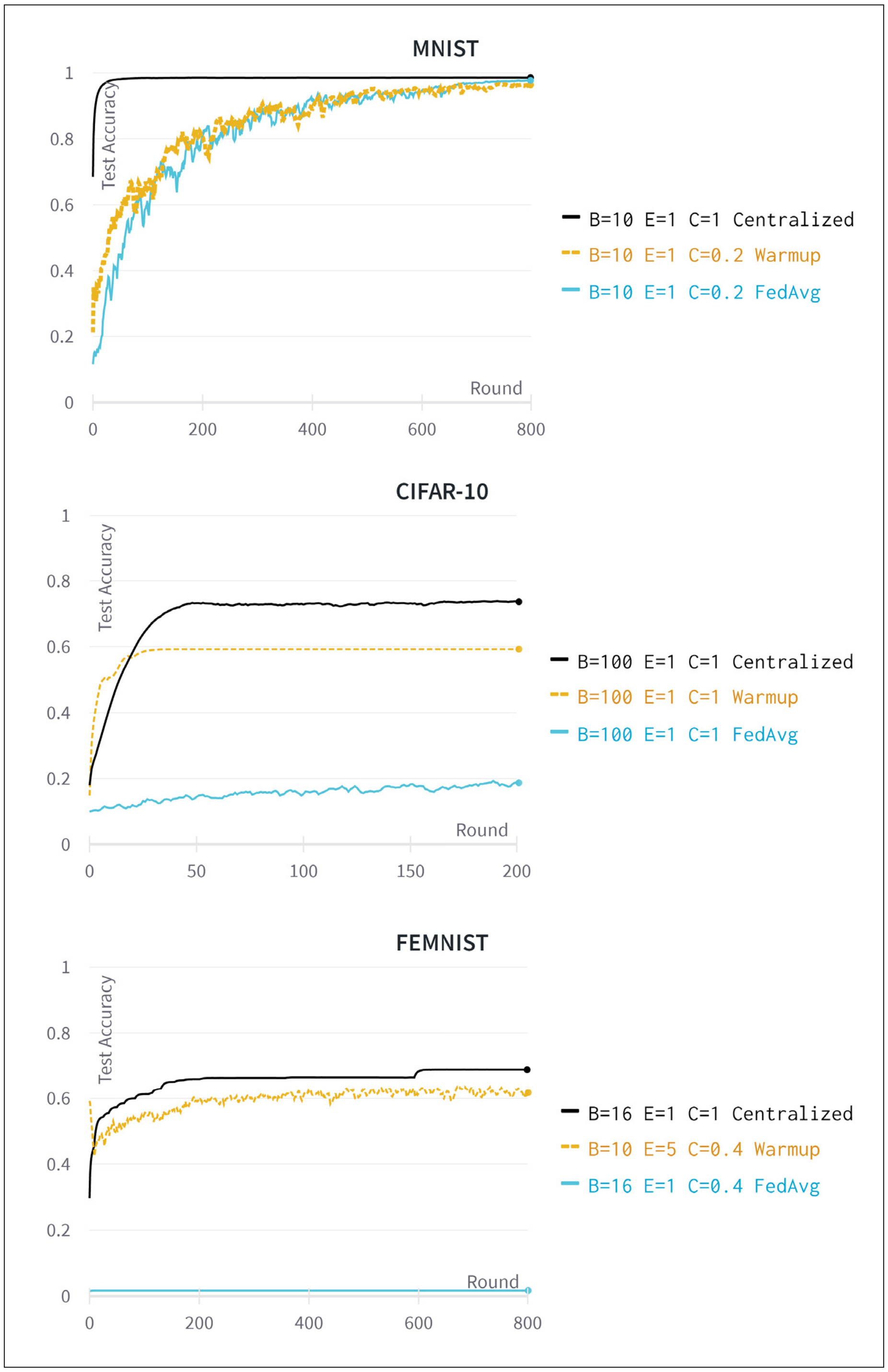

Figure 2.2 Test set accuracy vs. communication rounds for MNIST, CIFAR-10 and FEMNIST datasets

CIFAR-10. The CIFAR-10 dataset (Krizhevsky *et al.*, 2014) contains images of 32*32 pixels by image. The dataset contains ten different classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset is also widely used to train machine learning and computer vision algorithms. Our data partitioning of this dataset resulted in 10 clients, each with a unique label, and we used the full dataset data samples. For simplicity, we have selected all the clients per round. We run the experiments of CIFAR-10 with a six-layered CNN model having a kernel size of 3. The learning rate we used is 0.1 for the federated versions of the experiments and 0.001 for the centralized dataset. Figure 2.2 shows the different experimental results and their configurations. We also compared our federated results with a centralized version of the Cifar-10 dataset. We can notice how the centralized Cifar-10 experiment reached a convergence with only a few rounds compared to the federated non-IID version, which achieved inferior results. However, using our proposed warmup strategy, we increased the accuracy results by 30% while sharing only 5% of the data with the server.

Federated Extended MNIST. The FEMNIST dataset (Caldas *et al.*, 2018) contains images of 28*28 pixels by image, and it is built by partitioning the extended version of the MNIST dataset. The extended dataset is created by reference to the writer of the digit or character. It contains pictures of digits and English letters. The dataset has 62 classes, and each has a different number of data samples for the images. The partitioning of this dataset resulted in 62 custom-built clients for this dataset, each with a unique label. For the warmup experiment, we only used 5% of the data from each client. We used the same CNN model used for the MNIST dataset, but we updated the final output shape to 62 to match the configuration we have for the FEMNIST dataset. The learn rate we have used to run the federated results is 0.1 and 0.01 for the centralized version of the dataset. Figure 2.2 FEMNIST experiment shows the different experiments' results and their configurations. We can see that the warmup model gave the experiment result plotted in yellow an initial boost of 60% accuracy. It did not improve much further through the communication rounds. We can see from the plots that the warmup model achieved a similar result comparing the centralized version of the model, in contrast to FedAvg results, where the accuracy did not exceed 2%.

2.5.3 Discussion

While previous research focused on sharing users' data to create a machine learning model on a central server. However, we demonstrate that devoting FL techniques for continuous authentication is applicable when the data limitations are addressed. The continuous authentication behavioural data are of a unique shape, as each user has his label and does not share the data with other users. Therefore, we have a unique case of non-IID data per client, where each user has only their unique label and lacks other users' labels in their data. Such data label variety and uniqueness among the FL participants increases the weights divergence between the models created per client, thus decreasing the global model performance on aggregation. We showed that FL architecture removes the burden of exposing private data to the server while leveraging clients' learning models. Extensive experiments show that our novel warmup strategy improves the performance of the FedAvg algorithm by 40% for CIFAR-10 and 60% for FEMNIST datasets, as shown in Figure 2.2. Compared to Zhao *et al.* (2018), our approach did not share any data between the clients and managed to train the warmup model securely on the server without compromising the users' models by sharing data with other unknown devices. To further discuss the results obtained, we note that the total number of unique classes in the MNIST dataset is ten, which is relatively low compared to the FEMNIST dataset, which has 62 classes, over six times the number of classes presented in MNIST. In addition, the image representation in each of the samples in the MNIST dataset is straightforward compared to more complex images such as the CIFAR-10, where the images contain much more details. Therefore, machine learning was able to better learn from the images of MNIST since it could detect the variance of the labels more quickly than the other two datasets. The experiments have been done based on the results obtained considering the convergence of the model. We noticed that for both MNIST and FEMNIST, the model has room for improvements with the increased number of rounds until 800. On the other hand, for the CIFAR-10, we did not note any significant improvement in the results obtained after 200 rounds. Therefore, the model has achieved convergence with fewer rounds. The datasets we used have provided preliminary results about the applicability of the FL concept in the continuous authentication problem. However, realistic authentication datasets are needed to reconfirm our proposed approach's results, which we aim to tackle in our subsequent work.

2.6 Conclusion and future work

Continuous authentication is a promising technique that can provide clients with an extended protection mechanism against fraudulent activities, given that private data is shared with the central server to create an ML model using combined clients' data. To address the raised privacy issues, we proposed in this paper a federated learning approach offering continuous authentication while preserving the users' data privacy by keeping the user's raw data on the device intact. Moreover, we addressed the problem of the non-IID data through a warmup model that acts as a booster for the federated learning process. Experimental results showed that FedAvg could not handle the unique data distribution alone, while the proposed warmup strategy has increased the accuracy results by 40% for CIFAR-10 and 60% for FEMNIST. For future work, we plan to implement our proposed solution on real-world datasets for continuous authentication. We also aim to investigate other federated learning algorithms that tackle the non-IID problem to achieve better accuracy results with much fewer communication rounds.

CHAPTER 3

CRSFL: CLUSTER-BASED RESOURCE-AWARE SPLIT FEDERATED LEARNING FOR CONTINUOUS AUTHENTICATION

Mohamad Wazzeh[1,2] , Mohamad Arafeh[1,2] , Hani Sami[1,2] , Hakima Ould-Slimane[3] , Chamseddine Talhi[1] , Azzam Mourad[4,2] , Hadi Otrok[5]

[1] Department of Software and IT Engineering, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Artificial Intelligence and Cyber Systems Research Center, Department of Computer Science and Mathematics, Lebanese American University, Chouran, Beirut 1102 2801, Lebanon
[3] Department of Mathematics and Computer Science, Université du Québec à Trois-Rivières, 3351 boulevard des Forges, Trois-Rivières, Québec, Canada G8Z 4M3
[4] KU 6G Research Center, Department of Computer Science, Khalifa University, Shakhbout Bin Sultan St, Hadbat Al Za'Faranah, Zone 1, Abu Dhabi, United Arab Emirates
[5] Center of Cyber-Physical Systems (C2PS), Department of Computer Science, Khalifa University, Shakhbout Bin Sultan St, Hadbat Al Za'Faranah, Zone 1, Abu Dhabi, United Arab Emirates

Paper published in Journal of Network and Computer Applications, November 2024

3.1 Abstract

In the ever-changing world of technology, continuous authentication and comprehensive access management are essential during user interactions with a device. Split Learning (SL) and Federated Learning (FL) have recently emerged as promising technologies for training a decentralized Machine Learning (ML) model. With the increasing use of smartphones and Internet of Things (IoT) devices, these distributed technologies enable users with limited resources to complete neural network model training with server assistance and collaboratively combine knowledge between different nodes. In this study, we propose combining these technologies to address the continuous authentication challenge while protecting user privacy and limiting device resource usage. However, the model's training is slowed due to SL sequential training and resource differences between IoT devices with different specifications. Therefore, we use a cluster-based approach to group devices with similar capabilities to mitigate the impact

of slow devices while filtering out the devices incapable of training the model. In addition, we address the efficiency and robustness of training ML models by using SL and FL techniques to train the clients simultaneously while analyzing the overhead burden of the process. Following clustering, we select the best set of clients to participate in training through a Genetic Algorithm (GA) optimized on a carefully designed list of objectives. The performance of our proposed framework is compared to baseline methods, and the advantages are demonstrated using a real-life UMDAA-02-FD face detection dataset. The results show that CRSFL, our proposed approach, maintains high accuracy and reduces the overhead burden in continuous authentication scenarios while preserving user privacy.

3.2 Introduction

Traditional authentication methods face privacy and accuracy challenges when devices are used for extended periods. One-time passwords and PINs are vulnerable to cyber-attacks, such as credential theft, shoulder surfing, and session hijacking (Liang *et al.*, 2020). Continuous authentication is preferred for IoT and mobile devices, which utilize unique physical and behavioural traits like gait analysis, voice, facial images or motion patterns (Pritee *et al.*, 2024). This method implicitly verifies the user's identity while interacting with IoT devices. Continuous authentication, as shown in Figure 3.1, involves constantly verifying a person's identity using their unique physical or behavioural characteristics (Sánchez *et al.*, 2021). For example, how a person walks or moves while using a device can be used to confirm his/her identity in the background. Users' data is connected to a unique label representing their true identity. This makes it challenging to classify user data accurately using traditional ML tools. The data collected is not independent and identically distributed (non-IID) since data sample numbers and labels differ from one user to another. The classical continuous authentication approaches presented in various studies (Lee & Lee, 2017; Zhu *et al.*, 2019; Lu *et al.*, 2018) transfer behavioural data to an external server to train an ML model. However, such approaches may compromise users' privacy and security, exposing them to cyberattacks like spoofing and data hijacking. Continuous user authentication is constantly evolving, with new techniques being

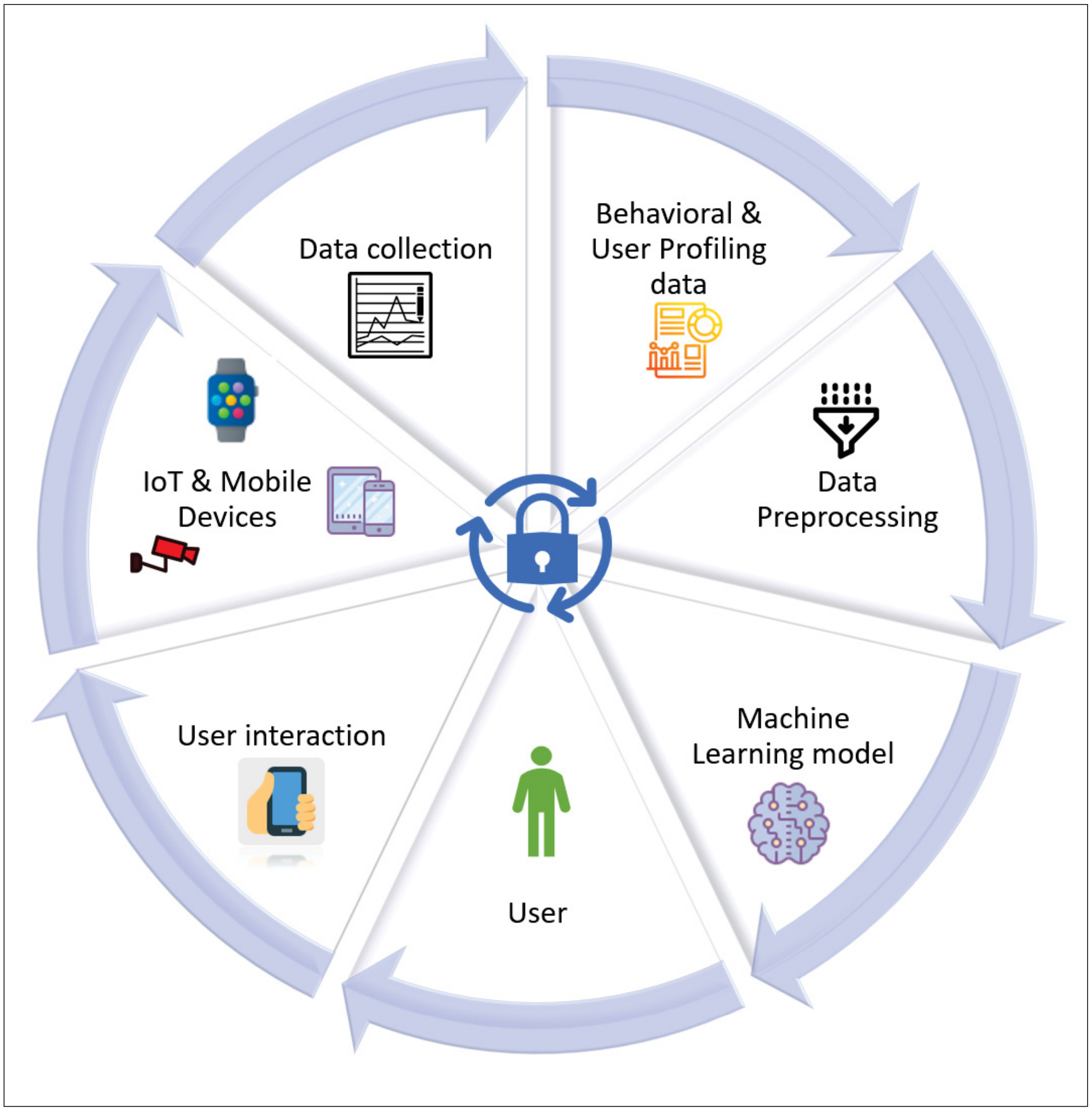

Figure 3.1 User Interaction and Data Flow for Continuous Authentication in Mobile and IoT Devices

developed to enhance traditional systems. Numerous studies have utilized distributed learning for continuous user authentication where a multi-class classification model is typically employed to train the users' models (Feng *et al.*, 2024; Wazzeh *et al.*, 2022a; Monschein *et al.*, 2021; Wazzeh *et al.*, 2023; Yazdinejad *et al.*, 2021; Wazzeh *et al.*, 2022b). One such approach is federated learning (FL), where clients train local models with their data and transfer weights rather than their raw data to the server for aggregation (McMahan *et al.*, 2017b). This approach demonstrates an advantage, as the client's models can be merged on the server end, enabling a singular model to distinguish between various client labels (Wahab *et al.*, 2021; AbdulRahman *et al.*, 2020b; Truong, Sun, Wang, Guitton & Guo, 2021). In addition, examples of binary FL are being utilized for continuous authentication (Hosseini *et al.*, 2021), wherein a softmax function with the client's model indicates similarity to feature data. However, achieving high accuracy in such cases requires many communication rounds and may not perform well in highly imbalanced datasets (Yang *et al.*, 2022).

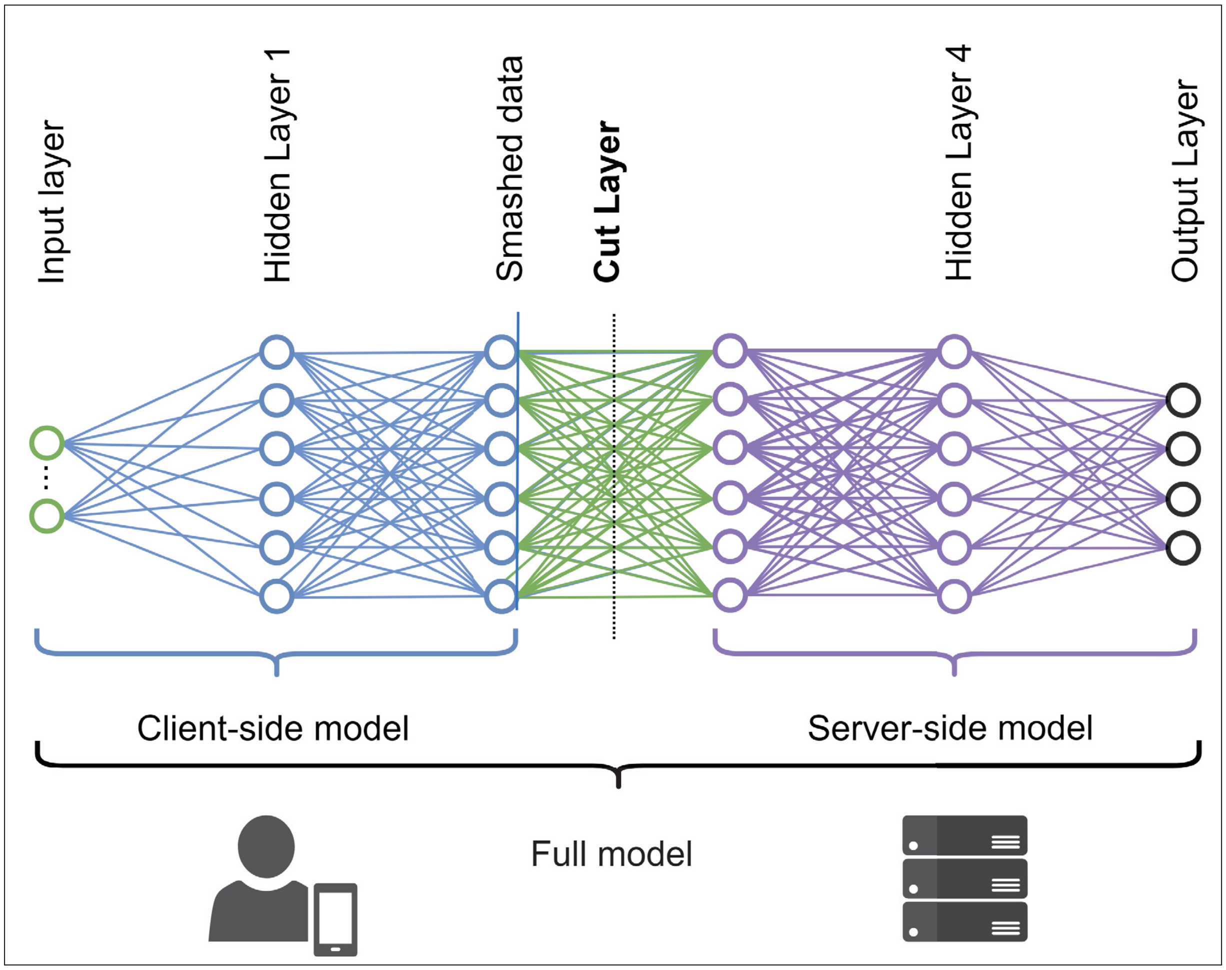

Figure 3.2 Neural Network Model Splitting

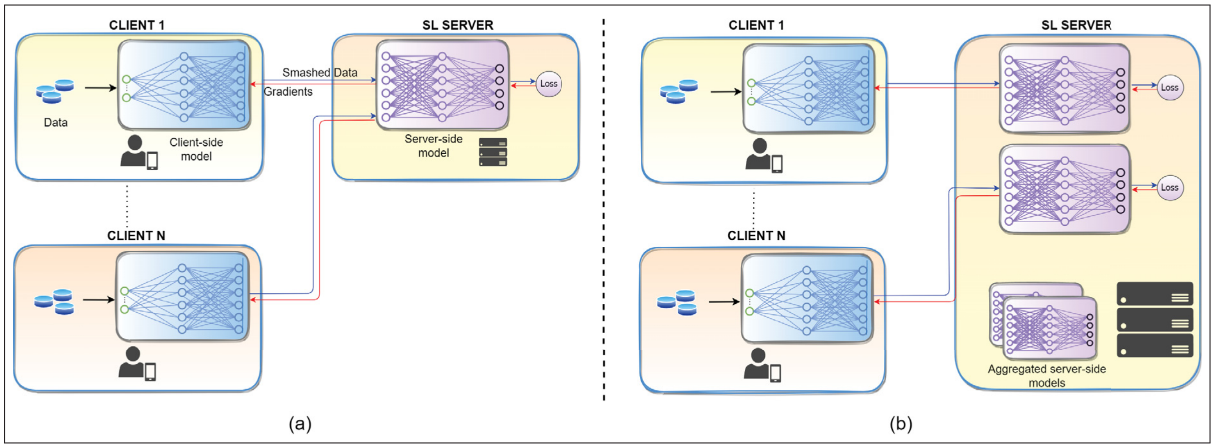

Figure 3.3 A comparison of client-server SL techniques: (a) Sequential SL, where each client sends model updates to the server one at a time, and (b) Parallel SL, where multiple clients send model updates simultaneously to the server

The FL approach facilitates collaborative training across nodes, sharing knowledge between clients in separate clusters and enhancing model resilience (Siraj & et al., 2022). This method improves model security and reduces the computational burden on clients (Gao *et al.*, 2021). While continuous authentication techniques are typically used in FL (Monschein *et al.*, 2021), they face limitations such as impracticality on resource-constrained IoT devices and model convergence issues due to unique label characteristics (Wazzeh *et al.*, 2022a). To address these challenges, integrating SL with FL is crucial. FL maintains privacy by ensuring raw data remains on client devices and enables parallel training, transforming the traditionally sequential SL process into a parallel one. This integration mitigates weight divergence issues, enhances efficiency and scalability, and allows simultaneous SL training across different nodes. By leveraging the strengths of both FL and SL, our framework improves the robustness and performance of continuous authentication systems while distributing computational load, making it feasible for IoT devices.

Conversely, SL provides a solution by allowing clients to train only specific neural network layers rather than the entire model, leading to faster training times and reducing the resource burden on the client (Vepakomma *et al.*, 2018). As represented in Figure 3.2, SL involves partitioning the neural network model at the cut layer into two parts, with the server undertaking the training process by solely transferring the compressed data of the cut layer weights from the client end, keeping all personal data at the client's device. Additionally, SL's non-aggregation feature has been shown to improve model performance and minimize weight divergence issues between clients (Gao *et al.*, 2020). Nevertheless, employing SL to resolve the problem of continuous authentication could be restricted by concerns related to security and training. This is because SL utilizes a sequential training methodology, which necessitates that each client wait for the training of the other client to terminate and receive model updates before continuing with their training process, as shown in Figure 3.3(a).

To overcome the issues of FL and SL and benefit from the strength of each one, we propose in this paper a novel framework for continuous authentication that combines FL and SL and performs multi-class classification to identify users. In contrast to the classical SL method shown in Figure 3.3(a), the clients in Figure 3.3(b) can train their models with a server in parallel to improve training efficiency (Gupta & Raskar, 2018; Duan *et al.*, 2022). Incorporating the strengths of FL and SL can enhance model security, data privacy, and neural network training accuracy (Thapa *et al.*, 2022). The collaborative training process entails a joint effort between the split server, the federated server, and the clients. Each client can train their models in parallel on the split server using separate server-side weights. By hosting parallel server-side weights for each client, they can train simultaneously, and these weights are then combined once all participants have completed their training rounds.

To tackle the challenge of device heterogeneity during training, we have introduced a strategy of organizing devices into clusters. We aim to mitigate the impact of varying computing capabilities across devices by utilizing a clustering technique. This involves grouping devices with similar capabilities, reducing communication overhead and enabling effective collaboration among clients within a cluster during the training process (Tu *et al.*, 2021; Wazzeh *et al.*, 2023).

Moreover, when applying clustering, another challenge arises in selecting proper clients to participate in learning. The existing research on continuous authentication using distributed learning lacks advanced client selection techniques and has not thoroughly studied their effects.

This can lead to inefficient resource utilization and the missing participation of unique users from the learning model, as the current approaches employ random client selection based on predetermined percentages that are unrealistic when considering the heterogeneity of clients' resource capabilities and the need for a continuous authentication model. Additionally, scalability issues with random and deterministic selection algorithms require further attention. It is crucial to address these challenges to ensure fair resource distribution and optimal performance in these architectures. To enhance the adaptability and robustness of resource allocation strategies, we propose, as part of our framework, a heuristic-based algorithm that selects clients based on their changing resource constraints and follows security objectives where each client holds a unique label and has to optimize its participation while maximizing the model performance. This study introduces a novel approach to achieving continuous user authentication on mobile and IoT devices with improved security and efficiency. In this paper, we describe the structure and algorithms of our proposed method and compare it with existing authentication methods to showcase its effectiveness. Our experimentation demonstrates that our method offers secure and efficient access control for IoT and mobile devices. Our contributions can be summarized as follows:

- Proposing a Cluster-Based Resource-aware Split Federated Learning (CRSFL) framework tailored for IoT devices. CRSFL ensures users' privacy by maintaining data on their respective devices.
- Introducing a filtering methodology that utilizes capacity constraints and a clustering process to guarantee clients' engagement with sufficient resource capacities. The constraints are established according to the initial model and device specifications to cluster comparable IoT devices.
- Applying a machine learning approach to predict the client device usage of resources. The machine learning evaluates each client's dynamic resource availability in every round and estimates its model training usage.
- Employing a heuristic client selection algorithm that optimizes resource allocation through a five-objective scenario based on a Genetic Algorithm (GA). This process considers the

client’s unique labels, number of devices, event rate, and history of client selection and minimizes wait time during model training rounds for efficient client selection.

- Conducting extensive experiments on real-life authentication dataset to compare and evaluate our approach to existing methods for continuous authentication.

The experimental results obtained by employing CRSFL reveal its superiority and offer insights for future research in this domain.

3.3 Related Work

Using split and federated learning techniques in decentralized machine learning tasks has shown promising results in improving user authentication on mobile and IoT devices. These methods ensure efficient access control, enhancing the devices’ security and privacy.

3.3.1 Federated learning in Continuous authentication

The authors discussed in (Wazzeh *et al.*, 2022a) presented an FL method that leverages warm-up models to improve the performance of authentication models and user verification in FL. In their approach, users contribute a portion of their data to host the booster model on the server, which is then utilized to initiate the federated learning rounds. Through their work, the authors showcased the effectiveness of their approach in handling unique non-IID data and enhancing model weights. Furthermore, the study in (Wazzeh *et al.*, 2022b) demonstrated that transfer learning could enhance the accuracy and resilience of models when applied in an FL framework for continuous authentication. This involves the client employing pre-trained model weights on a relevant contextual dataset to facilitate easier identification of class labels.

The work in (Monschein *et al.*, 2021) explores how to authenticate users within web-based systems by analyzing their behaviour and using machine learning models to detect anomalies. These models are trained federated by multiple organizations, which work together in a peer-to-peer fashion while safeguarding their local data. A data governance component is proposed to ensure that the models and data are high quality and compliant while aligning with the objectives and requirements of all participants involved.

The research work in (Yazdinejad *et al.*, 2021) outlines a drone authentication architecture that utilizes drones' radio frequency (RF) capabilities within IoT networks, leveraging a FL approach. The proposed model employs homomorphic encryption and secure aggregation to protect model parameters during data transmission and aggregation.

The authors in (Feng *et al.*, 2024) present a privacy-preserving aggregation scheme designed for FL in vehicular ad-hoc networks (VANETs). The approach enables collaborative training of a global model without revealing participating vehicles' local data or gradients. The scheme introduces a continuous authentication mechanism based on a non-interactive zero-knowledge proof protocol, which verifies the legitimacy and integrity of clients and their updates. Additionally, the scheme utilizes edge devices to assist with the aggregation process, reducing communication and computation overhead.

The difference between traditional FL methods and those tailored for continuous authentication primarily stems from the unique nature of continuous authentication data. In continuous authentication scenarios, each client has a distinct label with no overlap between clients, leading to significant convergence issues for the model. FL alone often struggles in this context due to the highly non-IID (non-independent and identically distributed) data distribution across clients, which poses challenges for model convergence and stability.

Considering these unique data characteristics, the related work suggests that the standard FL approach does not work well. This necessitates additional measures to ensure that the model weights do not diverge significantly from one client to another. Implementing these measures is crucial to maintaining model accuracy and effectiveness in continuous authentication.

To tackle these challenges, integrating Split Learning (SL) with FL can offer significant benefits by harnessing the parallelism in model training across distributed clients while addressing the issues of data heterogeneity and convergence. SL enables the partitioning of the model into components trained on the client and server sides, thereby reducing the computational burden on client devices and enhancing privacy.

The cited works demonstrate promising results in using a multi-class model in FL for continuous authentication, highlighting the potential of hybrid approaches to address the inherent challenges. Additionally, while some studies have explored the use of binary models (Hosseini *et al.*, 2021), it is worth noting that such approaches often require many rounds to achieve high-accuracy results. Nonetheless, it is essential to acknowledge that the capacity of IoT devices to train machine learning models may be restricted by their available resources. Furthermore, the issue of client selection in FL for continuous authentication has not been fully explored in the works above. The authors have adopted an approach to client selection based on random or percentage-based criteria without considering available resources in their methodology.

3.3.2 Split learning and Federated learning

The authors in (Thapa *et al.*, 2022) employ the split federated learning (SFL) technique to harness the benefits of distributed computing and parallel client-side training. By aggregating the weights, their approach ensures that clients can train their model in parallel with split and federated servers. Eliminating the need to wait for each client to train the other, as in the case of vanilla split learning.

In the work of (Arafeh *et al.*, 2023b), the authors proposed a unique double clustering approach to address the challenges posed by non-IID clients and stragglers in the IoT environment. Their approach involves two clustering layers - the first layer focuses on biased clients while the second layer handles stragglers. This method has yielded faster execution times through parallel execution and increased accuracy by clustering non-IID clients into IID clusters.

In the work in (Samikwa *et al.*, 2022), an SL framework is explored to address the issue of slow devices, also known as "stragglers," to minimize training time and energy consumption in IoT devices. The authors implemented a model split point that is adjustable based on the available resources of each device. Their research has demonstrated that this approach effectively reduces energy consumption and mitigates the impact of stragglers.

In (Wu *et al.*, 2023), the authors employed a cluster SFL technique to address the issue of training latency for client models in a heterogeneous IoT environment. The authors used a cut layer selection method to create user clusters and optimized client allocation into specific clusters using radio spectrum allocation and channel conditions.

The authors present an SFL approach version in a work of (Han *et al.*, 2021). They tackle the challenges of latency and communication efficiency by implementing local loss-based training for SL, as opposed to the global loss function of the entire model. This approach resulted in comparable accuracy results with a reduced latency.

The above-related work enhancements improve the distributed learning system by incorporating an SL component, effectively reducing the client's overhead burden. The integration of FL with SL allows for parallel training, mitigating the challenges posed by the sequential nature of SL. While the authors addressed the communication and channel conditions challenges that come with machine learning model training, they neglected to explore using an advanced selection algorithm for continuous authentication when determining client selection criteria. Our work goes beyond addressing straggling clients and execution time. We have developed a client selection algorithm that optimizes the selection of clients based on their resource capacities and data sample properties using a multi-objective algorithm.

3.3.3 Split learning and Federated learning in Continuous authentication

In their research, (Oza & Patel, 2021b) propose a method for user authentication that utilizes federated and split learning principles. The authors divide the training process between the client and server, but instead of sharing the model weights from the client, they share compressed statistical data for each client. The server then aggregates this data using the federated averaging (FedAvg) algorithm described in (McMahan *et al.*, 2017b). It is important to note that assuming training can be completed in a single round is unrealistic due to constantly changing user data and unavailable samples. Additionally, sharing mean and variance may raise privacy concerns.

The paper in (Wazzeh *et al.*, 2023) is the only work that explores SFL for continuous authentication in its framework components. The authors propose a cluster-based method for dividing FL models, but their approach only considers differences in unique labels without considering available resources. Additionally, client selection was done randomly, without considering resource capacity and dynamic availability for each round.

Our methodology builds upon existing research, taking into account the limitations noted. We optimize client selection in each round by grouping clients according to their resource capacity. We employ a machine learning algorithm to predict resources and an advanced heuristic algorithm that dynamically considers resource availability. As a result, we address previous works' limitations while improving the authentication process's efficiency and security.

3.4 Methodology

3.4.1 Overview

We have developed an innovative approach to ensure continuous user authentication using split and federated learning on mobile and IoT devices. Our technique prioritizes safeguarding user privacy and model security while utilizing the benefits of distributed computing. Figure 3.4 illustrates the architecture of our proposed framework. The architecture displays a list of clients and their resources, filtering out devices that cannot train the ML model and grouping them into clusters based on resource similarity. A predictive machine learning model is then trained using rounds history and utilized in the training rounds to select resources based on dynamically available resources. Next, a heuristic approach is employed to choose clients from the list of available devices. These clients then train the model in parallel using the split server while the server part's model aggregation occurs on the split server. The client model aggregation is carried out on the federated server, and the resulting globally aggregated models

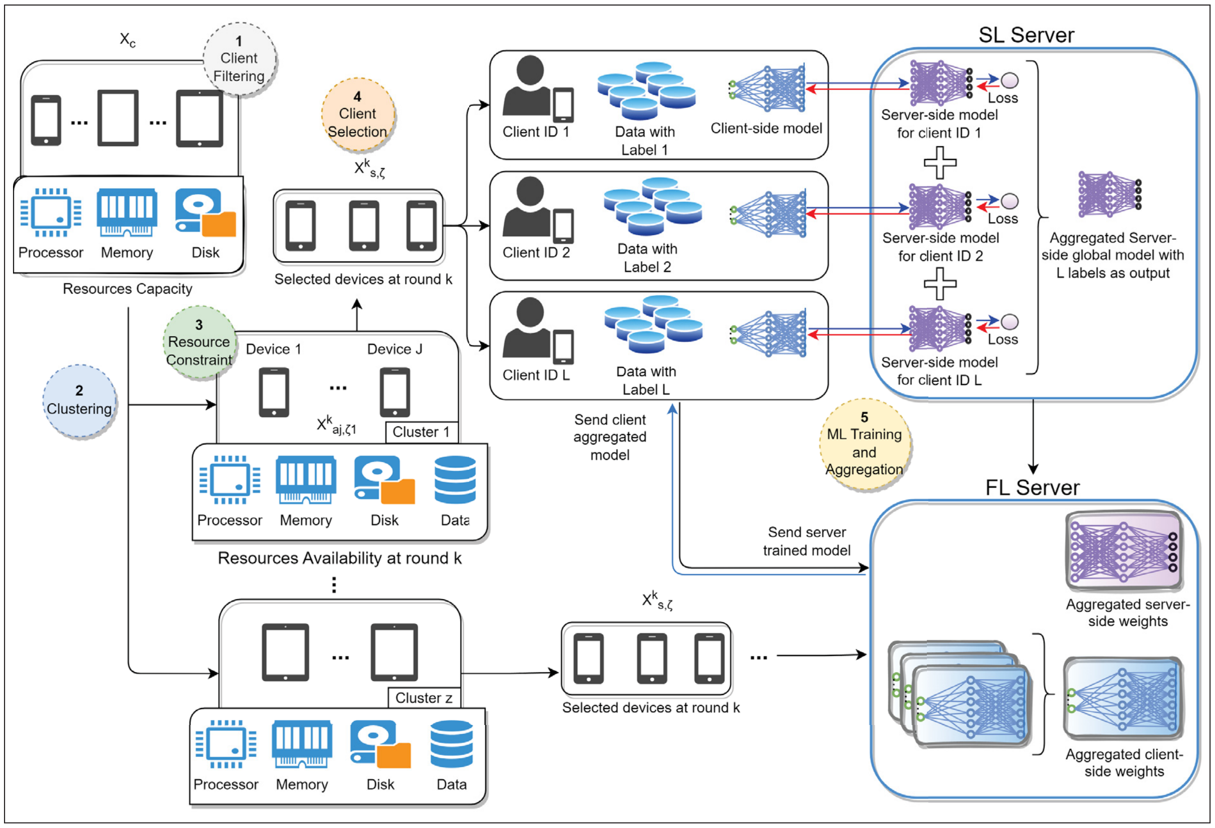
X_c
1 Client Filtering
Processor
Memory
Disk
Resources Capacity
4 Client Selection
$X^k_{s,\zeta}$
Selected devices at round k
3 Resource Constraint
Device 1
Device J
$X^k_{aj,\zeta 1}$
Cluster 1
2 Clustering
Processor
Memory
Disk
Data
Resources Availability at round k
Cluster z
Processor
Memory
Disk
Data
$X^k_{s,\zeta}$
Selected devices at round k
Client ID 1
Data with Label 1
Client-side model
Client ID 2
Data with Label 2
Client ID L
Data with Label L
Send client aggregated model
SL Server
Server-side model for client ID 1
Loss
Server-side model for client ID 2
Server-side model for client ID L
Aggregated Server-side global model with L labels as output
5 ML Training and Aggregation
FL Server
Send server trained model
Aggregated server-side weights
Aggregated client-side weights

Figure 3.4 Overall Cluster-based Resource-aware Split Federated Learning (CRSFL) Architecture

are subsequently transferred to the next cluster. The notable difference between conventional FL techniques and FL tailored for continuous authentication is due to the nature of client data in such scenarios. In traditional FL, clients may have shared labels or overlapping data, unlike in continuous authentication, where each client has a distinct label, presenting unique challenges in model convergence and performance optimization. This distinction highlights the crucial role of client selection in achieving optimal continuous authentication performance. In the following subsection, we will explain each of the architectural elements and their respective functions in more detail.

3.4.2 Architectural Stages

Our proposed approach, CRSFL, integrates various components for training machine learning models. These include clients, clusters, SL servers, and FL servers, which work collaboratively in layers or stages to select clients and conduct collaborative training effectively. These encompass client filtering, clustering, resource constraints, client selection, and ML training and aggregation. By thoroughly examining each aspect, we can comprehensively describe the stages in preparing for client selection within SFL scenarios.

3.4.2.1 Client Filtering

Each individual in the system is identifiable through a device with a distinct collection of data samples. Although a user may possess multiple devices in the authentication system, each is uniquely labelled. Nonetheless, all devices linked to the same user share a common label, such as the user's behavioural data. On the other hand, the labels for dissimilar users are always distinct, with each individual's behavioural data being unique to their characteristics. During the initial phase, we assess potential users to determine if they have the necessary resources to support the model. This involves comparing each user's resource capacity to the hosting requirements of the model. We assume each user has a fixed resource capacity that remains constant over time. Users who cannot meet the criteria are eliminated from the pool of potential participants. This approach ensures that only users with sufficient resources are considered for further selection.

3.4.2.2 Clustering

Clusters are groups of clients managed by a SL server. A cluster can consist of one or more devices per client, and devices with similar resource capabilities are grouped to optimize performance. We employ a k-means clustering technique that can handle various devices. The k-means algorithm starts by randomly initializing cluster centroids and then iteratively updating them until convergence. At each iteration, it assigns data points to the nearest centroid based on Euclidean distance and subsequently updates each centroid to reflect the mean of the data points assigned to it (Sinaga & Yang, 2020). The clustering process involves sorting filtered clients into clusters based on their resource capacities, aiming to create sets of devices with similar training capabilities. This approach enhances the efficiency of SFL by minimizing communication overhead and enabling clients in the same cluster to collaborate on training with minimal differences in their resource capabilities. The objective is to cluster clients so that their resource profiles closely resemble each other. Doing this can reduce training idle time between devices and overall latency in a parallel SL system. This process partitions devices into distinct groups, allowing devices within each group to train their device-side models simultaneously. This method is more effective than the vanilla SL approach, which trains devices sequentially (Vepakomma *et al.*, 2018). Clustering also addresses the straggler effect of device heterogeneity and network dynamics by grouping devices with matching computing capabilities (Tu *et al.*, 2021).

3.4.2.3 Resource Constraint

We systematically investigate each cluster to identify clients who may struggle with completing the model training process. To achieve this, we employ a Random Forest resource prediction model that draws on historical data from previous client training engagements. The Random Forest regression model is a highly effective machine-learning algorithm that utilizes decision trees to identify complex patterns within data and make predictions about the values of a target variable. The algorithm minimizes the mean squared error (MSE) between predicted and actual values to ensure precise predictions (Segal, 2004). Resource constraints must be considered when training a machine learning model on devices with limited resources (Li *et al.*, 2021a). Additionally, when deploying machine learning models across multiple devices, it is essential to consider the varying resources available on each device and the differing resource demands of the model. This may result in differing model requirements among clients depending on the size of their data samples and available resources (AbdulRahman *et al.*, 2020a). To overcome this challenge, we developed a machine learning model that uses a client's resource usage history from multiple training rounds to predict actual resource utilization. We gather usage data from devices with varying resource capacities to generate the input data for our prediction model. The prediction model enables us to identify clients who may experience dropout due to resource limitations, allowing us to filter out such clients from the available pool of candidates for model training.

3.4.2.4 Client Selection

The client selection process involves identifying the optimal clients for training our machine learning model. We ensure that only clients with sufficient resources to complete the training are forwarded to our heuristic selector algorithm, thus avoiding any early dropouts (Chahoud *et al.*, 2023). The heuristic selector algorithm then identifies the most appropriate clients with predetermined objectives. For more information on the algorithm and its objectives, please refer to section 5.5.

3.4.2.5 ML Training and Aggregation

This method entails training a multi-class classification neural network model to categorize input data into different classes, each corresponding to a specific client label. In Split Federated Learning (SFL), each chosen client device trains its model using raw data up to the cut layer. In addition to the number of samples and the unique client label, the resulting weights from this layer are transmitted to the split server for forward and backward propagation, completing the training process. No private data is sent to the server; only the final layer model weights, the number of samples, and the unique client label are transmitted. Once the client-model weights are updated, these weights are sent to the federated server for aggregation. In an SL architecture, the server concurrently receives the server-side portion of the neural network model along with the cut layer's weights or aggregated data from clients. The server then simultaneously trains the remaining layers of the model for each device. Upon completion, the server aggregates the server-side model weights from clients through averaging and sends the final aggregated server-side weights to the federated server. The FL server initiates neural network models, clusters clients, and aggregates their weights by averaging. The updated weights from clients and servers are subsequently transferred to the next cluster to continue the training process.

3.5 Problem Definition and Formulation for Client Selection

This section presents the problem definition and formulation for our proposed approach. We outline the mathematical formulation for the input, output, and client selection objectives. Additionally, we explain the client selection algorithm used and discuss its complexity.

3.5.1 Problem Definition

In this subsection, we will focus on tackling the challenge of client selection during SFL rounds. Initially, we evaluate potential clients during the initial phase to ensure they possess the necessary resources to support the model. We assume each client has a fixed resource capacity that remains constant over time. From there, we filter out clients who do not meet our requirements and group the remaining clients into clusters using K-means based on their resource capacities. This allows us to create sets of devices with similar training capabilities, ultimately leading to more significant SFL efficiency. When deploying machine learning models across multiple devices, it is crucial to consider the resources available on each device and the varying resource demands of the model. It is also important to note that differences in available resources or the size of data samples may result in differing model requirements among clients.

We have incorporated a dynamic client selection mechanism to ensure that our framework can effectively adapt to highly dynamic environments with numerous devices. This mechanism consistently evaluates and updates the resource capacities of clients and adjusts clusters accordingly. By integrating real-time monitoring and feedback loops, we can dynamically assign clients to clusters that best match their current capabilities, ensuring efficiency even as device conditions change. Collectively, these strategies uphold the resilience and scalability of our CRSFL framework, even in environments with fluctuating resources and numerous participating clients. Our selection process aims to select clients in split federated rounds according to specific objectives. During this phase, the chosen clients work with the split server to train the machine learning model, with selection occurring in each learning round. Our client selection objectives align with our ultimate goal of continuous client authentication. The objectives can be defined as follows:

1. Maximize the number of clients' devices: Select as many clients as possible during the learning phase.
2. Maximize the number of unique clients: During the learning phase, select as many devices with unique labels as possible since each represents a unique user requiring authentication.
3. Maximize the number of unique clients: During the learning phase, select as many devices with unique labels as possible since each represents a unique user requiring authentication.
4. Maximize the learning quality: Enhancing the model's learning quality is related to carefully selecting clients with many data samples.
5. Minimize the client idle time: Selecting clients with similar processing capabilities is necessary for optimizing the SL process.
6. Maximize the probability of a client being selected: The fitness function is designed to maximize client selection in learning rounds.

These requirements transform the problem into a multi-objective optimization problem, akin to a multi-objective Knapsack problem.

Multi-objective Knapsack Problem: The Multi-objective Knapsack Problem (MOKP) presents a unique challenge as it requires optimizing multiple conflicting objectives simultaneously, unlike the classic Knapsack problem that focuses on a single objective (Lust & Teghem, 2012). While the standard Knapsack problem involves selecting items with specific values and weights to maximize the total value without exceeding a weight constraint, the MOKP involves multiple knapsacks (objectives) where each item can contribute to different knapsacks with varying values and capacities.

Theorem 1. *The client selection problem in SFL, formulated as a Multi-objective Knapsack Problem, is NP-hard.*

Proof: To prove the NP-Hardness of our problem, we perform a reduction from the Knapsack Problem, specifically to the multi-objective Knapsack problem. Given an instance of the Knapsack Problem, with items, their weights, and values. We create an instance of our multi-objective optimization problem as follows:

1. *Items:* The clients in our problem correspond to the items in the Knapsack Problem, where a binary variable indicates whether a client is chosen in a round within a cluster.
2. *Weights:* The weights in our client selection problem represent selection cost. In our problem, this weight cost is related to whether a client has enough resources to finish a round, has a large data sample size, maximizes the number of unique labels of the selected client devices, the selection of the device reduces the total selected clients idle time and the client's selection is tracked to ensure its selection within a range of rounds.
3. *Values:* The characteristics of each client, such as the clients selected, their data sample size, their unique labels, their processing utilization and historical selection tracking, are used by the objective functions to determine their relevance to the fitness value, similar to the way item values are considered in the Knapsack Problem.

Thus, our objective in the multi-objective optimization problem is to maximize selected clients' fitness while adhering to the constraints outlined later in this section. By establishing this reduction, we conclude that our client selection problem is NP-Hard.

3.5.2 Problem Formulation

In this subsection, we will mathematically formulate our problem by discussing the input and output matrices to achieve an optimized client selection process and outline its objectives.

3.5.2.1 Input

Let $I \in \mathbb{N}$ denote the total number of devices with different capacities, and $X_c = (X_{c_1}, X_{c_2}, \ldots, X_{c_I})$ represent a set of client devices comprising I smartphones and IoT devices that have resources capacity features. The client-side model training process requires careful alignment of resource

Table 3.1 Frequently Used Notations

Notation	**Description**
$I \in \mathbb{N}$	Total number of devices
$K \in \mathbb{N}$	The total number of rounds
$k_c \in \mathbb{N}$	The number of clusters for the clustering model
$Z \in \mathbb{N}$	Total number of clusters
$J \in \mathbb{N}$	Number of filtered devices with dynamic available resources
X_c	Set of devices resources capacities
X_s	Set of selected clients
X_a	Set of filtered clients with dynamic available resources
X_u	Set of resources utilization estimation for devices
$X^K_{s_l,\zeta_z}$	Binary variable, 1 if client l is chosen in round K, 0 otherwise
ζ	Set of clusters
$\bar{X}$	The average processing utilization of selected clients
W_{fq}	Weight associated with objective function of index q
$F(x)$	Multi-objective optimization function
$\bar{\mathbf{w}}^{d,k}_{a_L,\zeta_z}$	Global aggregated weights at cluster z for client-side at round k
$\bar{\mathbf{w}}^{s,k}_{a_L,\zeta_z}$	Global aggregated weights at cluster z for split server-side at round k
G_{a_L,ζ_z}	Represents the filtered clients with available resources at cluster ζ_z at round k
$\Theta^{\in(mem,pro,dis)}$	Represent the prediction model of resource utilization

utilization with the model's specific requirements to run the model on the client side regardless of the data samples they possess. Removing clients with insufficient resources from the set I is important to ensure high round participation. As a result, the remaining set of clients, denoted as J, can be represented by:

$$J = \{i \in I : X^{\text{mem}}_{c_i} \geq M^{\text{mem}}_{c_i}, X^{\text{pro}}_{c_i} \geq M^{\text{pro}}_{c_i}, X^{\text{dis}}_{c_i} \geq M^{\text{dis}}_{c_i}\} \tag{3.1}$$

$X^{\textbf{mem}}_{c_i}$: Memory capacity of the client device.

$X^{\textbf{pro}}_{c_i}$: Processing unit capacity of client device.

$X^{\textbf{dis}}_{c_i}$: Hard disk capacity of client device.

$M^{\textbf{mem}}_{c_i}$: Memory loading requirement of the model.

$M_{c_i}^{\mathbf{pro}}$: Processing unit loading requirement of the model.

$M_{c_i}^{\mathbf{dis}}$: Hard disk loading requirement of the model.

The filtered client set J serves as input for the clustering component. Let $\zeta = \{\zeta_1, \zeta_2, \ldots, \zeta_Z\}$ denote the set of clusters, where Z is the total number of clusters, and X_{c_j,ζ_z} represents client j in cluster z.

Let us define X_a as a set of filtered clients with dynamically available resources. The challenge is to train a Random Forest model M on distributed, resource-constrained devices while accounting for varying resource demands and available data sample size($X_{a_j,\zeta_z}^{\mathrm{sam},k}$). We assume that the data sample size remains unchanged between the rounds. Moreover, the machine learning model's resource demands may differ across clients due to differences in input data requirements. Let K be the total number of rounds for training a machine learning model. Let us define X_{a,ζ_z}^{k} as the set of clients with dynamically available resources at round k in a cluster z.

Denote X_{u_j,ζ_z}^{k} as the estimated resource utilization of memory $X_{u_j}^{\mathrm{mem},k}$, processing power $X_{u_j}^{\mathrm{proc},k}$, and hard disk $X_{u_j}^{\mathrm{dis},k}$ for client j at round k within cluster z, during model training.

To ensure consistency and quality in the model's training, it is essential to filter out clients who lack sufficient resources and include only those who meet the minimum requirements. This process involves systematically removing inadequately resourced clients from the original set, denoted as J, resulting in a new set, denoted as L. By doing so, we can ensure that the remaining client set comprises individuals capable of contributing meaningfully to the model training process. Subject to the following constraints:

$$L = \{j \in J : X_{u_j}^{\mathrm{mem}} \leq M_{a_j}^{\mathrm{mem}}, X_{u_j}^{\mathrm{pro}} \leq M_{a_j}^{\mathrm{pro}}, X_{u_j}^{\mathrm{dis}} \leq M_{a_j}^{\mathrm{dis}}\} \tag{3.2}$$

$$X_{u_j}^{k} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{Tree}_t(X_{a_j,\zeta_z}^{k}) \tag{3.3}$$

Equation (3.3) establishes a relationship between observed resource use and the predicted value using a Random Forest regression model, where N is the number of decision trees in the ensemble, and $\text{Tree}_t(X^k_{a_j,\zeta_z})$ represents the prediction made by the t-th decision tree for the input features. The equation indicates that the final prediction $X^k_{u_j}$ is the average of predictions made by all decision trees in the Random Forest ensemble for the given input features. Finally, the input to our optimization problem is $X^k_{a_j,\zeta_z}$.

3.5.2.2 Output

The selection process output adheres to specific clients who can contribute effectively to the SL task by achieving predetermined objectives. We aim to maximize the selection in each round k by choosing a set of clients $X^K_{s_L,\zeta_Z}$, which contains binary values, where 1 represents if the client is selected and 0 otherwise. The list represents the number of compatible clients from L that adhere to the defined resource constraints and follow the objective functions.

3.5.2.3 Client Selection Objectives

The objective involves selecting a suitable group of clients (X^K_{s,ζ_z}) in every round k of a cluster ζ_z from a diverse pool of available clients ($X^k_{a_l,\zeta_z}$). Each client has a different resource profile, label and data sample size. Our objectives include the number of data sample sizes each client has ($X^{\text{sam},k}_{a,\zeta_z}$), the number of selected devices (X^k_{s,ζ_z}), the number of unique labels of the selected client devices $u(X^k_{s,\zeta_z})$, the differences in the processing capabilities of the selected clients $\mathbf{V}(X^{proc,k}_{u,\zeta_z})$ and finally to monitor the clients selected by tracking the historical client selected $hist(X^k_{s,\zeta_z})$ so that no client device is left out of the training process. Clients are chosen based on their varying resource limitations, and their security goals are maintained. Each client is assigned unique labels and is responsible for optimizing their participation while ensuring the continuous authentication model's maximum performance. Table 5.1 defines the summary of some important notations used.

We introduced a weighting mechanism based on the method of adjustable weights (Pardalos, Steponavičė & Žilinskas, 2012) to enhance flexibility in prioritizing different objective functions. These weights are decimal values ranging from 0 to 1; their sum always equals 1. This combination is expressed as follows:

$$W_{f1} + W_{f2} + W_{f3} + W_{f4} + W_{f5} = 1 \tag{3.4}$$

This allows us to assign varying importance to each objective while ensuring a balanced overall priority (Sami & Mourad, 2020). The clients selected need to adhere to the following objective functions:

1. Maximize the number of clients' devices:

$$f_1 = \max\left(\left(\sum_{l=1}^{L} X^k_{s_l,\zeta_z}\right) \cdot W_{f1}\right) \tag{3.5}$$

   - $\max_{X^k_{s,\zeta_z}}$ indicates that the maximization is performed over the set of selected clients in round k, within a cluster ζ_z, denoted by X^k_{s,ζ_z}.
   - $\sum_{l=1}^{L} X^k_{s_l,\zeta_z} \cdot W_{f1}$ represents the sum over all selected clients ($l = 1$ to L) in round l.
   - X^k_{s,ζ_z} is a set of binary variables indicating whether a client is selected or not in round k within a cluster z.
   - W_{f1} is the weight associated with the entire summation, indicating the importance of selecting clients.

   This selection is necessary to enable the model to accurately consider the differences in client features and labels. Therefore, having more selected clients can help speed up the model's process, as the data sample size between the client devices will be more significant. Clients' active participation in the learning phase contributes to a quick convergence of the model.

2. Maximize the number of unique clients:

$$f_2 = \max\left(\left(\sum_{l=1}^{L} u(X^k_{s_l,\zeta_z})\right) \cdot W_{f2}\right) \tag{3.6}$$

- $\max_{X^k_{s,\zeta_z}}$ indicates that the maximization is performed over the set of selected clients in round k, within a cluster ζ_z, denoted by X^k_{s,ζ_z}.
- $\left(\left(\sum_{l=1}^{L} u(X^k_{s_l,\zeta_z})\right) \cdot W_{f2}\right)$ represents the sum over all selected unique clients ($l = 1$ to L) in round k.
- X^k_{s,ζ_z} is a set of binary variables indicating whether a client is selected or not in round k within a cluster ζ_z.
- W_{f2} is the weight associated with the entire summation, indicating the importance of selecting clients.

Having more selected clients with different labels can help speed up the process of building the model. The active participation of clients in the learning phase ensures a thorough understanding and contributes to the swift convergence of the model. The model learned will differentiate between the different user's features.

3. Maximize the learning quality:

$$f_3 = \max\left(\left(\sum_{l=1}^{L} X^k_{s_l,\zeta_z} \cdot X^{sam,k}_{a_l,\zeta_z}\right) \cdot W_{f3}\right) \tag{3.7}$$

- $\sum_{l=1}^{L}\left(X^k_{s_l,\zeta_z} \cdot X^{sam,k}_{a_l,\zeta_z}\right) \cdot W_{f3}$ represents the sum over all selected clients ($l = 1$ to L) in round k, where $X^{sam,k}_{a_l,\zeta_z}$ is the sample size of client l in round k within a cluster z.
- W_{f3} is the weight associated with the entire summation, indicating the importance of selecting clients with larger sample sizes.

This ensures the model is exposed to a diverse and comprehensive set of training examples. The emphasis on selecting clients with high data samples is a strategic approach to increase the model's ability to generalize and make accurate predictions.

4. Minimize the client's idle time:

$$f_4 = \min\left(\left(\sum_{l=1}^{L} X^k_{s_l,\zeta_z} \cdot \mathbf{V}(X^{proc,k}_{u_l,\zeta_z})\right) \cdot W_{f4}\right) \tag{3.8}$$

- $\mathbf{V}(X_{u_l,\zeta_z}^{proc,k})$: Represents the variance of the processing utilization estimates among the selected clients. The variance can be calculated as follows: $\mathbf{V}(X_{u_l,\zeta_z}^{proc,k}) = \frac{1}{L}\sum_{l=1}^{L}\left(X_{u_l,\zeta_z}^{proc,k} - \bar{X}\right)^2$, where $\bar{X}$ is the average processing utilization calculated as $\bar{X} = \frac{1}{L}\sum_{l=1}^{L} X_{u_l,\zeta_z}^{proc,k}$.
- $X_{u_l,\zeta_z}^{proc,k}$: The processing utilization estimation of the model on client l in round k within a cluster z.
- W_{f4}: This is the weight associated with the entire objective function, indicating the importance of this objective in the overall optimization process.

This minimizes the variance of processing utilization estimates among the selected clients, encouraging the selection of clients with processing capabilities close to the average. This approach enhances the overall effectiveness of FL by ensuring that clients progress through rounds with minimal wait times.

5. Maximize the probability of a client being selected:

$$f_5 = \max\left(\left(\sum_{l=1}^{L} X_{s_l,\zeta_z}^{k} \cdot \text{hist}(X_{s_l,\zeta_z}^{k})\right) \cdot W_{f5}\right) \tag{3.9}$$

- $\text{hist}(X_{s_l,\zeta_z}^{k})$: Historical record for client l in round k within cluster ζ_z, where $\text{hist}(X_{s_l,\zeta_z}^{k}) = 1$ if client l was shown but not selected, and 0 otherwise.
- W_{f5}: The weight associated with the entire objective function, indicating the importance of this objective in the overall optimization process.

The function aims to engage a diverse range of participants who may have been overlooked. Additionally, maximizing client selection helps in improving model generalization and robustness. It encourages the participation of a broader range of data distributions and characteristics, thereby improving the overall quality of the learning rounds.

The optimization problem can be formulated as follows:

$$\begin{aligned}
&\text{Maximize} \quad \sum_{l \in X_a} X^K_{s_l,\zeta_z} \cdot W_{fq} \\
&\text{where} \\
&\quad X^K_{s_l,\zeta_z} \in \{0,1\} \quad \forall l, z, K \\
&\quad \sum_{q=1}^{Q} W_{f_q} = 1
\end{aligned} \tag{3.10}$$

The multi-objective optimization problem is represented as follows:

$$F(x) = f_1(x) + f_2(x) + f_3(x) - f_4(x) + f_5(x) \tag{3.11}$$

Where:

$f_1(x)$ = Number of active devices

$f_2(x)$ = Number of unique clients

$f_3(x)$ = Quality of learning

$f_4(x)$ = Reduction in device idle time

$f_5(x)$ = Prioritize new devices

3.5.3 Genetic Algorithm For Client Selection Problem

Multi-objective optimization problems inherently feature multiple solutions rather than a single optimal one, and these solutions are referred to as Pareto solutions (Murata, Ishibuchi *et al.*, 1995). Acquiring the Pareto set solution efficiently is crucial. A genetic algorithm (GA) proves advantageous in addressing this challenge due to its ability to explore the search space efficiently. The GA mirrors the natural selection process, favouring the reproduction of the fittest set of solutions for subsequent generations (Konak, Coit & Smith, 2006). Non-GA approaches, with their expansive search spaces as reported in Figure 3.5, require thorough exploration, often demanding significant computational resources and time to identify optimal solutions efficiently, as the number of possible solutions is exponential to the increased number of clients. Each chromosome is represented as a $X^k_{s_{\zeta_z}}$ matrix and signifies the decision made by the optimization model regarding the selection of the client for model training. Mathematically, $X^k_{s_l} \in [0, 1]$. This representation captures the binary nature of the decision, facilitating the optimization process within the GA framework. Figure 3.6 gives an overview of the genetic process. It starts with the initialization of the population and goes through the various stages of selection, Crossover, Mutation, and fitness evaluation. This cycle is repeated iteratively to carry out the GA operations.

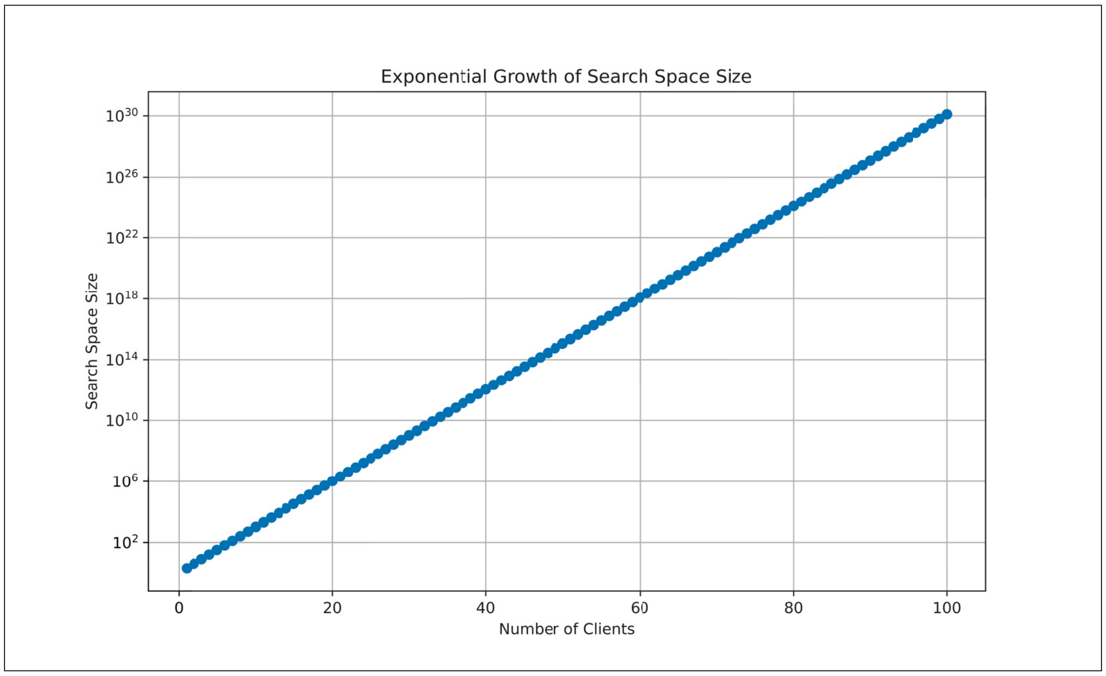

Figure 3.5 Search Space Size for Client Selection with Multi-Objective Optimization

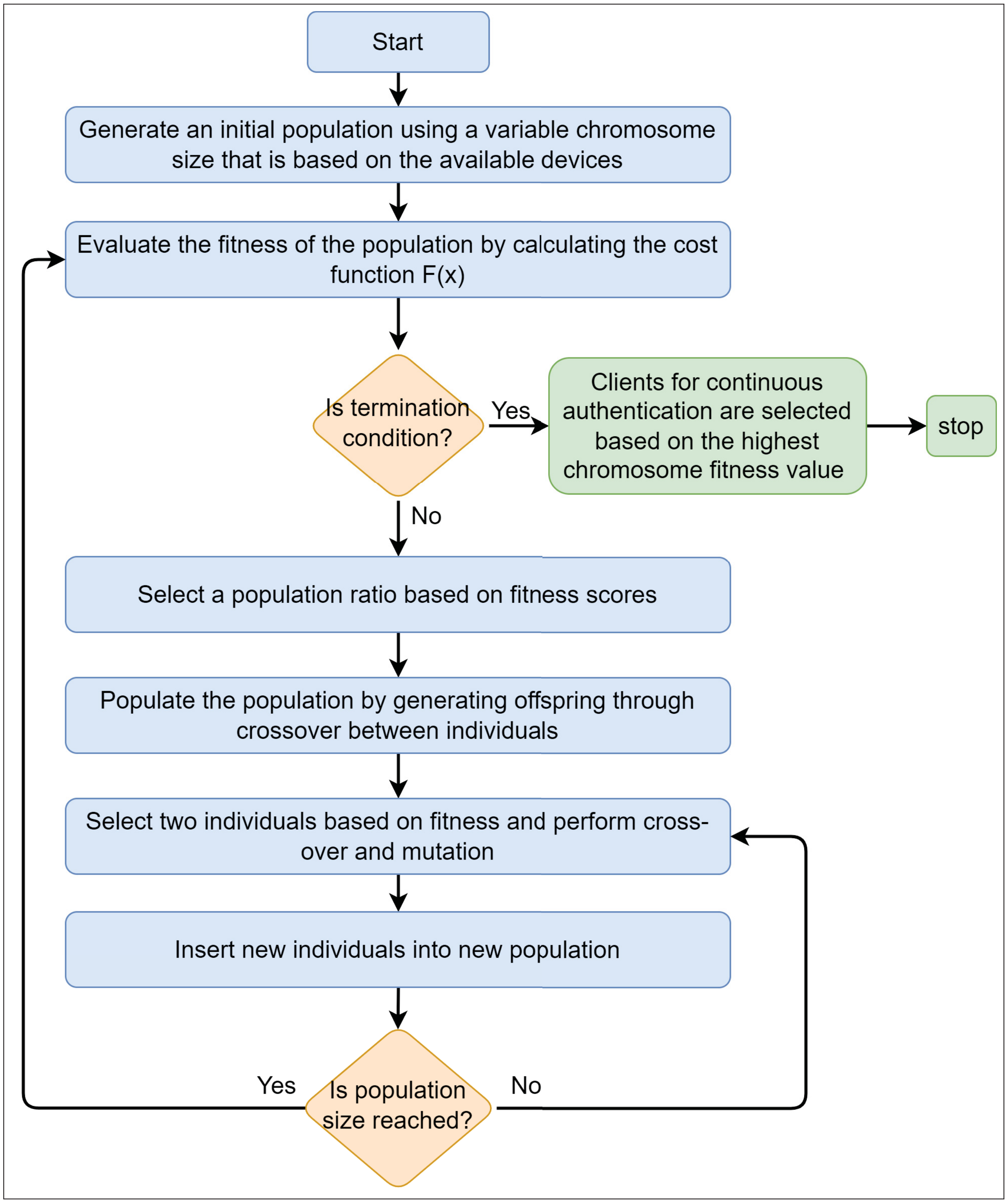
Start
Generate an initial population using a variable chromosome size that is based on the available devices
Evaluate the fitness of the population by calculating the cost function F(x)
Is termination condition?
Yes
Clients for continuous authentication are selected based on the highest chromosome fitness value
stop
No
Select a population ratio based on fitness scores
Populate the population by generating offspring through crossover between individuals
Select two individuals based on fitness and perform cross-over and mutation
Insert new individuals into new population
Yes
Is population size reached?
No

Figure 3.6 Genetic Algorithm Steps for Client Selection in Continuous Authentication

To efficiently solve a mixed-integer programming (MIP) problem, it is essential to analyze its time complexity and break it into various components. By doing so, we can identify the most computationally intensive parts of the problem.

In the *Initialization* phase, we generate a group of potential solutions (chromosomes) based on the total number of clients available for training. Our first step is to create an initial population of chromosomes utilizing a set of genes. Each gene is tailored to the specific problem we are addressing. In our case, a gene represents a client model within a chromosome encompassing a subset of clients. To create a chromosome, we randomly select genes, ensuring no duplicate genes are within the same chromosome.

During The *Selection* process, individuals are chosen from the population to become parents for the next generation. To filter the population, we select the best half before proceeding with Crossover and Mutation to create the new generation.

Crossover is a process that combines genetic material from two parent solutions to produce new offspring solutions. During Crossover, two chromosomes are randomly split at a position, and their segments are swapped when a randomly generated float number between 0 and 1 exceeds a predefined crossover ratio of 0.5 specified by the GA configuration.

Mutation is a process that introduces random changes in offspring solutions to maintain diversity. Offspring solutions replace some individuals in the current population to form the next generation.

The algorithm's *Termination* criteria determine when to halt the optimization process. These criteria are based on factors such as reaching a maximum fitness threshold or completing a specified number of generations. Genetic selection results in a chromosome that includes clients whose combined selection meets the predefined objectives.

To measure the complexity of the optimization process, we use N to represent the total number of decision variables, G to represent the number of Generations in the optimization process, L to denote the number of clients selected at each round, and Q to signify the number of objective functions. This complexity is comprised of various components, including:

$C_{\textbf{initialization}}$

Complexity of the initialization step: $O(N \times L)$.

$C_{\textbf{selection}}$

Selection process complexity: $O(N)$.

$C_{\textbf{crossover}}$

Crossover complexity over rounds: $O(N \times K)$.

$C_{\textbf{mutation}}$

Mutation complexity over rounds: $O(N \times K)$.

$C_{\textbf{fitness}}$

Complexity of calculating the multi-objective optimization function: $O(N \times J \times Q)$

Thus, the overall time complexity can be expressed as $O(G \times (C_{\text{initialization}} + C_{\text{selection}} + C_{\text{crossover}} + C_{\text{mutation}} + C_{\text{fitness}}))$.

3.6 CRSFL Framework

The process of our proposed architecture lays out the specific steps and procedures we use to ensure that the neural network model is trained across different clusters of clients. The method outlined in the prior section is followed to process the input of Algorithm 3.1. This involves carrying out client filtering, model prediction, and K-means clustering. Next, we will describe the presented pseudo-code for the proposed framework:

1. *Client Selection for training:* To begin training, we select the initial cluster of clients with the highest resource capacities. This ensures that the initial training is done with the most capable clients and that the dropout rate will be minimal compared to other clusters. We evaluate each client selected with available resources and data samples to train their respective client-side model. The SL server then requests client information about their available resources and the number of data samples. Algorithm 3.1 refers to this step in lines 1 $\rightarrow$ 8. Upon receiving this information, the server selects a combination of clients

Algorithm 3.1 Proposed Clustered-based Resource-aware Split Federated Learning (CRSFL) Algorithm.

```
   Input: $X_{a,\zeta_Z}, \Theta^{\in(\text{mem,pro,dis})}, K$
   Output: $\bar{\mathbf{w}}^{d,k}_{a_J,\zeta_z}, \bar{\mathbf{w}}^{s,k}_{a_J,\zeta_z}$
1  for $k$ in $K$ do
2      for $X^k_{a,\zeta_z}$ in $X^k_{a,\zeta_Z}$ do
3          $L = 0$
4          collect $X^{\text{mem},k}_{a_j,\zeta_z} \wedge X^{\text{pro},k}_{a,\zeta_z} \wedge X^{\text{dis},k}_{a_j,\zeta_z} \wedge X^{\text{sam},k}_{a_j,\zeta_z}$
5          $X^{\in(\text{mem,pro,dis}),k}_{u_j,\zeta_z} = \Theta^{\in(\text{mem,pro,dis})} = X^{\in(\text{mem,pro,dis}),k}_{a_j,\zeta_z}$
6          if $X^{mem,k}_{a_j,\zeta_z} \geq X^{mem,k}_{u_j,\zeta_z} \wedge X^{pro,k}_{a_j,\zeta_z} \geq X^{pro,k}_{u_j,\zeta_z} \wedge X^{dis,k}_{a_j,\zeta_z} \geq X^{dis,k}_{u_j,\zeta_z}$ then
7              $L+ = 1$
8              add $X^k_{a_j,\zeta_z}$ to $G_{a_L,\zeta_z}$
9          end if
10         $X^k_{s,\zeta_z}$ = ga_selector($G^k_{a,\zeta_z}$)
11         for $X^k_{s_l,\zeta_z}$ in $X^k_{s_L,\zeta_z}$ do
               // Client loads device-side global model
12             ($X^k_{s_l,\zeta_z}$).load_state_dict($\bar{\mathbf{w}}^{d,k}_{a_L,\zeta_z}$)
               // server loads server-side global model
13             server_weights($X^k_{s_l,\zeta_z}$).load_state_dict($\bar{\mathbf{w}}^{d,k}_{a_L,\zeta_z}$)
14             for e in range(epochs) do
15                 client_model_output = $M^{d,k}_{X_{s_l,\zeta_z}}$(client_data)
16                 smashed_data = client_model_output.requires_grad_(True)
17                 server_model_output = $M^{s,k}_{X_{s_l,\zeta_z}}$(smashed_data)
18                 loss = criterion(server_model_output, labels)
19                 loss.backward()
20                 client_grad = smashed_data.grad
21                 $M^{d,k}_{X_{s_l,\zeta_z}}$.backward(client_grad)
22             end for
23             send $M^{d,k}_{X_{s_l,\zeta_z}}$.state_dict() to fed_server
24         end for
           // Fed_Server updates global_client_aggregated_weights
25         $\bar{\mathbf{w}}^{d,k}_{s_L,\zeta_z} = \frac{1}{L}\sum_{l=0}^{L}(X^k_{s_l,\zeta_z}\mathbf{w}^{d,k}_{s_l,\zeta_z})$
           // Split_Server updates global_server_aggregated_weights
26         $\bar{\mathbf{w}}^{s,k}_{s_L,\zeta_z} = \frac{1}{L}\sum_{l=0}^{L}(X^k_{s_l,\zeta_z}\mathbf{w}^{s,k}_{s_l,\zeta_z})$
27     end for
28 end for
```

to maximize the fitness value collectively (line 9). This selection of clients is an NP-hard problem, as previously explained. Therefore, a heuristic approach is employed on the server for client selection, using a genetic algorithm to choose the best-suited clients to achieve the objectives set by the server. The selected participants then receive the global client-side neural network model from the federated server and train each on their data in parallel.

2. *Parallel Client-side Training:* Client clusters use a collaborative training approach where each client trains their model layers simultaneously, transmitting only the gradients of the cut layer. This is accomplished by processing individual sample data up to the cut layer of the neural network model through a forward pass. However, marginal errors may impact clients, such as dropping out due to a sudden increase in resource usage (Li *et al.*, 2022). Resource fluctuations could prevent clients from transmitting updated gradient information to the split server or cause them to go offline and miss receiving gradient information from the server to update their weights (lines 10 $\rightarrow$ 15).
3. *Parallel Server-side Training:* In parallel, the SL server simultaneously receives updates to the weights from the client side, trains the remaining layers of the model separately for each client, and continues the forward pass initiated by the client. The server calculates the model output to determine the learning loss rate once the forward pass is finished. After evaluating the loss, the SL server performs back-propagation through the remaining layers to calculate gradients for the parameters in those layers. The server then transmits these updated gradients to the corresponding clients, enabling them to modify their local parameters and proceed with the training process (lines 16 $\rightarrow$ 21).
4. *Weights Aggregation:* After undergoing multiple epochs of SL training, the SL process now involves a matrix that includes the client ID device and its server-side gradients. The split server then aggregates these weights to create a new global server-side model for the cluster. Additionally, the federated server acts as an aggregator for the client-side models, which may be the same split server or a new server component to enhance privacy and prevent complete model weights from being held by a single entity. The federated server performs a weighted aggregation for the client-side model to create a new global model for the same cluster. Finally, the Federated and Split servers send the aggregated client-side

model to the clients in the subsequent cluster and use the aggregated server-side model for the upcoming SL training. This guarantees that the clients in the following cluster will have some pre-trained model weight for both parts of the model (lines 22 → 23).

3.7 Experimental Setup

In the following section, we will provide details on our datasets, experimental setup, the experiments we performed, and the results we obtained from these experiments.

We conducted the experiments using the modular FL framework outlined in (Arafeh *et al.*, 2023a). In a Windows environment, our experiments were performed on $Pytorch^1$. Furthermore, $Docker^2$ containers were employed to simulate the cluster resources profile, with memory being restricted to 2GB for every container and varying CPU capacities. We utilized a machine equipped with eight-core processors with a Base speed of 3.59GHz and 32GB of RAM, and we employed different CPU counts of 1.5, 2, and 2.5 as CPU capacity for devices.

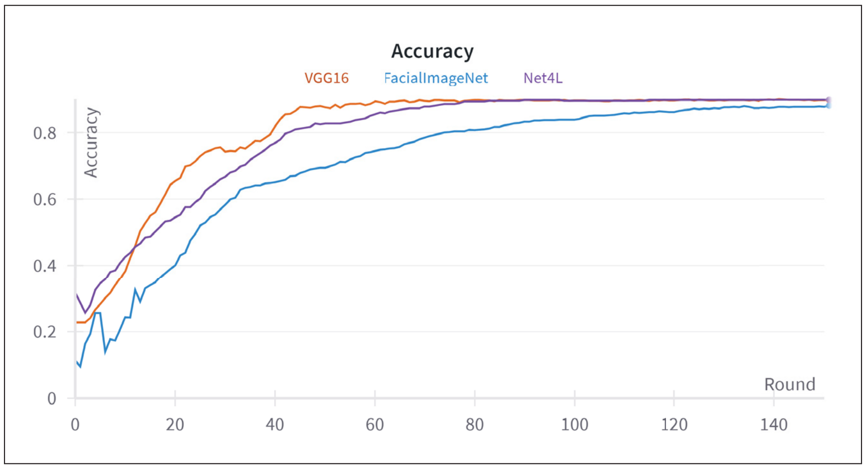

Figure 3.7 Accuracy Metric Performance for Different Machine Learning Models

We customized the configuration for our authentication use case and tested three different multi-class classification machine learning models in which we show the benchmark results in Figure 3.7. The figure shows a VGG16 model with 16 layers. In comparison, the FacialImageNet model uses five layers with two convolutional layers. Finally, the Net4L contains four layers that produce comparable levels of accuracy as the VGG16 model while requiring less processing power. The model requires 128*128 as input and 128*3 as hidden layers. For simplicity, we divided the model into half for the client and the server for SL. Following a thorough hyperparameter configuration process, we identified the optimal settings for our experiments. Our chosen optimizer was stochastic gradient descent, with a fixed learning rate of 0.01 and a momentum of 0.9. We also utilized the cross-entropy loss function. The model parameters were aggregated by averaging them in the FL experiment. Furthermore, the data split ratio was set to 8:2 for training and testing. We referred to the work of (Li & Lyu, 2023) when selecting the configuration for neural network training. In addition, we visualized the resulting plots using the Weights & Biases framework (Biewald, 2020).

3.7.1 Dataset

We conducted various experiments to evaluate the performance of our proposed approach for continuous authentication using a real-life facial authentication dataset.

UMDAA-02-FD Filtered: The UMDAA-02-FD Face Detection dataset (Mahbub, Sarkar, Patel & Chellappa, 2016b) presents a range of challenges for vision algorithms due to its diverse range of images, which includes partially visible faces, various lighting conditions, occlusions, and facial emotions. The dataset *umdaa*02[3] comprises 33,209 photos taken at 7-second intervals from 44 users, making it an ideal choice for continuous authentication scenarios. We processed the dataset to a fixed size 128x128 to reduce the computational burden. We created a custom dataset from the original data of UMDAA-02-FD to hold the valid user training samples. The new dataset is processed through several cleaning steps, ensuring that only the valid training images are considered for training; we removed two users from the initial set as they were sharing some familiar facial photos between them. We perform the following steps to process the data:

1. A sample of images falling below a specified brightness threshold was initially removed. We experimented with various threshold values for the brightness (ranging from 5 to 25), ultimately setting 15 as the threshold criterion.

2. We employed the Laplacian variance method with various predefined thresholds, determining that a threshold of 15 most effectively identified and eliminated blurred images (Bansal, Raj & Choudhury, 2016).
3. We converted images to gray-scale by applying data normalization using the minimum and maximum values of the input data followed by histogram equalization using the tools from $OpenCV^4$.
4. The images were then resized to 128x128 pixels for more efficient machine-learning training. It is worth noting that these preprocessing steps led to the removal of nearly 7% of inadequate training samples from the dataset and two users with shared data, thereby enhancing its overall quality and suitability for subsequent analysis.

Our experiments explored various dynamic configurations to mirror real-world scenarios where each device contains a portion of user data. We randomly assigned varying devices (ranging from 2 to 6) to handle the data for each unique client label, which simulates the diverse and variable participation of devices in real-world settings and underscores the complexity of data distribution. This complexity highlights the need for innovative solutions in data analysis and machine learning. To maintain the non-IID (non-Independent and Identically Distributed) nature of the data, we ensured that the number of data samples per device varied, further accentuating the variability. The data for each user was distributed to reflect non-IID characteristics. We allocated data points to each device using a data generator, randomly selecting between 100 and 150 data points per device. This range provides the desired variability while keeping the data distribution within a manageable scope. The total number of data samples for each user naturally varied due to the dataset's inherent non-IID characteristics, thus underscoring our experiment's complexity and real-world relevance.

In Figure 3.8, we depicted the distribution of user data. The bar chart in the figure displays the total number of data samples per label, accentuating the data's non-IID (non-identically distributed) nature. The chart distinctly illustrates user data distribution, highlighting the unique case of non-IID data distribution. Each client label (0 to 41) denotes a distinct user in the dataset. Random subsets of devices (2 to 6) were allocated to manage the data for each label. The data generator randomly allocated between 100 and 150 data points to each assigned device, ensuring a random distribution of data per device and preserving the non-IID nature of the data. The allocation of data points per device also mirrors users' varying data generation rates, simulating real-world scenarios where some users produce more data than others.

The non-IID nature played a crucial role in our experiments, as it closely resembles real-world conditions where data collected from different devices and users is often diverse. This diversity is a key challenge in distributed machine-learning systems. To ensure that each device had a unique data distribution, we used a data generator to select data points within a specified range randomly. Our experimental setup involved dynamically configuring devices to handle user data portions, with random assignments ensuring non-IID distribution. The visualization presented in Figure 3.8 further illustrates the data distribution, highlighting the distinctive characteristics of our non-IID dataset. These details clarify our experimental design and the steps to maintain realistic data distribution scenarios.

3.7.2 Experiments

Our experiments show different configurations when training a machine-learning model. We focus on accuracy by comparing the centralized experiment, in addition to metrics related to the number of dropping clients, the training time, traffic required, and total idle time.

Centralized Training (Cen): The concept of centralized experiments is widely used in machine learning. It involves transmitting the raw data to a server, where a centralized model is trained on the entire data.

Split Learning (SL): SL enables private machine learning by dividing the model between the client and server. This way, client data remains secure, and only model weights are transmitted during the local data training process. The experiment is performed sequentially, meaning the selected clients must wait for each other to complete the training before starting. To ensure a consistent training latency, we allocated a dedicated training latency budget exclusively for this experiment, matching the CRSFL experiment.

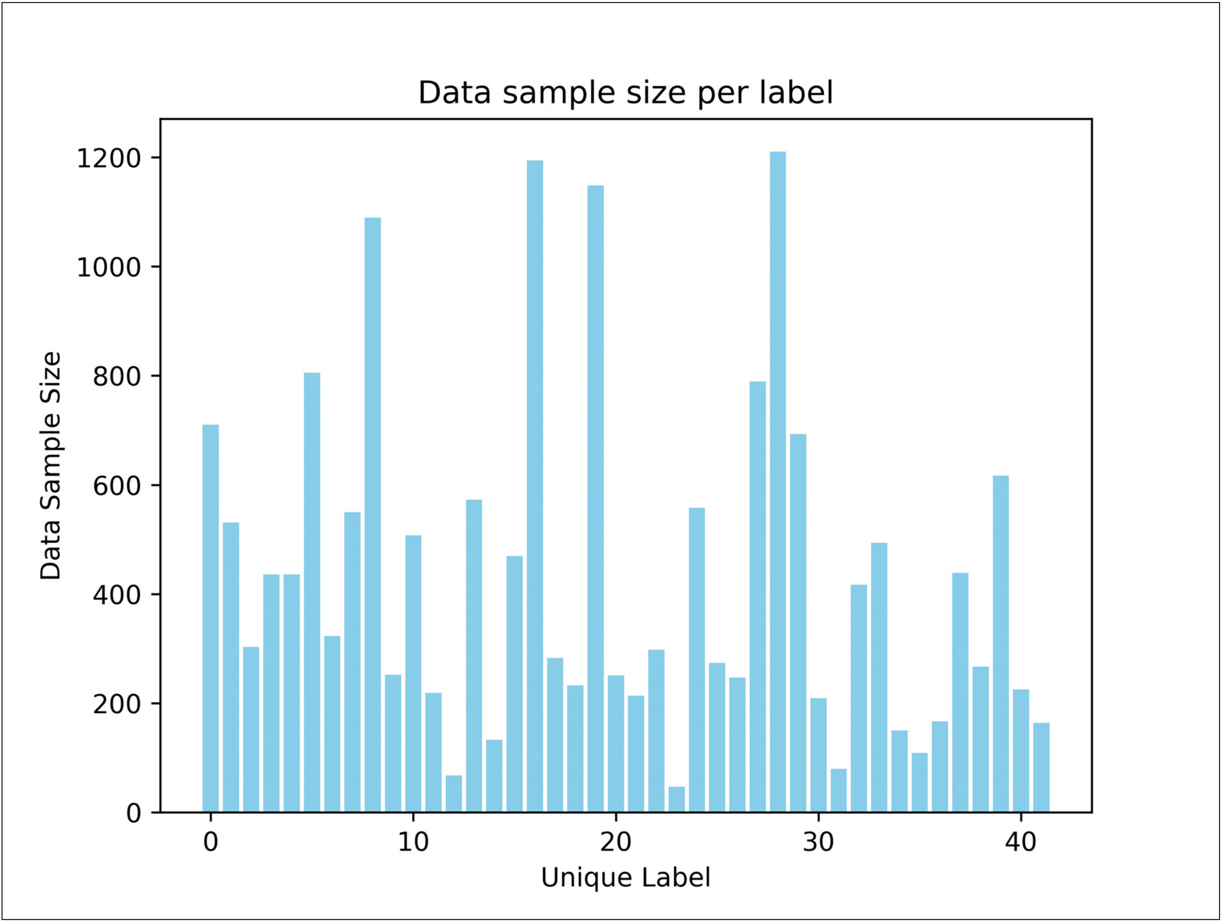

Figure 3.8 Data Sample Size per Client in UMDAA02-FD Dataset

Federated Learning (FL): The FL experiment implements a distributed machine-learning algorithm between client devices and an FL server. In this setup, each client independently trains the entire machine-learning model on their device and then shares the weights with the server. The server aggregates the models and sends back a single global model to the clients, completing a federated learning round.

Clustered Split Federated Learning (CSFL): We applied a CSFL algorithm as reported in (Wazzeh *et al.*, 2023). The approach does not consider client selection optimization; therefore, the clients are randomly selected in every cluster.

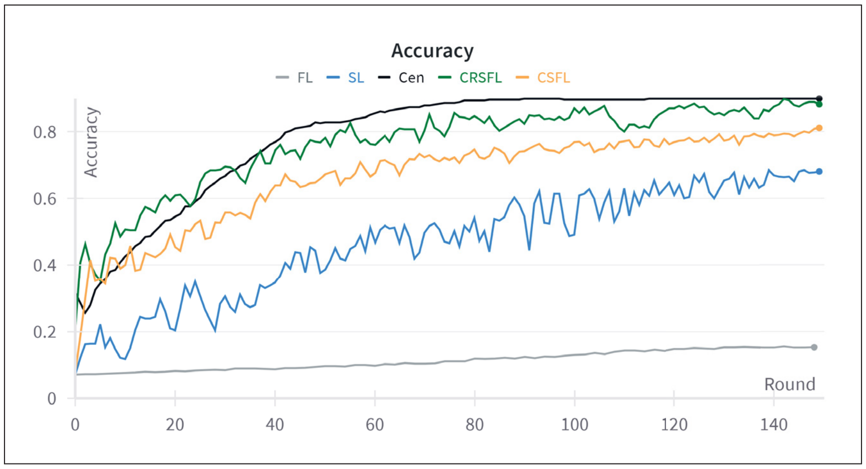

Figure 3.9 Test Set Accuracy per Round

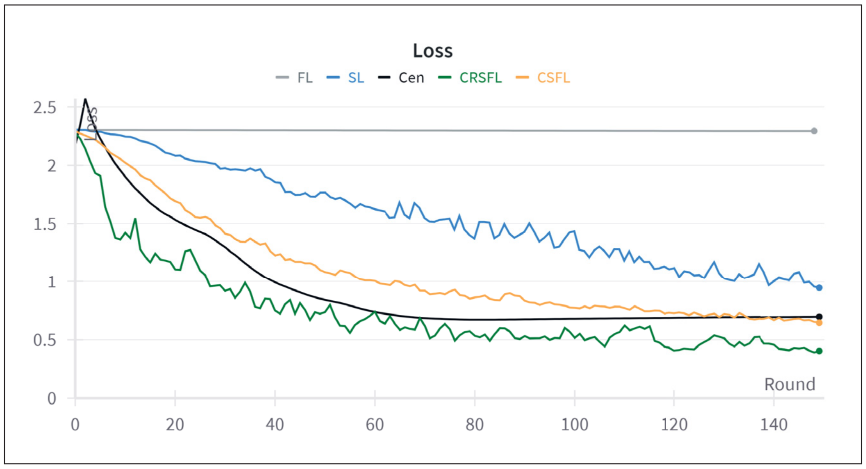

Figure 3.10 Test Set Loss per Round

Clustered Resources Split Federated Learning (CRSFL): We applied our proposed approach in this experiment, where the clients are selected from every cluster in every round based on the availability of the resources they each have. Client selection is optimized by using a heuristic GA approach, which is based on some predefined objectives.

3.7.3 Results

3.7.3.1 Performance Metrics: Accuracy and Loss

Figures 5.4 and 5.5 show the accuracy test results and loss for UMDAA02-FD over communication rounds. Starting with the centralized approach labelled "Cen," the model's convergence has reached an impressive 90% accuracy despite the dataset's significant challenges. However, CRSFl, our proposed approach, which utilizes GA selection, achieves a similar level of accuracy at 89% after 150 rounds. In contrast, the random selection approach, "CSFL," progresses steadily but remains at 80% after 150 rounds. The FL experiment could not converge after 150 rounds and remained under 20% accuracy and a high loss equal to 2.4. Finally, the split learning approach represented by "SL" slowly converges and only reaches 65% at the end of training. The slower convergence of SL and CSFL is due to their inefficient client selection processes and failed resource constraints resulting from random selection.

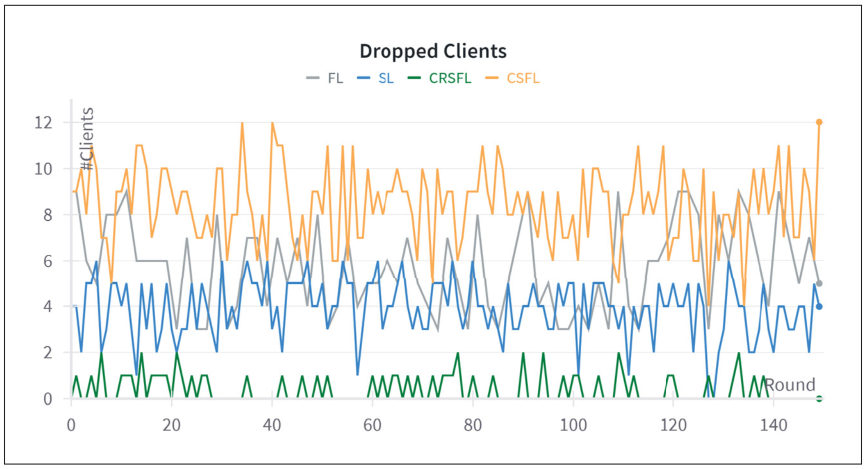

Figure 3.11 Total Number of Dropped Clients per Round

3.7.3.2 Client Dropout

Figure 3.11 discusses the number of clients dropping due to lacking resources for completing the training process; the count of the clients represents the total sum of the devices unable to

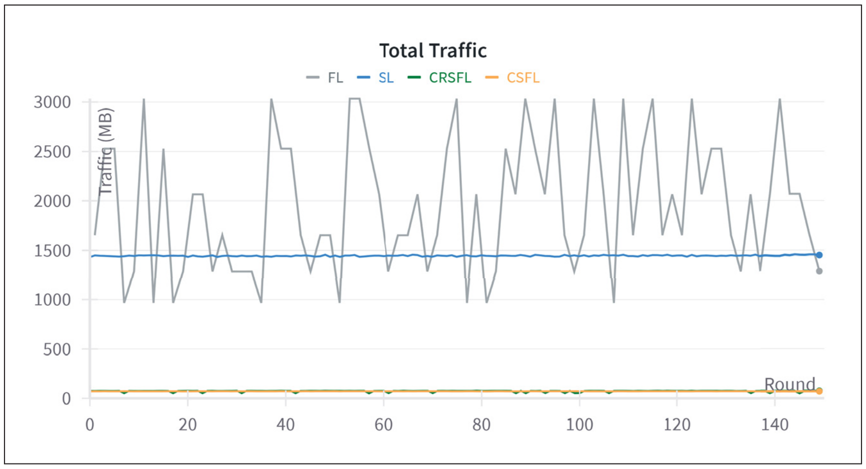

Figure 3.12 Clients Total Traffic Consumption in Megabytes (MB) per Round

complete the training process. The figure shows that the CSFL, which relies on random client selection, had the highest number of dropping clients, with an average of 8.5 clients dropping out per round. This is because the poor selection of clients who lacked the necessary resources resulted in their inability to complete the model training with the server. Conversely, with its sequential training process and fewer client selections in each round, the SL experiment had fewer clients, with an average of 4 clients dropping out per round, similar to the FL experiment. Finally, CRSFL has significantly fewer dropped clients than SL and CSFL due to our approach's careful selection of clients with adequate resources to complete their training. An average of 0.4 clients drop out per round. However, this small percentage of clients still drop out due to unexpected changes in their resources during model training.

3.7.3.3 Traffic Analysis

In Figure 3.12, the total traffic of clients in megabytes (MB) is shown during the exchange of gradients/weights with the servers to measure communication overhead. CRSFL and CSFL exhibit comparable traffic overhead (73 MB) because the clients only share their smashed data part of the model, and the global server-side model updates only after all clients have completed their parallel training of the model. However, FL and SL exhibit significantly higher traffic consumption. FL recorded an average of 2500 MB due to the need for each client to send the entire model weights and exchange them with the updated global model from the server. On the other hand, SL recorded 1500 MB because of vanilla SL's sequential order, where each client trains its weight and then transmits its model-side weights to the next client, significantly impacting the information exchanged between clients and servers.

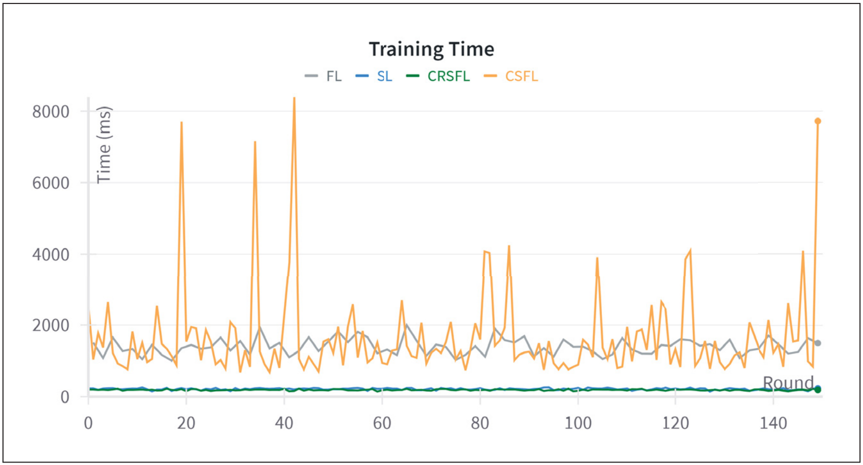

Figure 3.13 Clients Total Training Time in Milliseconds (ms) per Round

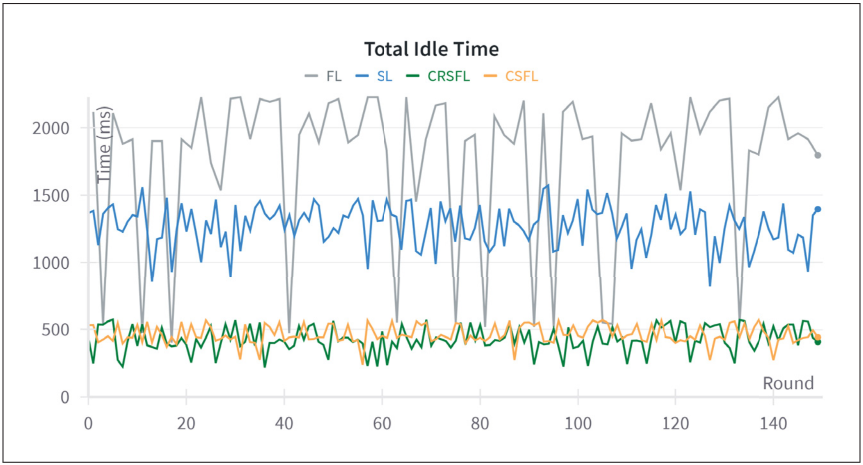

Figure 3.14 Clients Total Idle Time in Milliseconds (ms) per Round

3.7.3.4 Model Training Time

In the graphs provided, Figure 3.13 showcases the total training time for the selected clients per round. In contrast, Figure 5.9 illustrates the total idle time per round among the chosen clients during model training. The data indicates that SL and CRSFL demonstrate comparable training times, with SL averaging 250 ms and CRSFL averaging 200 ms. However, CSFL experiences variability in training times due to stragglers that delay model training. On average, CRSFL experiences 100ms less idle time than CSFL. This is because both clients simultaneously train the same section of the model. CRSFL strategically clusters devices with similar capabilities and employs a GA approach to select clients, minimizing the total idle time between them during training. The idle times in SL and FL are significantly higher, averaging 1300ms and 2000ms, respectively. This is due to clients having to wait for each other to finish the training process in the SL experiment, while in the case of FL, clients need to send their updated weights to the server for aggregation.

Upon analyzing the data presented in the plots, our proposed approach demonstrates high accuracy in model convergence. It ensures reduced training time, minimal client idle time, and a low dropout rate. These results indicate the approach's effectiveness in minimizing client idle time while maintaining high accuracy. We recognize that integrating multiple techniques can add complexity to the implementation process. However, our CRSFL framework utilizes distributed and clustering techniques to alleviate the client-side computational burden by transferring heavy computation to the server side, assuming it has significant computational power and can accommodate many clients. By partitioning the model between the client and the server, our approach alleviates the computational burden on the client side.

3.8 Conclusion

This paper presents a new technique for continuous authentication for mobile and IoT devices to verify user identity of the user continuously in the background. We proposed a Cluster-Based Resource-aware Split Federated Learning framework tailored for IoT-constrained devices. By prioritizing users' privacy through on-device data training and addressing critical challenges such as resource utilization optimization, client dropout mitigation, and efficient client selection, CRSFL offers a complete solution for continuous user authentication in IoT environments. By introducing novel techniques such as filtering methodologies, clustering based on resource capacities, machine learning-based resource prediction, and heuristic client selection algorithms, we have demonstrated improvements in reducing client dropping rates, minimizing training idle time, and enhancing the efficiency of FL processes. Our extensive experiments on real-life authentication dataset further validate the effectiveness of CRSFL, showcasing its superiority over existing methods for continuous authentication.

Future research aims to improve accuracy by exploring advanced model architectures like transformer networks. We strive to study the overhead burden on the client and the server while minimizing communication latency. Additional privacy-preserving techniques, such as differential privacy or secure multi-party computation, may also be investigated.

CHAPTER 4

A CONTINUOUS AUTHENTICATION APPROACH FOR MOBILE CROWDSOURCING BASED ON FEDERATED LEARNING

Mohamad Wazzeh[1,2] , Hakima Ould-Slimane[3] , Chamseddine Talhi[1] , Azzam Mourad[4,2] , Mohsen Guizani[5]

[1] Department of Software and IT Engineering, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Artificial Intelligence and Cyber Systems Research Center, Department of Computer Science and Mathematics, Lebanese American University, Chouran, Beirut 1102 2801, Lebanon
[3] Department of Mathematics and Computer Science, Université du Québec à Trois-Rivières, 3351 boulevard des Forges, Trois-Rivières, Québec, Canada G8Z 4M3
[4] KU 6G Research Center, Department of Computer Science, Khalifa University, Shakhbout Bin Sultan St, Hadbat Al Za'Faranah, Zone 1, Abu Dhabi, United Arab Emirates
[5] Mohammad Bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates

Paper published in IEEE Access, November 2024

4.1 Abstract

With the widespread use of smartphones and wearable devices, Mobile Crowdsourcing (MCS) has become a powerful method for gathering and processing data from various users. MCS offers several advantages, including improved mobility, scalability, cost-effectiveness, and the utilization of collective human intelligence. However, ensuring the authenticity of users throughout the data collection process remains a challenge. Current authentication methods, such as traditional PIN codes, two-factor authentication, and biometric authentication, often struggle to provide continuous verification while adequately protecting user privacy. This paper addresses this issue by proposing a new continuous authentication approach based on Federated Learning. This approach combines continuous identity verification with privacy preservation benefits, allowing for the ongoing validation of user authenticity during data collection while improving authentication accuracy. We also discuss the non-Independently and Identically Distributed issue in Federated Learning and employ transfer learning techniques based on feature

extraction to enhance the performance of the authentication models. We conducted extensive experiments using various datasets to evaluate the effectiveness of our proposed method. The results of this study demonstrate its potential to enhance the security and privacy of MCS systems.

4.2 Introduction

The utilization of Mobile Crowdsourcing (MCS) has emerged as a prominent approach for acquiring data from a diverse user base due to the pervasive presence of smartphones. MCS is a distributed data collection method that harnesses the capabilities of IoT devices to gather and process information from many participants, often referred to as the "crowd." MCS relies on crowdsourcing principles, which involve outsourcing tasks traditionally performed by employees or machines to a large, undefined group of people, typically via the Internet (Yang *et al.*, 2015). In the context of MCS, mobile devices play a central role, enabling individuals to contribute data through various built-in features such as sensors, GPS, cameras, and internet connectivity. MCS enables real-time, location-based, and context-aware data collection, supporting various applications such as urban planning, environmental monitoring, health tracking, and more. This approach allows organizations to access vast quantities of diverse data and encourages user participation in problem-solving and decision-making processes. With the widespread adoption of smartphones worldwide, MCS enables individuals to actively participate in data collection and problem-solving actively, bridging the gap between organizations and the crowd. Moreover, the advanced capabilities of mobile devices, such as built-in sensors, high-resolution cameras, and processing power, enable users to provide heterogeneous and rich data, including location information, multimedia content, and real-time feedback, thereby enhancing the value and utility of MCS. However, in MCS environments, the lack of reliable methods for verifying user authenticity poses substantial risks. Without confirming the credibility of users contributing data, the system becomes susceptible to malicious actors who could insert false or deceptive information, undermining the integrity and trustworthiness of the crowdsourced data (Yang *et al.*, 2015). Additionally, inadequate user authentication can lead to unauthorized access to sensitive data, potentially resulting in privacy violations and reputational harm for users and service providers (Feng *et al.*, 2017).

To further demonstrate the practical effectiveness and adaptability of the proposed method, we explore its application in real-world scenarios. One such scenario is in smart cities, where large-scale IoT infrastructures collect data for public services. Continuous authentication is essential for maintaining the integrity of sensitive data, and our framework effectively authenticates users in such environments. Another application is in healthcare, where wearable devices collect sensitive patient data. Our privacy-preserving federated learning approach can provide continuous, secure authentication without compromising data privacy. These case studies illustrate the versatility of our framework and its ability to handle real-world data challenges.

Nevertheless, verifying user identity legitimacy throughout the data collection poses a notable challenge. Conventional authentication methods rely on continuous validation assistance from users, which introduces vulnerabilities for unauthorized access and potentially fraudulent activities. The current authentication methods, such as token-based or password-based authentication, are susceptible to various types of attacks, including credential theft and session hijacking, which can lead to unauthorized access to the system (Mourad & Jebbaoui, 2015; Jebbaoui *et al.*, 2015). Furthermore, traditional authentication methods do not address the continuity of session authenticity. Leaving an unattended session could allow a malicious user or system takeover, putting the device and its private user data at risk (Hammoud, Otrok, Mourad, Wahab & Bentahar, 2018).

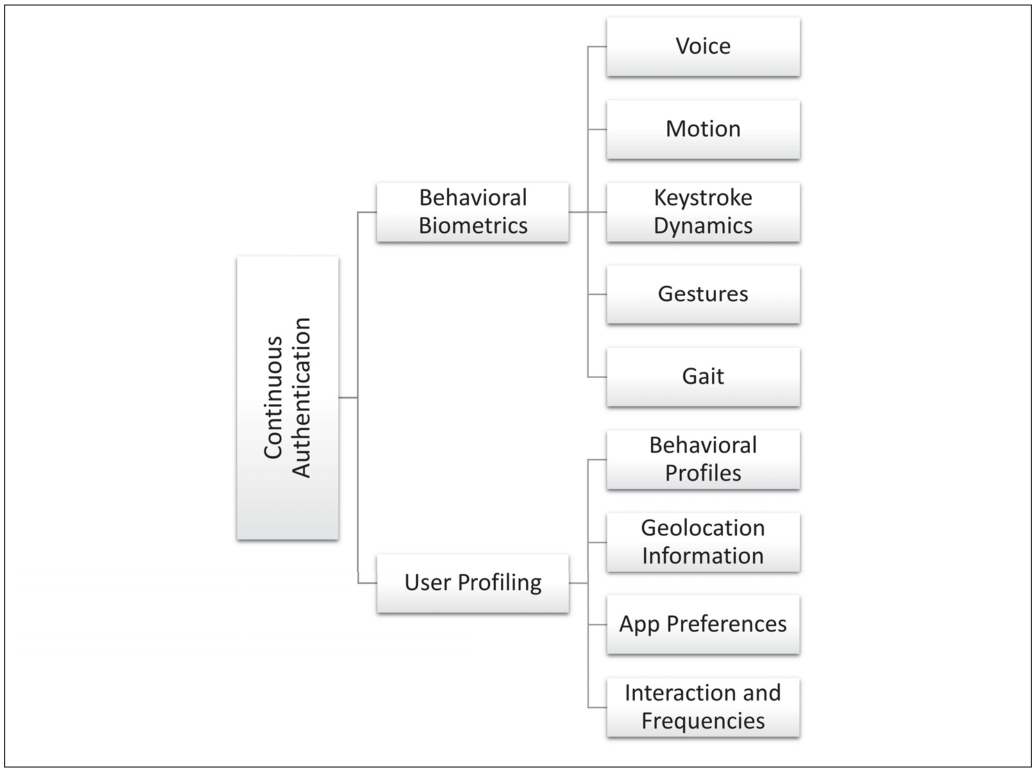

Figure 4.1 Integrated device sensors and user preferences for continuous authentication based on behavioural biometrics

Continuous authentication is a security mechanism that continuously observes a user's behaviour as they interact with a device to verify their identity implicitly in the background. Behavioural sensor data and user profiles are analyzed to create a unique profile that links user-device behaviour and validates users. The requirement for continuous authentication arises to sustain the integrity and dependability of crowdsourced data. This need involves establishing ongoing verification mechanisms for user identities as their behaviours and circumstances evolve (Yang *et al.*, 2015; Feng *et al.*, 2017). Behavioural patterns like typing speed, touchscreen interactions, and device usage can be dependable indicators for continuous user identity verification. Fig. 4.1 shows how applications collect behavioural biometrics and user profiling from sensors integrated into devices, including Cameras, Microphones, Accelerometers, Gyroscopes, GPS, and application statistics. Conventional behavioural authentication methods are limited in mobile crowdsourcing, especially regarding privacy preservation. Centralized approaches that rely on the centralized storage of user data pose significant concerns regarding data security and the risk of privacy breaches (Samet *et al.*, 2019; Lee & Lee, 2017; Zhu *et al.*, 2019). We can adopt an alternative approach based on Federated Learning (FL) to enhance users' privacy further. Federated learning, a decentralized machine learning paradigm, enables collaborative model training on local devices while preserving user data privacy (McMahan *et al.*, 2017a). FL offers a promising solution within continuous authentication by facilitating training in an authentication model based on behavioural patterns without compromising sensitive user information. User data remains decentralized, with local models performing computations and exchanging only model updates instead of raw data, ensuring the preservation of user privacy. However, implementing FL for continuous authentication poses unique challenges. The non-Independently and Identically Distributed (non-IID) nature of collected data, variations in duration, device sensors operating frequencies, activities performed, and unique user labels can significantly impact model accuracy (Wahab *et al.*, 2021; AbdulRahman *et al.*, 2020b,a). Figure 4.2 illustrates how continuous user authentication uses behavioural biometrics, contrasting it with the traditional approach, which sends data to a central server for training. In contrast, the federated learning approach keeps the data on the client side and trains the machine learning model locally on the device. In the research conducted by (Zhao *et al.*, 2018), a warmup model was proposed to enhance initial model weights by sharing a small portion of user data with the server and other participants. This method resulted in a significant 30% increase in accuracy for the CIFAR-10 dataset, but it also raised privacy concerns due to data exchange between devices. Recent studies have proposed innovative solutions to address these concerns to optimize model performance, reduce data sharing, and enhance privacy in federated learning systems. These advancements emphasize improving communication efficiency and integrating secure techniques to protect user data.

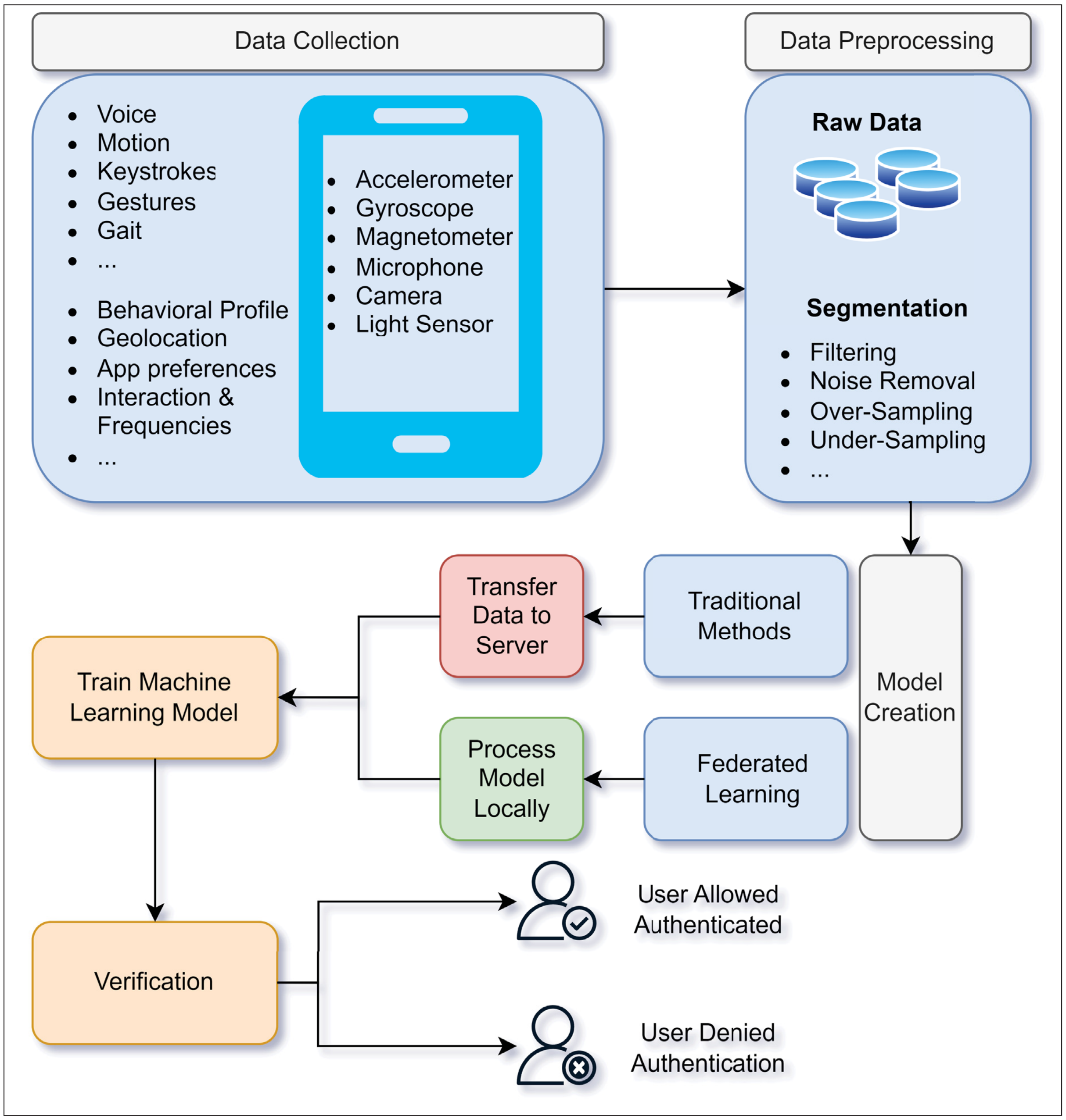

Figure 4.2 Overview of the continuous user authentication process showcasing traditional and federated learning techniques

This paper presents an approach that combines continuous behavioural authentication and FL to tackle user authentication challenges in MCS scenarios. Specifically, we introduce transfer learning and warmup techniques to enhance authentication accuracy while safeguarding user privacy. Transfer learning enables the transfer of pre-trained knowledge from a source domain to improve initial authentication performance (Weiss *et al.*, 2016). Additionally, the warmup approach refines the model by training it on a sample dataset before fine-tuning it with user-generated data in a Federated Learning setting. Our contributions can be summarized as follows:

- We propose a novel approach for enabling continuous authentication in MCS that leverages federated learning. Our framework enables sharing knowledge among multiple parties while maintaining the confidentiality of sensitive user data.
- The integration of transfer learning based on feature extraction and a warmup machine learning approach has demonstrated significant potential in boosting the precision and efficacy of continuous authentication models.
- Extensive experiments conducted on various datasets validate the effectiveness of our proposed approach in addressing the challenges of non-IID and weight divergence in continuous authentication. The experimental results show that our technique outperforms existing state-of-the-art methods, highlighting its potential for practical applications in the field.

4.3 Related Work

This section explores the existing literature on continuous authentication in Mobile Crowdsourcing and its applications in federated learning and transfer learning contexts.

Continuous Authentication. Traditional authentication methods, including password-based authentication, biometric authentication, and multi-factor authentication, are commonly employed for mobile user authentication (Yang *et al.*, 2015; Feng *et al.*, 2017; Liu, Wang & Peng, 2019). Researchers have provided an overview of these methods and evaluated their strengths and limitations in achieving user authentication. Literature has used continuous authentication to preserve session security on mobile and IoT devices. The works of (Sánchez *et al.*, 2021; Lee & Lee, 2017; Zhu *et al.*, 2019; Gascon *et al.*, 2014; Abuhamad *et al.*, 2020) are a few

Table 4.1 Comparison of Continuous Authentication Approaches in the Literature. Ac: Accelerometer. Gy: Gyroscope. Gr: Gravity. Ma: Magnetometer. Ts: Touch Screen. Lo: Location. MP: Multilayer Perceptron. IF: Isolation Forest. SVM: Support Vector Machine. KRR: Kernel Ridge Regression. LSTM: Long Short-Term Memory. FAR: False Acceptance Rate. TAR: True Acceptance Rate. FRR: False Rejection Rate. EER: Equal Error Rate.

Publication	**Features Level**	**ML**	**Participants**	**Accuracy**	**Authentication time**
(Samet *et al.*, 2019)	Ts	MP	34	100%	NA
(Valero, Sánchez, Celdran & Pérez, 2020)	App Usage, Lo	IF	NA	89%	Real-time
(Zhu *et al.*, 2019)	Ac, Gy, Gr	SVM	1,513	Accuracy: 95.6%, TAR: 73.28%	3.2s
(Sun *et al.*, 2020)	Ac, Ts	MVB	26	Accuracy: 94.24%, EER: 8.42%	0.001s
(Jorquera Valero *et al.*, 2018)	Ac, Gy, App Usage	IF	50	82.5%	3-5s
(Lee & Lee, 2017)	Ac, Gy	KRR	35	Accuracy: 98.1%, FAR: 2.8%, FRR: 0.9%	6s
(Gascon, Uellenbeck, Wolf & Rieck, 2014)	Ts	SVM	300	FAR: 1%, TAR: 92%	NA
(Abuhamad, Abuhmed, Mohaisen & Nyang, 2020)	Ac, Ma, Gy	LSTM	84	FAR: 0.95%, FRR: 6.67%, EER: 0.41%	0.5s, 1s
(Sánchez *et al.*, 2021)	Computer (Cp and App Usage), Smartphone (Ac, Gy and App Usage)	MP, XGBoost, RF and LSTM	5	Precision: 99.32%, Recall: 99.33%, F1-Score: 99.33%	2s

examples of traditional centralized approaches that have discussed the viability of training machine learning algorithms on the server side by sending all raw data to an external server for additional processing. Table 4.1 shows the literature review, including the machine learning aspects, the number of participants for data collection, and performance metrics used in each work. The outcomes from these studies have shown impressive levels of authentication and can be used on a user's smartphone without impairing its usability. However, as complete raw data is transferred from user devices to an external server where additional data processing occurs, these solutions lack the foundation for preserving user data security and privacy. The processing, storage, and memory resources of crowdsourcing mobile devices have significantly increased, making it possible to manage device data for processing and evaluation (Rahman *et al.*, 2020). Some works in (Valero *et al.*, 2020; Jorquera Valero *et al.*, 2018) have utilized anomaly detection techniques to train machine learning models on the device itself. The authors in these works used models that require low resource consumption to make the approach feasible on low-end devices. The models are trained using only the positive samples of the device. However, an analysis of this technique shows relatively lower accuracy scores and high false positives since the models lack negative data samples in the training process. Furthermore, the authors in (Sivaram, Rathee, Rastogi, Quasim & Saini, 2020) propose a secure two-stage auction mechanism based on blockchain technology to enhance security and privacy in MCS. The approach establishes a robust authentication process for users and ensures the overall protection of the process by tracking and monitoring entities and their communication behaviour. Utilizing blockchain technology adds an extra layer of trust and transparency to the system.

Federated Learning. Authentication systems require robust privacy-preserving techniques to ensure the security and confidentiality of user data (Elayan *et al.*, 2021). Federated Learning has emerged as a powerful approach to maintaining user privacy in mobile applications, including those involving continuous authentication (Mourad, Tout, Wahab, Otrok & Dbouk, 2020). FL enables training a shared model among multiple nodes, leveraging the global knowledge of individual models while preserving data privacy. This decentralized architecture is particularly relevant in Mobile Crowdsourcing, as it allows for the collaborative training of a global model among participants without raw data sharing. While FL has gained significant attention in various domains, its application in continuous behavioural authentication within MCS still needs to be improved. Recent studies have started exploring FL for continuous user authentication. For example, the study presented in (Zhao *et al.*, 2018) implemented a warmup model that enhanced initial weights by transmitting a small fraction of data to the server and other participants. Although this approach achieved a notable 30% accuracy improvement for the CIFAR10 dataset, it incurred the cost of exchanging data between users' devices, which may raise privacy concerns. To address these challenges, researchers have proposed innovative approaches. For instance, the work by (Monschein *et al.*, 2021) introduced a federated learning approach for peer-to-peer learning in continuous authentication. Instead of relying on FL between clients, they established federated learning between servers, which can help mitigate privacy risks associated with sharing raw data with the servers. Additionally, the study in (Wazzeh *et al.*, 2022a) proposed a federated learning architecture for continuous authentication, addressing the convergence challenges caused by non-IID and weight divergence issues. Additionally, the authors in (Wang *et al.*, 2022) discuss how their approach protects privacy in Mobile Crowdsourcing using an MCS federated learning system incorporating blockchain and edge computing. The proposed system is built on a federated learning framework that employs localized differential privacy techniques to safeguard sensitive data and location privacy. In the study presented in (Yazdinejad *et al.*, 2021), a drone authentication system is proposed that leverages drones' radio frequency capabilities within IoT networks through a federated learning framework. The system ensures data security by employing homomorphic encryption and secure aggregation techniques, which protect the model parameters during data transmission and aggregation processes. The authors in (Feng *et al.*, 2024) present a privacy-preserving aggregation framework designed for federated learning in vehicular ad-hoc networks (VANETs). Their approach enables collaborative training of a global model without exposing local data or gradients from participating vehicles. The scheme employs a non-interactive zero-knowledge proof protocol to authenticate clients and verify the integrity of their model updates, ensuring security.

Transfer Learning. Machine learning algorithms often require significant data to achieve optimal performance. When individuals interact with IoT devices, the sensors capture biometric behaviour data. While each user has a unique interaction style, there are similarities in the types of interactions and device sensors. Transfer learning represents an approach that allows knowledge from one model to be transferred to similar problems. It is beneficial when limited training data is available for an ML model to achieve high performance. Transferring knowledge from a model developed for a specific task can improve performance for other tasks, reducing resource requirements and enhancing the model's robustness (Weiss *et al.*, 2016). In a recent publication, the authors in (Singh *et al.*, 2024) introduced a Personalized Device Authentication Scheme for IoT devices within a zero-trust network. This method utilizes Q-learning-based decision-making and Transfer Fuzzy Learning to address the shortcomings of current device authentication approaches. The focus is on personalized authentication tailored to individual device security requirements. The process involves establishing device profiles, assigning unique hash values, and employing encryption to guarantee data confidentiality and integrity. In (Kong *et al.*, 2020), a continuous authentication technique was utilized through gesture interaction with IoT devices. A transfer learning technique was incorporated for cross-domain adaptation, eliminating the requirement for training the model in various environments. Through their study, the authors demonstrated that this knowledge-based approach maintained a high level of accuracy performance even with a smaller quantity of data samples. Another research proposed in (He *et al.*, 2022) focused on continuous authentication for smartphone users using sensor data related to gait. Transfer learning as a feature extractor played a vital role in extracting relevant features from the data, leading to efficient model training and a noticeable improvement in the model's robustness. When tested in diverse usage environments, the trained model successfully extracted distinctive features and achieved significantly better accuracy scores.

4.4 Proposed Framework

The proposed framework addresses the difficulties of non-IID data and weight divergence in continuous authentication for Mobile Crowdsourcing applications. This section describes the framework's architecture presented in Figure 4.3, highlighting its distinctive characteristics and benefits.

4.4.1 Problem Formulation

Let K be the total number of clients, where each client k has its dataset D_k of size N_k and a unique label y_k. We represent D_k as a sequence of data samples:

$$D_k = d_1^k, d_2^k, \ldots, d_{N_k}^k \tag{4.1}$$

where d_s^k is the s-th data sample of size F in D_k, and F is the set of features in each d_s^k.

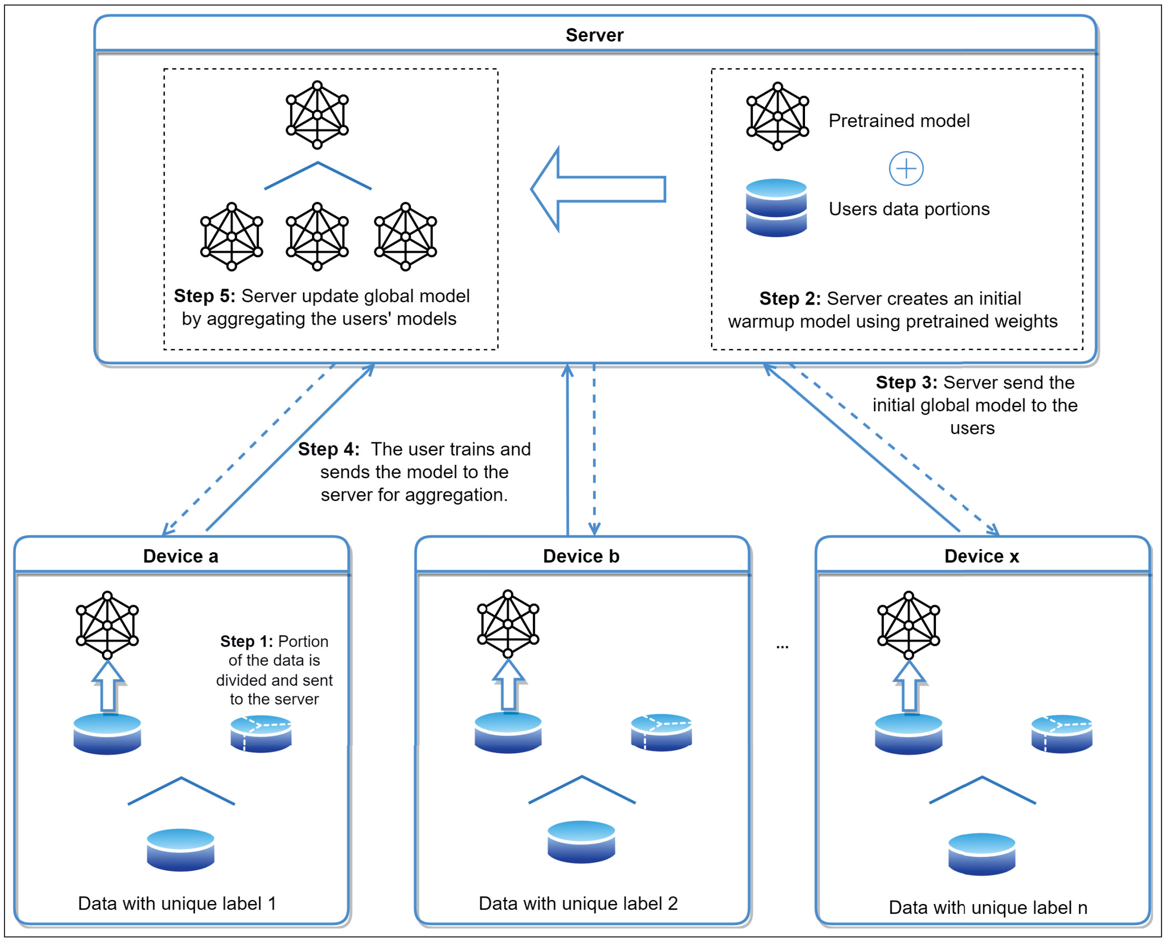
Server
Pretrained model
Users data portions
Step 2: Server creates an initial warmup model using pretrained weights
Step 5: Server update global model by aggregating the users' models
Step 3: Server send the initial global model to the users
Step 4: The user trains and sends the model to the server for aggregation.
Device a
Device b
Device x
...
Step 1: Portion of the data is divided and sent to the server
Data with unique label 1
Data with unique label 2
Data with unique label n

Figure 4.3 Collaborative Knowledge Transfer Architecture for Continuous Authentication

The input to the neural network model is represented as a sequence of all the clients' datasets:

$$\mathbf{D} = D_1, D_2, \ldots, D_K \tag{4.2}$$

The model's accuracy output for a particular client k is denoted as A_k, and the local accuracy prediction for each sample in D_k is represented as a vector:

$$A_k = (a_1^k, a_2^k, \ldots, a_{N_k}^k) \tag{4.3}$$

where a_s^k is the local accuracy prediction for the s-th data sample d_s^k.

Let Y be the true label of the clients, represented as a vector:

$$Y = (y_1, y_2, \ldots, y_K) \tag{4.4}$$

Our approach involves a current domain Q_S and a learning task T_S for clients to perform. Additionally, we have a target domain of a different dataset represented by Q_T and the learning task T_T. We use transfer learning to leverage the knowledge in Q_S and T_S to improve the prediction target function $f_T(\cdot)$ in Q_T. In our approach, Q_S is not equal to Q_T, and T_S is not equal to T_T.

4.4.2 Non-IID and Unique Label

The classic federated averaging algorithm is based on the assumption that the data are independent and identically distributed (IID). However, the accuracy of this algorithm for continuous authentication is often low due to the non-IID characteristics of behavioural biometric data. This occurs because users typically possess varying numbers of data instances and unique labels that are not shared with others. Recent studies, such as (Zhao *et al.*, 2018; Oza & Patel, 2021a; Li, Diao, Chen & He, 2021b), have explored the use of non-IID data with FedAvg, where clients only have access to positive data on their devices and not to other clients' labels. However, the non-IID data distribution in continuous authentication presents a significant challenge, as the FedAvg algorithm needs to converge optimally due to unique client labels. Moreover, previous studies have revealed that data skewness among users causes divergence in model weights, leading to bias toward specific clients and negatively affecting global model performance. These studies also found that models exhibit the highest performance degradation when the label distribution is highly skewed. Additionally, experiments in (Li *et al.*, 2021b) indicated that the ML accuracy in federated learning does not increase when each user has only one label for the collected data.

4.4.3 Framework Details

To address the issues of weight divergence in applying federated learning for continuous authentication, we propose a novel approach that utilizes transfer learning techniques to train a warmup model. Using the feature extractor, this initial model will leverage pre-trained weights and adapt to the similarities in the clients' data. The weights generated by the warmup model will be employed to minimize the impact of weight divergence in the models. In the following steps, we summarize our proposed framework illustrated in Figure 4.3 as follows:

1. The user interacts with their device to collect behavioural data segmented into smaller subsets. A fraction of these data samples are randomly selected and transmitted to the server for initial training. This step is only carried out in the first round of each user's participation, ensuring minimal data transfer and reducing initial overhead.

2. The server aggregates these portions of the users' data to train an initial machine-learning model centrally. In this process, the server leverages pre-trained model weights as feature extractors to enhance the learning efficiency. A predetermined number of layers in the model are frozen to function as these extractors, while the rest of the layers are trained to fine-tune the model to the incoming user-specific data. This division ensures that the model capitalizes on prior learning while still adapting to the new data provided by the users.

3. Once the initial model is created, the server sends the global model weights to the clients. These initial weights, informed by the aggregated data, serve as the starting point for each client's local model training.

4. Each federated participant utilizes the received global model weights to train their model locally using their private data. The client only updates the trainable layers, excluding the frozen feature extractor layers. Upon completing local training, the updated model weights are sent back to the server for aggregation.

5. The server collects and aggregates the updated local model weights from all participating clients to create an updated global model. This updated global model, which now reflects the combined knowledge of all clients, is then sent back to the clients, completing one round of federated learning. The process repeats for subsequent rounds, refining the model with each iteration.

The proposed approach effectively mitigates the issue of weight divergence in applying federated learning with the FedAvg algorithm. It directly addresses the challenges posed by non-IID data distributions, enabling continuous authentication for individual users' behavioural data, which have unique labels for each group of samples. By starting with robust initial weights derived from transfer learning and warmup strategies, this approach prevents model divergence and significantly improves the overall performance of the federated model.

Algorithm 4.1 Collaborative Knowledge Transfer Algorithm for Federated Learning

```
1:  Notations:
2:    $C$: fraction of users selected per round,  $K$: total users
3:    $n_i$: samples for user $i$,  $n$: total samples
4:    $B$: batch size,  $E$: local epochs,  $\eta$: learning rate
5:    $\ell$: loss function,  $D$: global training dataset
6:
7:  Input: $\zeta, C, K, n_i, n, B, E, \eta, \ell, D$
8:  Output: $w_{r+1}$    ▹ Updated global model
9:  Server side:
10:   Create or load initial model weights $w_0$
11:   $w_0 \leftarrow$ Train$(w_0, \zeta)$    ▹ Warmup with $\zeta$ data
12:   for $r = 1, 2, \ldots$ do
13:     $p \leftarrow \max(C \cdot K, 1)$
14:     $S_k \leftarrow$ (randomly select $k$ users)
15:     for each $k \in S_k$ do
16:       $w^k_{r+1} \leftarrow$ TrainUser$(k, w_r)$
17:     $w_{r+1} \leftarrow \sum_{i=1}^{k} \frac{n_i}{n} w^i_{r+1}$    ▹ Federated averaging
18: Function Train$(w, D)$:
19:   $\beta \leftarrow$ (split $D$ into batches of size $B$)
20:   for epoch $i = 1$ to $E$ do
21:     for each mini-batch $b \in \beta$ do
22:       $w \leftarrow w - \eta \nabla \ell(w; b)$    ▹ Gradient step
23:   return $w$
24: User side:
25:     TrainUser$(k, w)$ :
26:         return Train$(w, D_k)$
```

The detailed process of our algorithm is summarized as pseudo-code presented in Algorithm 4.1.

The collected data samples from users ζ are used to train initial model weights w_0 or load pre-trained model weights, depending on the problem's complexity and the available data samples. The transfer learning step is performed by freezing several first layers of the model and keeping the remainder for the model to be trained. After the training phase and at the end of R, the total number of rounds or until model convergence, the server randomly selects a set of available participants of p. In **TrainUser**, machine learning training is requested (k, w_r) from each user k. The user, in turn, trains a new model using the data collected by their device D_k in federated learning round w_{r+1}^k. The users' data D are shuffled and split into training and testing sets with a mini-batch size of B. The model weights w are updated based on a specified learning rate η, and the loss function used is represented by cross-entropy ℓ. To update the global model weights, we averaged the model weights following the step of the FedAvg algorithm as follows:

$$w_{r+1} = \sum_{k=1}^{k} \frac{n_k}{n} w_{r+1}^k \tag{4.5}$$

In Equation 4.5, the weights aggregation step computes a superior global model by averaging the weights of the models trained by each user k, weighted by the number of samples n_k used by each user, relative to the total number of samples n across all users. This aggregated model can differentiate and identify the data labels each participating user uses. The resulting global model is then transmitted to the selected users for the following federated learning round until convergence.

The cross-entropy loss function $\ell(w; b)$ can be represented as follows:

$$\ell(w; b) = \sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.6}$$

Here, M represents the total number of classes or labels, $y_{o,c}$ represents the correct classification of an observation o for a class c, and p represents the predicted probability.

We performed extensive experiments on various datasets, including a facial dataset such as the University of Maryland Active Authentication-02 Face Dataset (UMDAA-02-FD) (Mahbub, Sarkar, Patel & Chellappa, 2016a). Each user's image samples are facial images containing a human face. Therefore, using pre-existing model weights previously trained to identify human faces can significantly boost accuracy performance. Transfer learning is a helpful approach to transferring the knowledge gained from learning to one problem and applying this acquired knowledge to solve another problem. Therefore, we can leverage a previously trained model to boost training performance. This work uses transfer learning as a feature extractor since we are dealing with facial images from the UMDAA-02-FD dataset. The images represented in these datasets are limited. Thus, we can benefit from pre-existing publicly available datasets, such as the VGGface2 (Cao, Shen, Xie, Parkhi & Zisserman, 2018) dataset, which contains millions of images with over 9000+ classes. Therefore, applying transfer learning of the pre-trained model, such as the InceptionResnetV1 (Szegedy, Ioffe, Vanhoucke & Alemi, 2017) model, over this dataset is a suitable selection since we are dealing with human facial pictures. We use the Inception model by training the last five layers and freezing the previous ones in our approach. We tackled the challenge of model adaptation created by the transfer learning frozen layers by following an iterative process of selecting and fine-tuning models. We conducted experiments with various pre-trained models and layer configurations. We obtained the optimal results by fine-tuning the last five layers and modifying the number of output classes to match the specific requirements of our dataset. Therefore, we can customize the model to classify the new classes in the UMDAA-02-FD dataset. The knowledge gained from the pre-trained model weights helps us quickly identify the faces in the images, which improves the model's accuracy instead of training a model from scratch.

The following section presents our datasets in more detail alongside the experiments and configurations.

4.5 Experimental Settings

4.5.1 Environment Setup

To deal with the high-dimensional image data, we utilized a graphics processing unit (GPU) to complete our experiments efficiently. Specifically, we used a high-resource Windows platform with 16 CPUs running at 2.90 GHz and 16 GB of memory. The CPU processed the inferences, while the GPU GTX 1660 Super handled the models' training phase. Our code was developed based on the FedML GitHub repository (He *et al.*, 2020), and we used the Weights & Biases (WandB) (Biewald, 2020) package to visualize our experimental results.

4.5.2 Datasets

To test the viability of our method, we conducted thorough experiments using different publicly available datasets with varying characteristics. The datasets we used to assess our approach are MNIST (Deng, 2012), CIFAR-10 (Krizhevsky *et al.*, 2014), FEMNIST (Caldas *et al.*, 2018), and UMDAA-02-FD. We standardized all datasets to fit the continuous authentication format. This involved creating customized samples of the data and assigning each user a unique set of data belonging to only one label, mirroring the unique non-IID data use-case often seen in behavioural datasets. We selected the MNIST, FEMNIST, and CIFAR-10 datasets as benchmark datasets for future comparisons. On the other hand, the UMDAA-02-FD dataset was designed for facial identification and is tailored to our problem of continuous user authentication. The data was preprocessed as follows:

1. For all the data available in a dataset, we group all the data and sort them based on the label of the data row.
2. For every label we have in a dataset, we assign the data that belongs to this label to a client, to which each client has access to only one unique label of the data. Each dataset's total number of classes is represented in Table 4.2 alongside the dataset's data feature and total sample size.

Table 4.2 Details of Datasets Used in the Experiment

Dataset	Train Examples	Test Examples	Features	Classes
UMDAA-02-FD	26,566	6,641	16,384	44
CIFAR-10	50,000	10,000	1,024	10
FEMNIST	671,585	77,483	784	62
MNIST	60,000	10,000	784	10

3. The total number of samples differs for every client we have in a dataset. To prepare the client data, we have implemented a random client distributor to select the clients' numbers and the minimum and maximum number of samples we need for each client.

UMDAA-02-FD. The UMDAA-02-FD dataset (Mahbub *et al.*, 2016a) consists of facial images specifically captured for authentication purposes using the frontal camera of smartphones. This dataset includes 44 users with 33,209 images taken at intervals of 7 seconds, capturing a wide range of illumination, poses, occlusions, and facial expressions, as well as instances with no face presented or total black images. UMDAA-02-FD is well-suited for continuous authentication in mobile crowdsourcing environments due to its emphasis on facial imagery captured in real-world situations. The implicit collection of images during user interactions aligns with the dynamic nature of crowdsourcing, enabling seamless and background user authentication to ensure uninterrupted user experiences.

MNIST. The MNIST dataset (Deng, 2012) captures the recognition of handwritten characters, encompassing over 70,000 samples with 784 features for each data point. This dataset is a valuable representation of data collection in mobile crowdsourcing environments, where users frequently contribute inputs through writing or drawing. By emulating the recognition of human writing characters, MNIST enables us to evaluate the model's ability to comprehend user-generated data, which is essential for continuous authentication systems. The familiarity of this dataset facilitates a baseline performance assessment that can later be compared with more complex datasets.

CIFAR-10. The CIFAR-10 (Krizhevsky *et al.*, 2014) dataset is a well-known benchmark for training computer vision algorithms, consisting of 60,000 images categorized into ten classes. The dataset is valuable as it contains a wide range of images representing various object categories. This allows us to assess the robustness of our model across different contexts. Although CIFAR-10 does not specifically capture user behaviours typical of mobile crowdsourcing, its diverse image classes enable a more generalized understanding of how the model can adapt to different visual inputs. This adaptability is crucial for real-world applications, where user interactions vary significantly. Our data distributor has created ten client partitions, each containing 6,000 images.

FEMNIST. The FEMNIST dataset is an extension of the MNIST (Caldas *et al.*, 2018) dataset, encompassing over 700,000 data samples. Each sample consists of a 28x28 pixel image representing digits from zero to nine and English alphabetic letters in lowercase and uppercase. This dataset significantly broadens the scope of character recognition tasks by encompassing a diverse array of handwritten styles and variations, making it particularly pertinent for applications in continuous authentication. With numeric and alphabetic characters included, it enables the simulation of user inputs commonly encountered in real-world mobile crowdsourcing scenarios, where users frequently submit written responses or undertake data entry tasks. In our study, we designed a custom partition to separate and randomly distribute data samples for each class, providing each client with a unique label and a sample range of 2000 to 3000. This method allows us to mimic user behaviour in a controlled environment and demonstrates the FEMNIST dataset's versatility in representing user interactions across different scenarios. While it offers valuable insights into character recognition, we recognize the potential for more comprehensive datasets that can capture the dynamic nature of mobile interactions. Nevertheless, FEMNIST provides a solid basis for evaluating the effectiveness of our continuous authentication approach, serving as a dependable benchmark for character recognition tasks in mobile crowdsourcing settings.

4.5.3 Implementation Details

4.5.3.1 Experiments

To test our hypotheses, we conducted extensive experiments on various datasets. Our performance evaluation involved three different experiments.

We initially trained a conventional machine learning model using the baseline method, in which all the raw sensor data collected from the clients is transmitted to a centralized server for model training. This approach, which assumes full access to client data, is called the *Centralized* method. The centralized approach serves as a benchmark to evaluate the performance of other methods under ideal circumstances where the server has access to complete datasets from all users.

Next, we applied the Federated Averaging (FedAvg) algorithm to assess the performance of federated learning (FL) as a decentralized alternative. In this setup, the clients perform local training on their own devices, and only the model updates (i.e., weights or gradients) are sent to the server for aggregation, preserving data privacy. Our figures 4.4, 4.5, and 4.6 show the results of this experiment, which is labelled as *FedAvg*. The Comparison between the Centralized method and FedAvg allows us to highlight the trade-offs in performance, communication overhead, and privacy preservation.

To further improve the FedAvg method, we experimented with our approach based on a warmup and transfer learning-based federated learning algorithm, named *Warmup*. In this method, transfer learning is leveraged by keeping most pre-trained model architecture intact while retraining only the final five layers, focusing on the specific task. This transfer learning technique allows us to fine-tune the model with task-specific data while retaining general knowledge from the pre-trained weights. The warmup phase begins with a small subset of the dataset that is first processed on the server to initialize the model with pre-trained weights.

During the warmup phase, we adopted a percentage-based approach to client data collection in all experiments. Specifically, each client contributes a fraction of their local data, which is then used to update the global model. To maintain confidentiality and protect sensitive warmup data during transmission, we implemented a selective data transmission process, where only a small random subset of 5% of each client's data samples is shared with the server. This selective data transmission reduces communication overhead and enhances data privacy by limiting the exposure of raw data.

Our experimental results demonstrate that using pre-trained model weights during the warmup phase, rather than initializing the model with random weights, as is common in classical FL experiments, significantly improves the overall model performance. Pre-trained weights provide a more robust starting point for federated learning, as they encapsulate generalizable knowledge that accelerates convergence and improves accuracy. After clients complete local training, the server aggregates their updated model weights using the FedAvg algorithm to produce a global model. This global model is then distributed back to the clients and serves as a classifier across all user devices, continually refined through subsequent rounds of FL. Our current methodology involves sample-based average aggregation, proven effective in federated learning. However, alternative approaches such as alpha integration, which optimally combines models based on least mean-square error (LMSE) and minimum probability of error (MPE) criteria, show promise for improvement. Our future research will investigate the implementation of this approach, potentially enhancing model robustness and performance, particularly in environments with diverse data distributions (Salazar, Safont, Vergara & Vidal, 2023a). To address the potential risk of overfitting caused by the small sample size in our datasets, we utilized cross-validation throughout the model training process. This method provided us with a more dependable assessment of model performance and ensured its generalizability. Furthermore, we conducted thorough experiments to fine-tune the learning rate and the number of epochs, thereby improving the model's capacity to learn without overfitting. To determine the optimal learning rate for each dataset and experiment, we conducted a series of experiments utilizing neural network models, including the convolutional neural network (CNN) and the InceptionResnetV1 pre-trained model, trained initially on the VGGFace2 dataset. These models are well-suited for image classification tasks due to their capacity to capture intricate patterns in image data and have consistently demonstrated high accuracy and model convergence in prior works (McMahan *et al.*, 2017a; Szegedy *et al.*, 2017; He *et al.*, 2020). In particular, combining CNN and deep pre-trained models such as InceptionResnetV1 has effectively reduced the overall training time while maintaining robust classification performance across various datasets.

Our approach involves utilizing both traditional models and transfer learning techniques, focusing on feature extraction, to improve model accuracy even when working with a limited number of data samples. We can adapt pre-trained models for specific tasks by employing transfer learning, reducing the computational and data resource requirements typically associated with training models from scratch. This strategy is particularly advantageous in FL environments, where the number of samples available from each client varies significantly.

The training sample size plays a crucial role in determining model performance. Previous studies, such as (Salazar, Vergara & Vidal, 2023b), have shown that the relationship between sample size and classification accuracy can be analyzed using learning curves. These curves estimate the decrease in classification error as the training set size increases. Through transfer learning, we aim to mitigate the potential impact of limited data by leveraging pre-trained weights, which enable faster convergence and reduce the number of communication rounds required in the FL process. This results in a more efficient learning process, especially when working with smaller datasets while balancing sample size and classification performance.

Specifically, we used the InceptionResnetV1 model pre-trained on human faces from the VGGFace2 dataset (Cao *et al.*, 2018) as a foundation for our image classification tasks. We fine-tuned the model by retraining the last five layers and adjusting the number of output classes to match the number of categories in our target datasets. This selective retraining allowed the model to specialize in our specific classification tasks while retaining the general features learned from the VGGFace2 dataset. Notably, this approach proved particularly beneficial for the UMDAA-02-FD dataset, where the InceptionResnetV1 model exhibited strong performance in detecting and classifying human faces. The ability of the pre-trained model to generalize from its original training data and quickly adapt to new tasks demonstrates the efficacy of combining transfer learning with FL, allowing for more efficient training and improved overall accuracy.

4.5.3.2 Models

We conducted experiments on the UMDAA-02-FD datasets using different machine-learning models. One of these models was a ResNet, which was trained from scratch. Another model, InceptionResNetV1, was pre-trained on the VGGFace2 dataset. For feature extraction, we utilized the pre-trained InceptionResNetV1 model and fine-tuned the network's last five layers to function as a classifier. For the MNIST and FEMNIST datasets, we trained a CNN model similar to the one used in (McMahan *et al.*, 2017a). CNNs are popular for visual training systems as they can handle high-dimensional input data. A CNN model with six conv2D layers was used for federated and centralized experiments on the CIFAR-10 dataset. We observed that using a CNN with fewer layers resulted in inferior results.

4.5.3.3 Evaluation

We used the accuracy metric to evaluate the performance of the models over the predicted labels. The accuracy is the ratio of correct predictions to the total number of predictions. Specifically, we use the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.7}$$

In this scenario, TP represents the number of times the model correctly predicts a positive label. TN represents the number of times the model correctly predicts a negative label. FP refers to the cases where the model predicts a positive label for a sample with a negative label. Lastly, FN relates to the cases where the model predicts a negative label for a sample that has a positive label.

In addition, we compute the average accuracy avg_{acc} of the clients K models as follows:

$$avg_{acc} = \frac{1}{K}\sum_{k=1}^{k} acc_k = \frac{1}{K}(acc_1 + \cdots + acc_k) \tag{4.8}$$

Here, acc_k represents the accuracy of the k-th client model, and K is the total number of clients in the FL setting. A notation set for the experiments and graphs is selected. For instance, we use C to represent the total number of participants specified in an FL round, B is the batch size of client data, and E is the total number of local epochs used to train the machine learning model.

4.5.3.4 Results

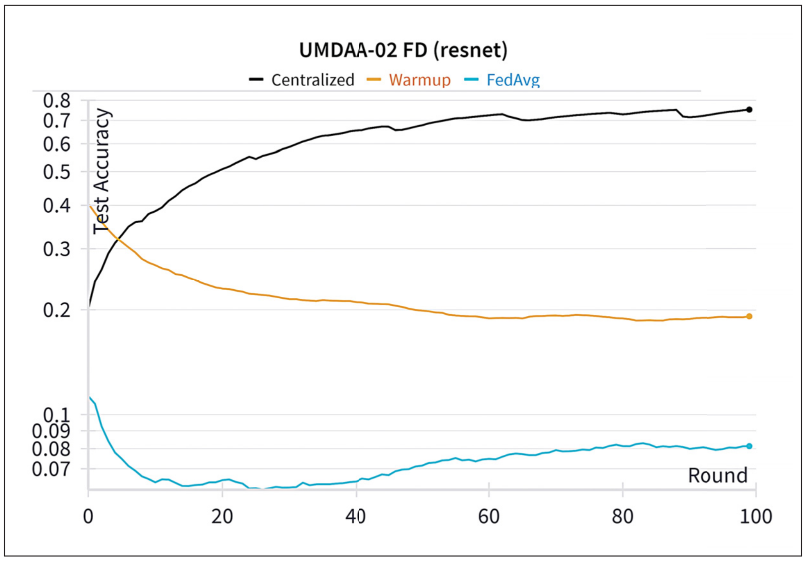

Figure 4.4 Test set accuracy vs. communication rounds for the UMDAA-02-FD dataset using the Resnet56 model

The results of using a ResNet model with the UMDAA-02-FD dataset are displayed in Fig. 4.4. The plot indicates that the federated averaging algorithm did not produce high accuracy results, and even the warmup approach of the ResNet model only improved accuracy by 10%

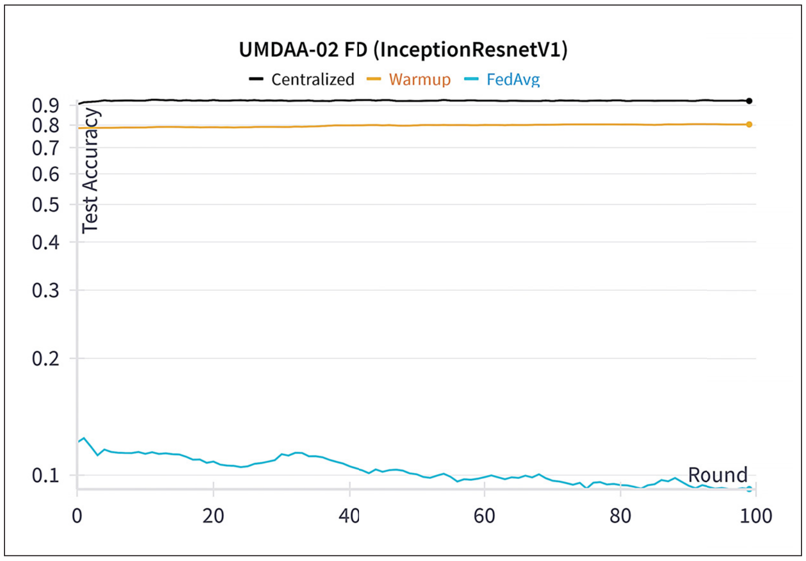

Figure 4.5 Test set accuracy vs. communication rounds for the UMDAA-02-FD dataset using the InceptionResnetV1 pre-trained model

after 100 rounds. The centralized experiment also had an accuracy below 80% after 100 rounds, suggesting limitations in applying the ResNet architecture to this particular dataset in a federated setting. While ResNet has demonstrated exceptional performance in handling complex data inputs, as proposed by (He, Zhang, Ren & Sun, 2016), its performance in decentralized contexts may be hindered by the high variance in data distribution across clients, resulting in suboptimal convergence. We used a learning rate of 0.001 and 0.01 for the federated and centralized experiments, respectively, with a batch size of 24 and a single epoch.

To mitigate these limitations, we took a different approach and utilized transfer learning with the InceptionResnetV1 model, pre-trained on the VGGFace2 dataset. This allowed us to leverage pre-trained weights for identifying faces in the UMDAA-02-FD dataset, significantly reducing the need for extensive training data. We improved accuracy by using a low learning rate of 0.001, a batch size of 24, and a single epoch for federated and centralized experiments, as shown in Fig. 4.5. However, even with the pre-trained model, the FedAvg algorithm could not effectively classify the data samples, with accuracy dropping below 10% . Despite this, the warmup experiment closely matched the centralized experiment's performance, with a final accuracy of

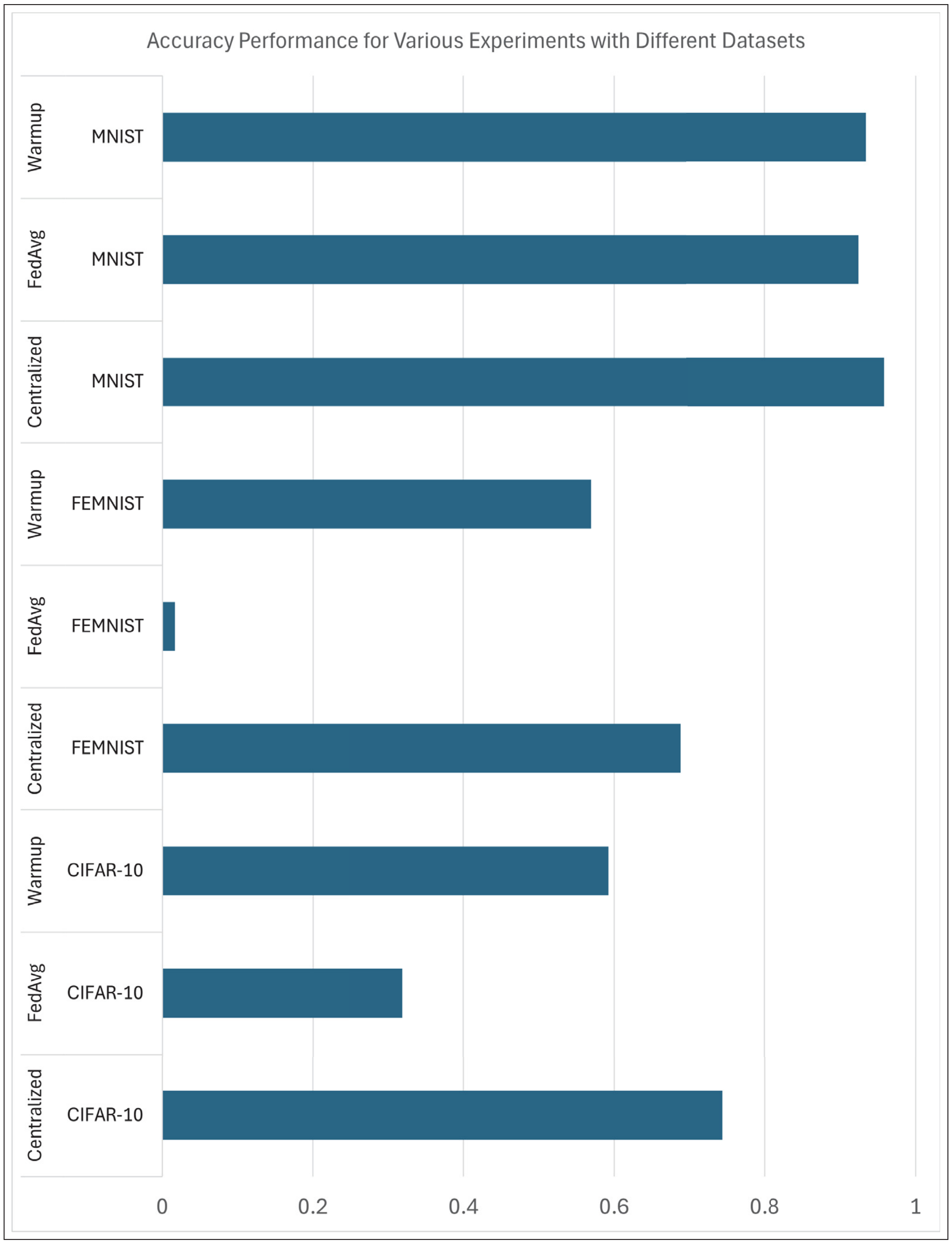

Figure 4.6 Comparison of accuracy test results for different datasets

80%. This highlights a limitation of the FedAvg algorithm in handling high-dimensional data like images in decentralized environments, where warmup approaches that employ pre-trained models offer significant performance benefits. Our approach resulted in a 70% improvement in performance, which is 60% higher than that of the ResNet model.

In the MNIST experiment, we found that a small data distribution per user was sufficient for accurately identifying users, as seen in Figure 4.6. We used a warmup approach where 5% of each client's data was utilized to initialize the model. We ran 800 communication rounds to achieve convergence, randomly selecting 20% of clients per round. The FedAvg algorithm achieved 97% accuracy for single-labelled data across ten clients, demonstrating that simpler datasets with lower complexity are more suitable for federated approaches. However, a limitation observed in these experiments is the slower convergence in federated settings compared to centralized ones. Although the warmup model slightly improved convergence, the final accuracy was still 96.22%, compared to 98.53% for the centralized model. This indicates a persistent gap between federated and centralized results, particularly in complex tasks.

In the CIFAR-10 experiments, where the dataset complexity is higher, we experimented with different learning rates (lr), finding that lr=0.1 worked best for federated learning, while lr=0.001 was optimal for centralized experiments. As presented in Figure 4.6, the centralized approach achieved remarkable results with fewer rounds. In contrast, the warmup approach performed 30% better than the FedAvg algorithm in terms of accuracy. This suggests that in high-dimensional tasks such as CIFAR-10, warmup strategies based on pre-trained models offer a significant advantage. However, it is important to note that the warmup model, although performing better than FedAvg, still lagged behind centralized methods regarding convergence speed and overall accuracy. In Figure 4.6, we observed that the warmup model provided an initial 60% accuracy boost compared to the classical FedAvg experiment, where accuracy remained below 2%, indicating that classical federated approaches struggle in handling high-dimensional, multi-class datasets. This highlights a critical limitation in the scalability and effectiveness of standard federated learning algorithms in complex real-world tasks.

These experiments collectively highlight the potential and the limitations of applying federated learning in different scenarios. While the warmup approach demonstrates clear benefits over

FedAvg, particularly when leveraging pre-trained models, it is evident that the federated learning framework still struggles to match the performance of centralized approaches, especially in tasks requiring high-dimensional data classification. This gap calls for further exploration of advanced techniques, such as personalized federated learning or hybrid architectures, to bridge the performance disparity while maintaining the privacy-preserving benefits of decentralized learning.

4.5.4 Discussion

While traditional machine learning methods focused on centralized models, the inherent privacy and security risks associated with transferring all client data to a central server became a significant concern, particularly with sensitive behavioural data. Our proposed federated learning (FL) approach offers an advantage by keeping user data local. However, federated learning also introduces challenges, particularly data heterogeneity and communication overhead. Specifically, the non-IID (non-independent and identically distributed) nature of each client's behavioural data in our experiments posed a significant issue for the FedAvg algorithm, which assumes balanced and uniform data distribution across clients. The unique, non-IID behavioural data, where each client's data represents different labels or behaviours, directly affects the model's ability to converge effectively in a decentralized setting. This limitation resulted in slower convergence and lower initial accuracy in FedAvg, highlighting the difficulty of applying traditional federated algorithms to highly personalized datasets. Furthermore, data imbalance across clients, where certain users provided significantly more data than others, increased the challenge by causing weight divergence during aggregation. This led to unstable updates and inconsistent model performance, especially in the initial training rounds. We introduced a novel strategy to mitigate these issues, incorporating an initial backbone model to reduce weight divergence and improve accuracy. By leveraging pre-trained model weights and transfer learning techniques, we stabilized the training process and ensured that the data's personalized nature did not hinder model convergence. While this approach yielded improved accuracy across all datasets, it also highlighted a key limitation: the success of transfer learning relies heavily on selecting an appropriate pre-trained model, which may not always be available for all types of data. In scenarios where a suitable backbone model is unavailable, the observed performance gains may decrease, potentially limiting the effectiveness of this approach. Additionally, our experiments on the three datasets, CIFAR-10, FEMNIST, and UMDAA-02-FD—demonstrated that our strategy resulted in significant accuracy improvements of 40%, 60%, and 70%, respectively, compared to traditional methods. However, despite these significant improvements, our approach is not without its limitations. The results showed that in highly complex datasets like CIFAR-10, where the images contain high-dimensional features, the warmup and transfer learning strategy led to slower convergence than the centralized model. This reflects a common limitation in FL frameworks: while privacy is preserved, achieving the same efficiency level as centralized learning remains a challenge, particularly for more complex, multi-class datasets that require larger models and longer training times. Moreover, relying on communication rounds between the server and clients in the FL framework poses an additional limitation. In our experiments, particularly with the UMDAA-02-FD dataset, achieving the desired accuracy required an increased number of communication rounds (up to 1000). This is a limitation in practical deployment scenarios concerning communication costs and latency. In real-world FL applications, minimizing communication overhead remains a challenge, and our method, while improving accuracy, did not significantly reduce the number of communication rounds needed for convergence. While authentication is essential for security, continuous authentication can affect user experience if not properly managed. Potential usability issues include frequent interruptions and high resource consumption. It is recommended that adaptive authentication intervals based on user behaviour and resource-efficient data collection methods be implemented to mitigate these concerns. These strategies can minimize disruptions while ensuring security. Our novel approach using a backbone model and transfer learning substantially improved model accuracy while maintaining data privacy. However, limitations in communication efficiency, data heterogeneity, and the availability of suitable pre-trained models continue to be challenges in achieving optimal performance in federated learning environments.

4.6 Conclusion and future work

Continuous authentication is vital in ensuring the security and privacy of sensitive data in MCS systems. It offers an essential layer of defence that makes it significantly more difficult for adversaries to compromise user authentication mechanisms. In this work, we proposed a novel privacy-preserving federated learning approach designed to address the unique challenges posed by non-IID behavioural data. Our approach effectively balances the need for model accuracy with stringent privacy and security requirements by leveraging minimal data sharing and employing a warmup transfer-knowledge strategy.

We specifically tackled the issue of weight divergence caused by non-IID data, particularly behavioural data that is highly individualized. By implementing a pre-trained model as a feature extractor, we improved model convergence, leading to substantial gains in accuracy. Transfer learning significantly enhanced the model's performance when working with image-based datasets, such as facial recognition tasks. Experimental results demonstrated that our approach led to considerable improvements in accuracy—40% for the CIFAR-10 dataset, 60% for FEMNIST, and 70% for the UMDAA-02-FD dataset—without compromising user privacy. However, while our approach yields promising results, there are still challenges to address. One of the primary limitations is the communication overhead associated with federated learning, as achieving convergence requires many communication rounds between the server and clients. This can increase costs and delay real-time deployment. Additionally, the reliance on pre-trained models highlights the need for an adaptable strategy when such models are unavailable.

Future work will focus on several key areas to further enhance the practicality and scalability of our method. We plan to explore the development of personalized models tailored to each client's unique data distribution. This approach will improve accuracy for users without requiring significant changes to the global model. We will also investigate strategies to reduce the number of communication rounds necessary for model convergence. This includes optimizing the frequency of model updates and exploring techniques like federated compression to minimize the data exchanged between clients and servers.

Moreover, we aim to extend the transfer learning framework to cover a broader range of tasks and datasets, particularly in scenarios where pre-trained models are not readily available. This could involve dynamic model selection or adaptation based on the nature of the data. Finally, another critical focus will be refining our approach to work efficiently in resource-constrained environments, such as IoT devices with limited processing power and storage, ensuring our method remains scalable and lightweight.

CHAPTER 5

DYNAMIC SPLIT FEDERATED LEARNING FOR RESOURCE-CONSTRAINED IOT SYSTEMS

Mohamad Wazzeh[1,2] , Ahmad Hammoud[2,3,4] , Azzam Mourad[5,2] , Hadi Otrok[6] , Chamseddine Talhi[1] , Zbigniew Dziong[4] , Chang-Dong Wang[7] , Mohsen Guizani[3]

[1] Department of Software and IT Engineering, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Artificial Intelligence and Cyber Systems Research Center, Department of Computer Science and Mathematics, Lebanese American University, Chouran, Beirut 1102 2801, Lebanon
[3] Department of Machine Learning, Mohamed Bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates
[4] Department of Electrical Engineering, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[5] KU 6G Research Center, Department of Computer Science, Khalifa University, Shakhbout Bin Sultan St, Hadbat Al Za'Faranah, Zone 1, Abu Dhabi, United Arab Emirates
[6] Center of Cyber-Physical Systems (C2PS), Department of Computer Science, Khalifa University, Shakhbout Bin Sultan St, Hadbat Al Za'Faranah, Zone 1, Abu Dhabi, United Arab Emirates
[7] School of Computer Science and Engineering, Sun Yat-sen University, Haizhu District, Guangzhou, Guangdong Province, China

Paper submitted to IEEE Internet of Things Journal, September 2024

5.1 Abstract

Efficient resource utilization in Internet of Things (IoT) systems is challenging due to device limitations. These limitations restrict on-device machine learning (ML) model training, leading to longer processing times and inefficient metadata analysis. Additionally, conventional centralized data collection poses privacy concerns, as raw data has to leave the device to the server for processing. Combining Federated Learning (FL) and Split Learning (SL) offers a promising solution by enabling effective machine learning on resource-constrained devices while preserving user privacy. However, the dynamic nature of IoT resources and device heterogeneity can complicate the application of these solutions, as some IoT devices cannot complete the training task on time. To address these concerns, we have developed a Dynamic Split Federated

Learning (DSFL) architecture that dynamically adjusts to the real-time resource availability of individual clients. DSFL ensures optimal utilization of resources and efficient training across heterogeneous IoT devices and servers. Our architecture detects each IoT device's training capabilities by identifying the number of layers it can train. Moreover, an effective Genetic Algorithm process strategically selects the clients required to complete the split federated learning round. Cooperatively, the clients and servers train their parts of the model, aggregate them, and then combine the results before moving to the next round. Our proposed architecture enables collaborative model training across devices while preserving data privacy by combining FL's parallelism with SL's dynamic modelling. We evaluated our architecture on a sensory dataset, showing improved accuracy and reduced overhead compared to baseline methods.

5.2 Introduction

Federated Learning (FL) involves distributing the training of machine learning models across multiple IoT client devices for enhanced predictions while preserving privacy. However, training on resource-constrained devices can overwhelm their limited processing power, memory, and storage, leading to inefficiencies and performance degradation. This leads to delays, as all devices must wait for the slower ones to finish their training tasks, slowing down the entire process (Zhu *et al.*, 2024). Although FL aims to leverage decentralized data, it often fails to consider the resource limitations of clients, leading to inefficiencies when aggregating local models into a global model (Chahoud *et al.*, 2023; Wahab *et al.*, 2021; Rahman *et al.*, 2020; Wazzeh *et al.*, 2022a). Additionally, devices may be preoccupied with other tasks, further reducing the resources available for model training and delaying subsequent server-side aggregation steps, which are crucial for timely training (AbdulRahman *et al.*, 2020a). Selecting clients without considering resource constraints may lead to stragglers, in which slower devices restrict overall progress. Effective client selection should prioritize balancing resource usage to ensure that devices can complete their training tasks without overburdening, thus avoiding stragglers in the learning process.

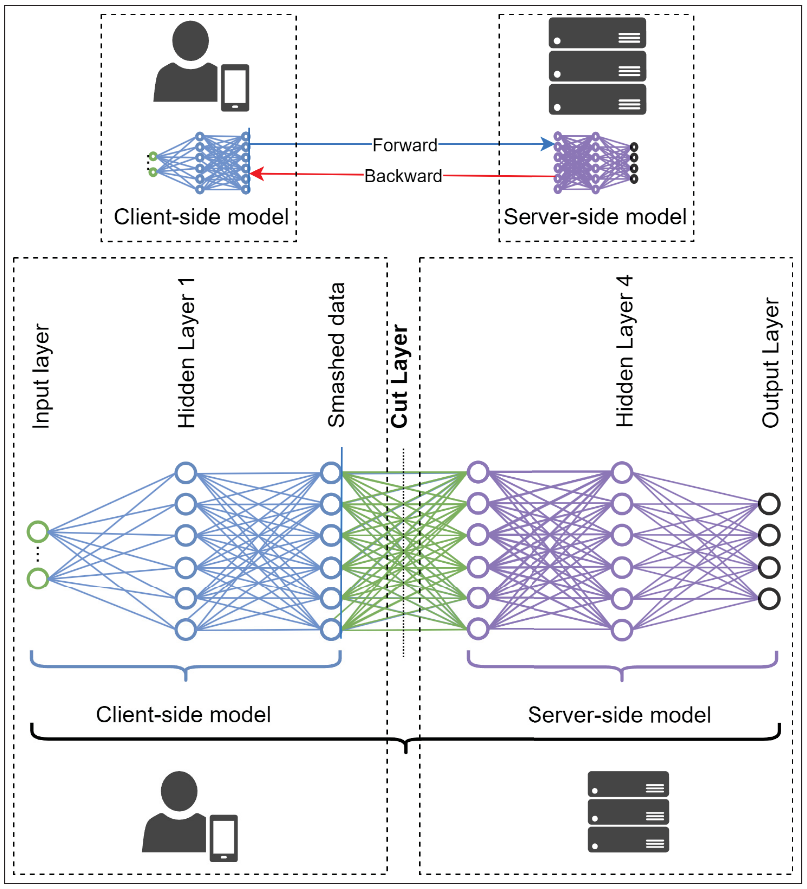

Figure 5.1 Split learning between client and server

The Split Learning (SL) methodology addresses the on-device resource issues by splitting the neural network into two fixed segments of layers. In SL, as illustrated in Figure 5.1, the model is divided such that the client is responsible for training only up to a specific number of layers, known as the split point or the cut layer. At the same time, the server handles the rest of the layers. In a two-party setup, the client processes sample features during the forward pass, generating output weights sent to the server, called the smashed data. The server then computes gradients during the backward pass and transmits them back to the client in a backward propagation, which updates its client-side model accordingly. This partitioning ensures that clients only work with a subset of the layers, reducing resource demands and enhancing model security (Vepakomma *et al.*, 2018). By offloading the deeper layers of the model to the server, SL helps manage client resource constraints and makes the training process more efficient (Wazzeh *et al.*, 2023).

The main drawback of SL is that it requires clients to adhere to a fixed model layer architecture. This limitation becomes impractical in dynamic environments with variable client resources, leading to inefficient training and increased dropout rates(Samikwa *et al.*, 2024). To mitigate these challenges, previous works (Thapa *et al.*, 2022; Gao *et al.*, 2020) suggest combining FL and SL, maintaining decentralization benefits while enabling parallel local training and reducing overhead inefficiencies. To address resource limitations and inefficiencies, we propose Dynamic Split Federated Learning (DSFL), which integrates both SL and FL. DSFL enables clients to adjust the layers they handle dynamically based on their available resources. This flexibility improves resource utilization and reduces server overhead. For instance, a client with fewer resources can handle fewer layers, enhancing overall training efficiency across the network. While FL addresses privacy concerns by enabling training a shared model across decentralized devices without centralizing data (McMahan *et al.*, 2017b), it often encounters challenges related to resource limitations and inefficiencies. To overcome these challenges, client selection ensures that the most suitable devices participate based on their resource availability. We optimize this selection process using a GA that aligns with predetermined objectives, enhancing resource allocation and improving overall system performance (Li, Chen & Teng, 2024). Our approach leverages dynamic split points, allowing clients to adjust their training layers based on available resources. This optimization improves resource utilization and training performance. Incorporating parallel client training via FL reduces training time and enhances overall efficiency (Zhu *et al.*, 2024). Our contributions can be summarized as follows:

1. Implementing a novel Dynamic Split Federated Learning (DSFL) architecture for IoT devices, we integrate split and federated learning with model split. This approach minimizes client training overhead and optimizes resource utilization by allowing clients to adaptively handle model layers based on their available resources.
2. Predicting the cut layer using a machine learning model to optimize the allocation of computational tasks between clients and servers, thereby minimizing training time and ensuring efficient resource allocation.
3. Devising a GA approach with multiple objectives to manage resource constraints, data inconsistencies, and other essential factors by choosing clients based on predetermined

objectives. This approach manages the overhead burden between clients and servers, enhancing overall system efficiency.

4. Evaluating key performance metrics such as the processing overhead, training time, data communication overhead, client idle time, client dropout rate, accuracy loss, and the number of selected clients was crucial for a comprehensive assessment of our approach. These metrics offer valuable insights into the system's efficiency, effectiveness, and practical implications, emphasizing its influence on resource management and overall system performance.
5. Conducting various experiments using benchmark techniques to compare and analyze our approach against centralized learning, SL and resource-aware split federated learning (RSFL).

Results demonstrate that our method achieves nearly the same level of accuracy as the centralized approach while significantly reducing client idle time and improving model convergence and training time. The experiment results show that our method outperforms baseline techniques during training, with clients requiring less training time and exhibiting superior stability. To validate our approach for optimizing resource allocation and client selection, we utilized a Fall Events and Daily Activities dataset (Ojetola, Gaura & Brusey, 2015). Although our study primarily focused on the dataset to evaluate the framework's efficacy, our approach is versatile and applicable to various sensor-based datasets facing resource constraints.

5.3 Literature

Given the evolving nature of IoT devices and resource-constrained environments, there is a crucial need to enhance training processes, reduce overhead burdens, and strengthen security measures. Effectively addressing these concerns requires researchers to explore innovative techniques capable of overcoming the challenges presented by IoT devices. This section overviews current research on distributed learning techniques within IoT devices and resource-constrained systems.

In the study by (Thapa *et al.*, 2022), the authors implement Split Federated Learning to leverage the benefits of distributed computing and parallel client-side training. Their approach aggregates model weights, enabling clients to train their models in parallel with split and federated servers, thereby overcoming the sequential training bottleneck in traditional Split Learning. However, the authors do not detail how the fixed split point between the client and server was selected, nor do they propose a strategy for client selection. The study described in (Wu *et al.*, 2023) introduces a cluster-based Split Federated Learning approach to mitigate training latency in diverse IoT environments. This method employs a predetermined split point for clusters and assigns clients based on radio spectrum availability and channel conditions. However, the research indicates that the clusters had to be operated sequentially, and the client selection process within each cluster was not optimized. Furthermore, the study suggests that the resource allocation of devices and the idle time of clients were not clearly defined, potentially impacting training efficiency and increasing client idle time. In (Samikwa *et al.*, 2024), the authors proposed the Dynamic Federated Learning (DFL) approach to tackle the challenges associated with heterogeneous data and resource limitations in edge IoT environments. DFL combines resource-aware split computing for deep neural networks with dynamic clustering of participants based on layer-wise neural network similarity using Centered Kernel Alignment. The authors did not address a client selection strategy within each cluster. This oversight could lead to delays, as clients in different clusters have to wait for others to complete their tasks, and the presence of just one slow performer could restrict the progress of the entire training round. In the work of (Wazzeh *et al.*, 2024b), the authors proposed a Resource-aware Split Federated Learning approach for Fall Detection in the Metaverse. They combined SL and FL to train machine learning models for users, selecting clients based on their available resources. However, their approach used a fixed split point for all clients, which did not account for device heterogeneity. This limitation is significant because it fails to adapt to variations in client capabilities, potentially leading to inefficiencies. For example, clients with lower resource availability may struggle to perform training effectively if assigned more complex layers or tasks. As a result, these clients might not complete the training on time, causing delays and imbalances in the overall learning process. Additionally, no optimization for client selection was implemented beyond considering resource availability, which could increase the overhead burden on the server to manage the server-side models for the clients. Without dynamic adjustment of split points and optimization strategies, the system may experience sub-optimal performance and resource allocation, impacting the efficiency and effectiveness of the federated learning process. In their work, (Wazzeh *et al.*, 2024a) introduces a cluster-based resource-aware Split Federated Learning approach, which involves filtering and clustering clients based on their resource capacities. Client selection is refined within each cluster using a heuristic approach for specific objectives, such as continuous user authentication. This method utilizes split and federated learning to facilitate parallel client training. However, the approach does not incorporate dynamic split points, leading to limitations in flexibility. As a result, clients with varying resources are constrained to train the same client-side model, which restricts the ability to train additional layers or the entire model on the client side.

Given the growing significance of ensuring safety and timely intervention in health monitoring, fall detection has emerged as a crucial area of research. Our work involves using a fall dataset to investigate different methods for optimizing detection algorithms and sensor usage. Recent studies on fall detection have delved into various approaches to enhance detection algorithms and sensor utilization. These works, which include testing models such as LSTM-RNN, CNN, Decision Trees, and k-NN (Bharathkumar *et al.*, 2020; Hussain *et al.*, 2019; Chelli & Pätzold, 2019), have demonstrated high accuracy and effective sensor use. However, these approaches often rely on fixed model configurations and do not adequately address the challenges of adapting to dynamic client capabilities. This oversight can result in inefficiencies during deployment, particularly in heterogeneous environments where device capabilities vary (Hemmatpour *et al.*, 2019).

The existing literature lacks comprehensive coverage of the challenges associated with utilizing low-end devices that have restricted resources in the context of machine learning. Previous studies have primarily focused on traditional machine learning and federated learning techniques, which often demand significant resources and are unsuited for devices with limited computational power, memory, and storage capacity. These limitations present obstacles to deploying complex machine-learning models or algorithms for various tasks. Furthermore, devices of this nature typically

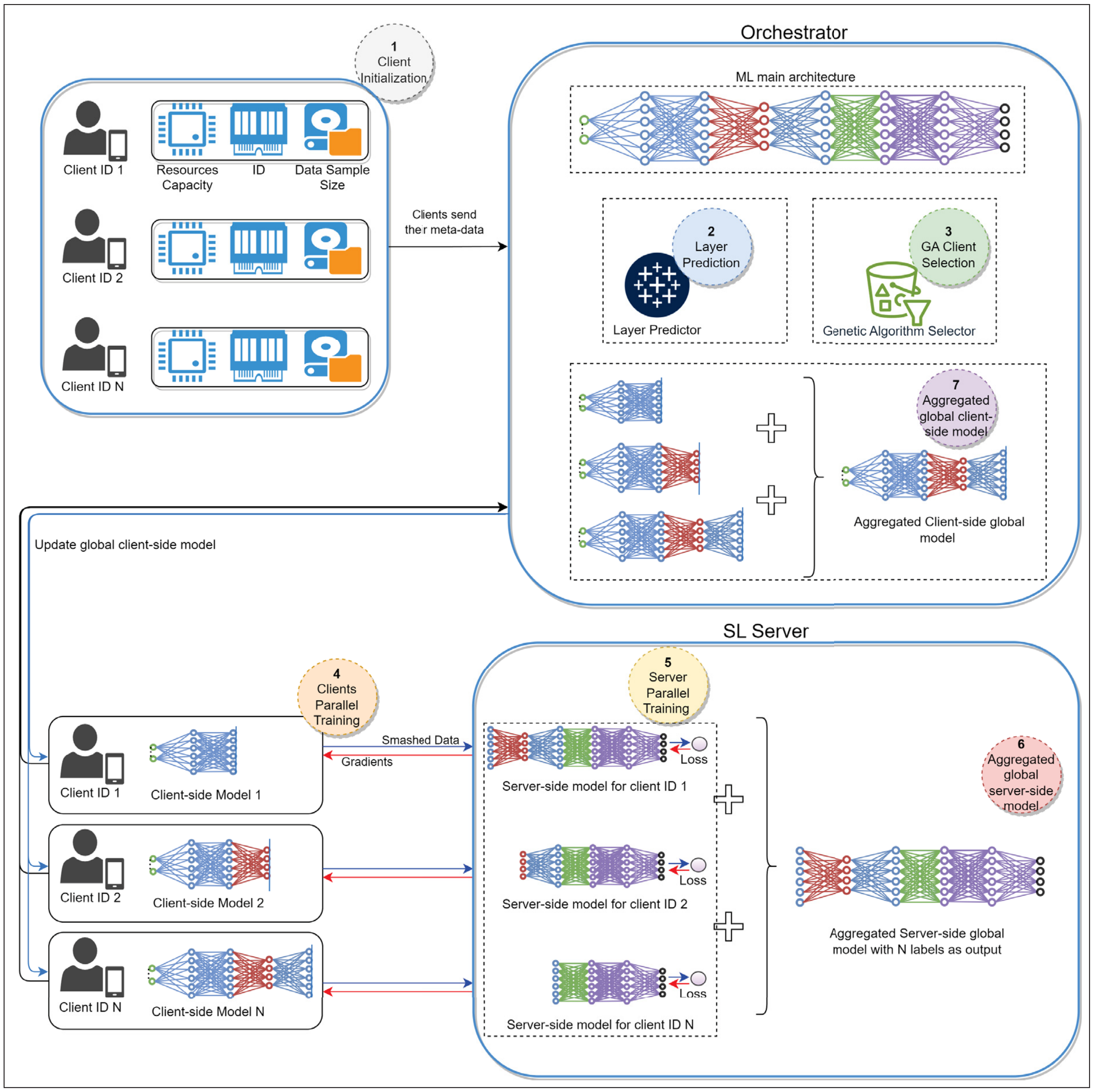
1
Client Initialization
Client ID 1
Resources Capacity
ID
Data Sample Size
Client ID 2
Client ID N
Clients send ther meta-data
Orchestrator
ML main architecture
2
Layer Prediction
Layer Predictor
3
GA Client Selection
Genetic Algorithm Selector
7
Aggregated global client-side model
Aggregated Client-side global model
Update global client-side model
SL Server
4
Clients Parallel Training
5
Server Parallel Training
Smashed Data
Gradients
Client ID 1
Client-side Model 1
Client ID 2
Client-side Model 2
Client ID N
Client-side Model N
Loss
Server-side model for client ID 1
Loss
Server-side model for client ID 2
Loss
Server-side model for client ID N
6
Aggregated global server-side model
Aggregated Server-side global model with N labels as output

Figure 5.2 Dynamic Split Federated Learning architecture with multiple clients

have limited battery capacities, presenting challenges for sustained operation. Additionally, slower processors restrict real-time processing capabilities, leading to delayed responses.

5.4 Methodology

5.4.1 Overview

We have designed an innovative architecture integrating split and federated learning on IoT devices to train machine learning models efficiently. This approach prioritizes user privacy while enabling dynamic resource optimization by harnessing the benefits of distributed computing. The architecture, depicted in Figure 5.2, highlights the critical key players of our framework. The architecture shows that each client within the architecture is defined by its resource capacities, including memory, disk space, processing power, and a unique identifier and data sample size. The orchestrator manages a predictive machine learning model trained on this metadata, essential for determining the optimal split point in training rounds. The split point selection is crucial, as it adjusts the computational burden based on the available resources, ensuring efficient utilization of each client's capabilities. The orchestrator collaborates with clients during each training round. Meanwhile, the split learning server manages the server-side model, allowing clients to train their models effectively with minimal resource strain.

To effectively tackle the challenges posed by diverse client profiles and varying resource availability, we strategically utilize a GA heuristic to choose clients from the available pool. Predefined objectives guide this selection process to ensure the most suitable clients participate in each training round based on their current resources. Through optimizing client selection, the GA method boosts overall system efficiency, reduces training time, and minimizes idle periods, thereby enhancing the effectiveness of the learning process.

In the subsequent section, we will interpret each of the architectural elements and their respective functions in greater detail.

5.4.2 Framework Architecture

Our proposed approach, DSFL, integrates multiple key players to train dynamic machine learning models. These include clients, an orchestrator, and SL servers. Clients are individual devices or entities that train local models on their data and contribute to the overall model. The orchestrator manages the overall training process, coordinating between clients and servers. The SL server aggregates client updates and refines the model. The architecture involves stages such as Client Initialization, Layer Predictor, GA Client Selection, Client Parallel Training, Server Parallel Training, SL Server Aggregator, and FL Server Aggregator. By examining each stage, we comprehensively describe preparing for decentralized training within DSFL scenarios, engaging all parties.

5.4.2.1 Client Initialization

The clients within our framework represent IoT devices, encompassing any device that connects to the internet, gathers data, and possesses the necessary resources to train a machine learning model. The metadata for device resource capacity includes memory, processing power, and disk space. While these resources remain constant, their availability can fluctuate between rounds. The initialization process begins with the clients sending their metadata to the orchestrator for processing. Each client device is identifiable by a unique ID, which is used for all subsequent client-server communications. The device ensures that no raw private data is transmitted during training, thus preserving the client's data privacy.

5.4.2.2 Layer Predictor

The Layer Predictor is a critical component of our framework. It estimates the training time of each client's model layer and selects the optimal configuration to minimize model idle time. It customizes the machine learning model based on each client's metadata, such as data sample size, available resources (memory, processing power, disk space), and the number of global model layers, ensuring the training process is optimized for each client's environment. To train the Layer Predictor, clients provide their metadata to the orchestrator, generating a local dataset that reflects each client's environment. The Layer Predictor uses this dataset to train a machine-learning model that estimates training durations for various resource and layer combinations. During inference, the Layer Predictor predicts the optimal cut layer for each client based on their current metadata, further reducing idle time and enhancing efficiency.

5.4.2.3 GA Client Selection

Selecting clients is essential for effectively training our machine learning model, as it ensures that only the most suitable participants are involved. This initial selection process filters potential clients to confirm they possess the necessary resources, such as memory, processing power, and disk space, to complete the training without early dropouts, thereby enhancing reliability and efficiency. Additionally, it prevents the inclusion of clients who may not meet the resource requirements, thereby reducing potential disruptions. Following the initial filtering, we employ a heuristic selector algorithm that is crucial in our client selection process. This algorithm evaluates the remaining clients based on predefined objectives, including resource availability, data sample size, and idle time, to identify the optimal subset for training. The heuristic approach is crucial as it systematically balances these factors to maximize training efficiency and effectiveness, addressing the complexities of resource constraints and ensuring an optimal distribution of computational tasks. This method improves the overall performance of the distributed training environment by making informed decisions on client selection based on a combination of resource availability and cost considerations.

5.4.2.4 Client-Server Training Synchronization

Once the client selector has identified suitable clients for selection, these clients receive the latest client-side machine-learning model from the orchestrator. Clients train an ML model independently and in parallel using their local device and dataset while the SL server integrates and processes the models in sequence. Upon completing the training for their respective layers, each client transmits the results of their cut layer or the processed data, known as smashed data, to the split server layer via forward propagation. The split server then takes over to complete the training for the remaining model segments, which involves calculating the loss function to evaluate the model's performance. After the split server has completed its training part, it sends the calculated gradients back to the clients. This feedback allows the clients to resume the backward propagation process, using the received gradients to update their local model weights.

5.4.2.5 SL and FL Server Aggregation

The model aggregation process is carried out independently after training client and server-side models. The orchestrator aggregates the client-side weights by averaging them, combining knowledge from all participating clients into a unified client-side global model. This global model reflects the collective training from the client side. At the same time, the SL server handles the aggregation of server-side model weights for the layers it has trained. It ensures that local updates are integrated into its portion of the model, completing the global aggregation overseen by the orchestrator. Once the aggregation processes are complete, the orchestrator sends the updated client-side global model to the clients. Training, aggregating, and distributing models continue iteratively until convergence is achieved, ensuring the final global model reaches optimal performance through collective learning from all clients. This systematic approach ensures that the most suitable clients receive the updated models and start the training process, thus enhancing the efficiency and accuracy of subsequent training rounds.

5.5 Problem Definition & Formulation

This section outlines the problem definition and presents its formulation to address it. We explore the layer prediction and client selection, offering a structured framework for our approach. Additionally, we analyze the client selection algorithm and its associated complexity.

5.5.1 Problem Definition

Layer Predictor: The Layer Predictor addresses the challenge of optimizing layer configurations for client-side models in distributed learning environments. The problem involves selecting the most appropriate layer configuration for each client, considering the client's unique resources and data sample sizes. The goal is to predict the best configuration that maximizes model performance while accounting for the client's meta-data and training time. This solution facilitates efficient training across diverse client environments, aiming to enhance the client's ability to train a model effectively while safeguarding data privacy.

Client Selection: The Client Selection problem considers the selection of a subset of clients to take part in the training process of a distributed learning model. Each client has distinct resource availability and data samples in every learning round. Each round aims to select clients that maximize overall learning effectiveness and reduce any delays in the training process. This involves increasing model diversity by involving a wide range of clients, increasing the model's knowledge by including larger data samples and minimizing differences in idle time between clients to ensure a faster and more synchronized training process. The selection process must balance these objectives by a weighting factor to optimize the model's performance in each training round.

Our framework can dynamically assign clients to specific model split points based on their current capabilities. This selection process identifies clients for split federated rounds to efficiently train distributed machine learning models while addressing the challenges of device heterogeneity, such as varying resources and data sample sizes. The objectives of this process are as follows:

1. Maximize Client Participation: Aim to include the highest possible number of client devices in each learning phase. This approach ensures wide participation and enhances the diversity of data contributions, thus strengthening the overall learning process.
2. Maximizing Learning Effectiveness: Enhancing the model's learning quality through strategically selecting clients with significant data samples. This selection guarantees that the training process leverages a more diverse and richer dataset, resulting in a more resilient and precise model.
3. Minimizing Client Waiting Time: The optimization of the SL process involves the careful selection of clients with comparable processing durations to reduce idle time and guarantee the efficient utilization of all participating devices in parallel.

The specified requirements transform the problem into a multi-objective optimization challenge similar to the multi-objective Knapsack problem (Bazgan, Hugot & Vanderpooten, 2009). In this context, the objectives are to maximize the number of participating clients, enhance learning quality by selecting clients with required data samples, and minimize client idle time by choosing clients with similar training times. Balancing these objectives is crucial, as improvements in one area may impact the others, requiring a strategy that effectively addresses these competing goals to achieve an optimal solution.

Multi-objective Knapsack Problem: The Multi-objective Knapsack Problem (MOKP) requires optimizing multiple conflicting objectives simultaneously, in contrast to the traditional Knapsack problem, which focuses on a single objective. While the standard Knapsack problem involves selecting items to maximize total value within a weight constraint, the MOKP deals with multiple objectives, where each item can contribute to various objectives with different values and capacities.

Theorem 1. *In the context of Split Federated Learning, addressing the client selection problem by formulating it as a Multi-objective Knapsack Problem shows that it falls within the NP-hard complexity class.*

Proof: To demonstrate the NP-hardness of our problem, we establish its reduction from the well-known Knapsack Problem to its multi-objective version. Given an instance of the Knapsack Problem with items, weights, and values, we create a corresponding multi-objective optimization problem in the following manner:

1. *Items:* In our scenario, clients represent the items in the Knapsack Problem. Each client is associated with a binary decision variable signifying its selection in a round.
2. *Weights:* The weights in our client selection problem represent the selection cost, including factors such as a client's resource availability for a given round, the size of the client's data sample, and the reduction of overall client idle time.

3. *Values:* The values in our problem are determined by client qualities, including resources, data sample size, and training time. These characteristics define the client's fitness, akin to item values in the Knapsack Problem.

Our goal in this multi-objective optimization problem is to maximize the fitness of selected clients. This objective is crucial, as it directly impacts the performance of the Split Federated Learning system. The constraints ensure this goal is achieved, demonstrating the importance of the client selection problem in split federated learning and its NP-hard nature.

5.5.2 Problem Formulation

This subsection will present a mathematical formulation of the problem, providing a detailed description of both model layer split point prediction and the client selection process. Furthermore, it will outline the objectives of these processes to achieve optimization. Table 5.1 summarizes some essential notations used.

5.5.2.1 Layer Predictor Formulation

Let $C = \{C_1, C_2, \ldots, C_N\}$ be a set of clients, where N represents the total number of clients. Each client C_i is characterized by the following resource capacities:

$\mathcal{P}_i$	: Processing capacity
$\mathcal{H}_i$	: Memory capacity
$\mathcal{D}_i$	: Disk space
$\mathcal{S}_i$	: Data sample size

These resource capacities $\mathbf{r}_i = \{\mathcal{P}_i, \mathcal{H}_i, \mathcal{D}_i, \mathcal{S}_i\}$ for each client C_i are sent to Layer Predictor module. To generate the local dataset $\mathcal{D}_i^{\text{local}}$, the capacities are randomized based on their maximum values, allowing for variation in the dataset creation. This local dataset $\mathcal{D}_i^{\text{local}}$ is then used to train a Regressor machine learning model noted as $\mathcal{R}$. The regressor model aims to

Table 5.1 Summary of LaTeX Notations

Notation	Description
$C = \{C_1, C_2, \ldots, C_N\}$	Set of clients where N is the total number of clients
C_i	Individual client
$\mathcal{P}_i$	Processing capacity of client C_i
$\mathcal{H}_i$	Memory capacity of client C_i
$\mathcal{D}_i$	Disk space of client C_i
$\mathcal{S}_i$	Data sample size of client C_i
$\mathbf{r}_i = \{\mathcal{P}_i, \mathcal{H}_i, \mathcal{D}_i, \mathcal{S}_i\}$	Resource capacities of client C_i
$\mathcal{D}_i^{\text{local}}$	Local dataset for client C_i
$\mathcal{R}$	Regressor machine learning model used for prediction
$\mathcal{L}_i^*$	Optimal layer configuration for client C_i
$T_i(\mathcal{L})$	Predicted training time for layer configuration $\mathcal{L}$ on client C_i
T^{idle}	Idle time
C_{selected}	Set of selected clients
w_1, w_2, w_3	Weight factors for client selection
f_1, f_2, f_3	Objective functions for client selection
$F(x)$	Combined multi-objective function
Θ	Model weights

predict the optimal layer configuration $\mathcal{L}_i^*$ for each client C_i, based on the provided resource capacities $\mathbf{r}_i$ and the split point layer of the model. The model $\mathcal{R}$ serves as a predictive tool for estimating the training time $T_i(\mathcal{L})$ associated with different layer configurations $\mathcal{L}$, taking into account each client's resource capacities $\mathbf{r}_i$. To identify the optimal layer configuration $\mathcal{L}_i^*$ for each client C_i, the following procedure is employed:

a. Prediction of Training Times: For each client C_i, the pre-trained regressor model $\mathcal{R}$ is utilized to predict the training time $T_i(\mathcal{L})$ for all possible layer configurations $\mathcal{L}$, based on the client's resource constraints $\mathbf{r}_i$.

b. Determination of Optimal Configuration: The optimal layer configuration $\mathcal{L}_i^*$ is selected by minimizing the discrepancies in training times across clients. This approach aims to assign split points to equalize the training duration among clients while preserving the model's effectiveness.

Adhering to this methodology optimizes the selection of layer configurations across clients to balance training times and resource utilization, thereby enhancing the distributed model's overall performance.

The process is as follows:

i. For each client, retrieve the maximum available resources and sample size.

ii. Use the pre-trained model to estimate the training time for each layer selection.

iii. Store the predicted training times and the corresponding layer for each client.

iv. Compute the median training time across all layer selection and clients.

v. For each client, identify layer selection where the predicted training time is less than or equal to the median time. Among these configurations, select the one with the maximum number of layers to maximize client efficiency and minimize server overhead while considering client idle time during the selection process.

vi. If no layer selection has a training time within the median range, choose the layer with the training time closest to the median. In this case, prioritize the settings with the highest number of layers to balance client efficiency and minimize server overhead, ensuring client idle time is also considered.

The optimal layer configuration $\mathcal{L}_i^*$ for each client C_i is determined using the following approach. This ensures that the selected configuration balances efficiency and performance by minimizing training time while accounting for the resource constraints of each client.

The LaTeX code below illustrates how to represent this process mathematically:

$$\mathcal{L}_i^* = \arg\min_{\mathcal{L}} \left(|T_i(\mathcal{L}) - \tilde{T}(\mathcal{L})| \right) \tag{5.1}$$

Where:

- $T_i(\mathcal{L})$ represents the predicted training time for the i-th client with layer configuration $\mathcal{L}$.
- $\tilde{T}(\mathcal{L})$ denotes the median of the predicted training times across all clients and layer configurations.

This approach ensures that the optimal layer configuration is selected based on minimizing the difference between the client's predicted training time and the median training time across configurations, thus promoting balanced training durations and effective utilization of client resources.

5.5.2.2 Client Selection Problem

Algorithm 5.1 Client Filtering Algorithm based on Resource Availability and Model Requirements

```
Input  : $\mathcal{P}_i^n$, $\mathcal{H}_i^n$, $\mathcal{D}_i^n$ (client resources) and $M_{\mathrm{p}_i}^l$, $M_{\mathrm{m}_i}^l$, $M_{\mathrm{d}_i}^l$ (model requirements) for
         each client $C_i$ and model layer $M_i^l$, for each round $n$
Output : $\mathcal{A}^n$ (selected clients) for each round $n$
1  for each round $n$ do
2      Initialize $\mathcal{A}^n$ as an empty set;
3      for $i \leftarrow 1$ to $I$ do
4          Initialize availableResources based on client resources $\mathcal{P}_i^n$, $\mathcal{H}_i^n$, $\mathcal{D}_i^n$ of $C_i$
             for round $n$;
5          Initialize requiredResources based on $M_{\mathrm{p}_i}^l$, $M_{\mathrm{m}_i}^l$, $M_{\mathrm{d}_i}^l$ for $M_i^l$;
6          if $\mathcal{P}_i^n \geq M_{p_i}^l \wedge \mathcal{H}_i^n \geq M_{m_i}^l \wedge \mathcal{D}_i^n \geq M_{d_i}^l$ then
7              Add $C_i$ to $\mathcal{A}^n$;
8          end if
9      end for
10     $\mathcal{A}^n \leftarrow$ Client selection algorithm using available and required resources;
11 end for
12 return $\mathcal{A}^n$ for each round $n$
```

A client's resource capacities $\mathcal{P}_i^n$, $\mathcal{H}_i^n$, and $\mathcal{D}_i^n$ are used to determine if it can host a model M_i^l, where l is the number of the client-side model assigned layer, and L is the total number of layers in the model architecture. The model deployment requirements in terms of processing speed, memory, and disk space are represented as follows:

$$M_i^l = [M_{\mathrm{p}_i}^l, M_{\mathrm{m}_i}^l, M_{\mathrm{d}_i}^l], \forall i \in \{1, \dots, I\} \tag{5.2}$$

Where:

$M^l_{\mathbf{p}_i}$: Processing usage requirement of model M^l_i

$M^l_{\mathbf{m}_i}$: Memory requirement of model M^l_i

$M^l_{\mathbf{d}_i}$: Disk space requirement of model M^l_i

In each round n, the subset $\mathcal{A}^n$ is used to select C_i if the resources for loading M^l_i are available. A model M^l_i can be hosted on C_i in round n if the following constraints are satisfied:

$$\mathcal{A}^n = \{i \in I : \mathcal{P}^n_i \geq M^l_{\mathrm{p}_i}, \mathcal{H}^n_i \geq M^l_{\mathrm{m}_i}, \mathcal{D}^n_i \geq M^l_{\mathrm{d}_i}\} \tag{5.3}$$

Algorithm 5.1 outlines the steps in filtering clients for training. The primary goal of the algorithm is to identify an optimal subset $\mathcal{S}^n$ of clients C_i from the overall client pool C and deploy suitable models to these clients in each round n. The algorithm ensures that the required resources for the client models are satisfied for every client. It does this by comparing each client's current available resources (such as memory, processing power, and storage) with the resources required by the model. This comparison ensures that only clients capable of effectively training the assigned model are selected, optimizing the client's heuristic pre-selection of clients.

The client selection process in each round aims to optimize the overall training efficiency and knowledge acquisition across the participating clients. Without a GA, more straightforward approaches like random or resource-based selection may lead to imbalanced workloads, inefficient resource utilization, and slower model convergence due to uneven client contributions. After passing through the Layer predictor to determine the appropriate model layer $\mathcal{L}_i$ for each client C_i, the selection of clients $C_{\text{selected}} \subseteq C$ is guided by a heuristic GA that considers the following three objectives:

a) Maximize Client Selection The first objective is to maximize the number of selected clients $|C_{\text{selected}}|$ to enhance the diversity and robustness of the learned model. The goal is to include as many clients as possible to increase the distributed knowledge gained across different environments. This objective can be mathematically expressed as:

$$f_1 = \max_{C_{\text{selected}}} (w_1 \cdot |C_{\text{selected}}|) \tag{5.4}$$

Where w_1 is the weighted factor assigned to this objective.

b) Maximize Data Sample Size: The second objective is to maximize the total data sample size across all selected clients, which enriches the model's learning process by exposing it to a larger and more diverse dataset. This is formulated as:

$$f_2 = \max_{C_{\text{selected}}} (w_2 \cdot \sum_{C_i \in C_{\text{selected}}} \mathcal{S}_i) \tag{5.5}$$

Where $\mathcal{S}_i$ denotes the data sample size of the client C_i, and w_2 is the weighted factor for the data sample size objective.

c) Minimize Idle Time Differences: The third objective focuses on reducing the disparity in idle time among clients during the training process. Idle time is defined as the duration a client remains inactive or waiting for other clients to complete their tasks. By minimizing these differences, we aim to synchronize client activities more effectively, accelerating the overall training process in each round. This objective is mathematically formulated as follows:

$$f_3 = \min_{C_{\text{selected}}} (w_3 \cdot \sum_{C_i, C_j \in C_{\text{selected}}} \left| T_i^{\text{idle}} - T_j^{\text{idle}} \right|) \tag{5.6}$$

Here, T_i^{idle} represents the idle time for client C_i, which is the time the client spends not actively participating in the training process. The term w_3 is a weighted factor that adjusts the importance of minimizing idle time differences relative to other objectives. By minimizing the sum of absolute differences in idle times between all pairs of selected clients, this objective helps ensure that no client is disproportionately inactive compared to others, leading to more balanced and efficient training rounds.

The multi-objective optimization problem is represented as:

$$F(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) - w_3 \cdot f_3(x) \tag{5.7}$$

Where:

$$\begin{aligned} f_1(x) &= \text{Minimize Training Time Differences} \\ f_2(x) &= \text{Maximize Data Sample Size} \\ f_3(x) &= \text{Minimize Idle Time Differences} \end{aligned}$$

Multi-objective optimization problems often result in multiple Pareto-optimal solutions instead of a single best solution. The Pareto set represents these optimal trade-offs (Murata *et al.*, 1995), and efficiently obtaining this Pareto set is crucial for solving such problems. Genetic Algorithms are particularly effective for this purpose because they can efficiently explore the solution space. GAs simulate the natural selection process by favouring the reproduction of the fittest solutions across generations (Konak *et al.*, 2006). In contrast, non-GA approaches must thoroughly explore a vast solution space, which can be computationally intensive and time-consuming as the number of clients increases exponentially.

In this particular context, each chromosome within the GA is denoted as a matrix C_{selected}, which signifies the decisions made by the optimization model regarding client selection for model training. The matrix contains values within the range of $[0, 1]$, representing the binary nature of client inclusion in the training process. This binary representation simplifies the optimization process within the framework of the GA.

The optimization process begins with the **Initialization** phase, where a set of potential solutions, or chromosomes, is generated from the available clients for training. Each chromosome, representing a subset of clients, is created by encoding individual clients as genes, ensuring that each gene within a chromosome is unique to promote diversity.

In the subsequent **Selection** phase, the population is filtered to retain only the top-performing individuals, who are then chosen as parents for the next generation.

This selection leads to the **Crossover** phase, where genetic material from two parent chromosomes is combined to produce offspring solutions. The chromosomes are split at randomly chosen positions, and segments are exchanged based on a crossover ratio, typically set at 0.5, to create new genetic combinations.

The **Mutation** phase introduces random alterations into the offspring solutions to preserve genetic diversity and replace some individuals in the current population, forming the next generation.

The process concludes based on **Termination** criteria, usually defined by reaching a maximum fitness level or completing a set number of generations. This results in a final chromosome representing a selection of clients meeting the specified objectives.

5.6 DSFL Framework

This section discusses algorithms used and optimization strategies that showcase how our architecture works together. As applications that rely on data evolve, it is crucial to ensure that model updates are efficient, communication overhead is minimized, and resources are optimized. Algorithm 5.2 provides an overview of each component, its function, and the client training procedure.

The DSFL algorithm is developed to enable distributed learning across diverse client devices with varying resource capacities. This algorithm combines client-side and server-side operations to divide model training tasks, ensuring efficient utilization of available resources. The DSFL process begins by initiating a loop that iterates over multiple communication rounds, as in line 23. The algorithm performs several critical functions in each round, including server initialization, client-side training, server-side training, and server-side aggregation. The primary objective is

Algorithm 5.2 Dynamic Split Federated Learning Algorithm

```
Input  : r, $\Theta^{(n)}_{\text{global}}$
Output : $\Theta_{\text{final}}$
1  Function Init Server($C, \Theta^{(n)}_{global}$)
       // Orchestrator Server-side operations
2          Gather client resource data: $\mathbf{r}_i$ for each $C_i \in C$
3          Run Layer Predictor to determine $\mathcal{L}^*_i$
4          Run client selection optimization to get $C_{\text{selected}}$
5          Transmit $\Theta^{(n)}_{\text{server}}$ (server-side) and $\Theta^{(n)}_{\text{client}}$ (client-side) to $C_{\text{selected}}$
6      return $C_{\text{selected}}$
7  Function ServerTrain($\{\Theta^{(n)}_{server}, \mathbf{w}^{client}_i\}$)
       // SL Server-side training
8          Initialize server-side model using $\{\mathbf{w}^{\text{client}}_i\}$
9          Train $\Theta^{(n)}_{\text{server}}$ with combined info
10         Update model: $\Theta^{(n+1)}_{\text{server}}$
11     return $\Theta^{(n+1)}_{\text{server}}$
12 Function Client($C_i, \Theta^{(n)}_{server}, \Theta^{(n)}_{client}$)
       // Client-side operations
13         Train $\Theta^{(n)}_{\text{client}}$ locally
14         Extract and transmit client-side weights (smashed data) $\mathbf{w}^{\text{client}}_i$ to SL server
15         $\Theta^{(n+1)}_{\text{server}} \leftarrow$ ServerTrain($\{\Theta^{(n)}_{\text{server}}, \mathbf{w}^{\text{client}}_i\}$)
16         Send the entire client model $\Theta^{(n)}_{\text{client}}$ to orchestrator
17     return $\mathbf{w}^{\text{client}}_i$
18 Function ServerAggregate($\{\Theta^{(n+1)}_{server}, \Theta^{(n)}_{client}\}$)
       // Orchestrator server-side aggregation
19         Aggregate client-side models:
20         $\Theta_{\text{client-final}} = \frac{\sum_{i=1}^{|C_{\text{selected}}|} \mathcal{S}_i \, \Theta^{(n)}_{\text{client}}}{\sum_{i=1}^{|C_{\text{selected}}|} \mathcal{S}_i}$
21         Aggregate server-side models:
22         $\Theta_{\text{server-final}} = \frac{\sum_{i=1}^{|C_{\text{selected}}|} \mathcal{S}_i \, \Theta^{(n+1)}_{\text{server}}}{\sum_{i=1}^{|C_{\text{selected}}|} \mathcal{S}_i}$
23     return $\Theta_{\text{client-final}}, \Theta_{\text{server-final}}$
24 Function DSFL($C, \Theta^{(n)}_{global}$)
       // Learning over multiple rounds
25         for each round n do
26             $C_{\text{selected}} \leftarrow$ Init Server($C, \Theta^{(n)}_{\text{global}}$)
27             for each $C_i \in C_{selected}$ do
28                 $\mathbf{w}^{\text{client}}_i \leftarrow$ Client($C_i$, $\Theta^{(n)}_{\text{server}}$, $\Theta^{(n)}_{\text{client}}$)
29             end for
30             $\Theta_{\text{final}} \leftarrow$ ServerAggregate($\{\Theta^{(n+1)}_{\text{server}}, \Theta^{(n)}_{\text{client}}\}$)
               // Obtain the final aggregated model
31         end for
32     return $\Theta_{\text{final}}$
```

to update and refine the global model progressively, represented as $\Theta^{(n)}_{\text{global}}$, by exploiting the diverse resources of selected clients.

The algorithm begins by initializing the server using the `Init Server` function (lines 1–6). The server then collects resource data $\mathbf{r}_i$ for each client $C_i \in C$, where C denotes the set of all clients. Using this resource data, the server employs a *Layer Predictor* to determine the optimal number of layers $\mathcal{L}_i^*$ for each client. Following this, the server applies a client selection optimization algorithm to identify a subset of clients, referred to as C_{selected}, that will take part in the current training round. The global model is then divided into two parts: $\Theta_{\text{server}}^{(n)}$, representing the server-side portion, and $\Theta_{\text{client}}^{(n)}$, representing the client-side portion. These model components are then sent to the selected clients.

Once the chosen clients have received their respective model components, they proceed with local training, as described in the `Client` function (lines 12–17). Each client C_i trains the client-side model $\Theta_{\text{client}}^{(n)}$ using its local dataset and available resources. Upon completing local training, the client extracts and transmits the resulting client-side weights, referred to as *smashed data* $\mathbf{w}_i^{\text{client}}$, to the SL server. Additionally, the entire client model $\Theta_{\text{client}}^{(n)}$ is sent to the orchestrator to contribute to the aggregation process (line 30).

The SL server-side model training occurs within the `ServerTrain` function (lines 7–11). Here, the server initializes its model using the combined client-side weights $\mathbf{w}_i^{\text{client}}$. Subsequently, the server-side model $\Theta_{\text{server}}^{(n)}$ is trained using this combined information. After the training is complete, the server updates the model to obtain $\Theta_{\text{server}}^{(n+1)}$ for the next round. It is important to note that this process only uses the client's smashed data, ensuring that the server does not depend on the entire client model at this stage.

The final step in each round involves aggregating client-side and server-side models through the `ServerAggregate` function (lines 18–21). The server uses a weighted sum approach to aggregate the client-side models, with each client's model $\Theta_{\text{client}}^{(n)}$ weighted by its respective data sample size $\mathcal{S}_i$. Similarly, the server-side models $\Theta_{\text{server}}^{(n+1)}$ are aggregated using the exact weighting mechanism. The resulting aggregated models denoted as $\Theta_{\text{client-final}}$ and $\Theta_{\text{server-final}}$ represent the finalized outputs of the current round.

The DSFL algorithm goes through multiple rounds, gradually improving the global model by integrating contributions from selected clients in each round. The final result of the DSFL algorithm, denoted as Θ_{final}, represents the combined model produced after all rounds have concluded and convergence has been achieved. The DSFL algorithm effectively tackles the challenges posed by varying resource capacities and dynamic environments in federated learning. By adaptively dividing the model training tasks between clients and the server, the algorithm ensures optimal utilization of each client's resources, enabling efficient and scalable distributed learning.

It is important to note that it is feasible to maintain client labels on the client side using a U-shaped split learning configuration that does not necessitate sharing labels with the server. This configuration requires wrapping the network around the server's end layers and transmitting the outputs back to clients (Vepakomma *et al.*, 2018). However, it is crucial to acknowledge that this approach will increase communication and computational costs for both parties.

Moreover, Differential Privacy (DP) can be employed for feature protection to secure the privacy of weights (Thapa *et al.*, 2022). The training process on both client and server sides occurs in parallel, enabling each client to operate independently without waiting for others. The federated server independently aggregates client-side weights through averaging. The final step involves clients sending their trained local model to the federated server for aggregation. This method effectively combines FL's privacy-preserving advantages with SL's distributed nature by selecting and training clients based on their available resources.

5.7 Experimental Setup

In this section, we present a comprehensive overview of our experimental setup. We enclose a detailed discussion of the data distribution, preprocessing techniques, experiments conducted, and results obtained. Furthermore, we discuss the client-server collaborative approach during the training process.

Our experiments utilized the modular FL framework outlined in (Arafeh *et al.*, 2023a), combined with Split Learning (SL) scripts, and were conducted in a Windows environment using $PyTorch$[1] with CUDA 12.1 on an NVIDIA GeForce RTX 3070 GPU with 8GB VRAM. To replicate the resource profiles of client devices, we used $Docker$[2] containers. Each container was allocated between 2-4GB of memory and varied CPU capacities to simulate diverse device conditions. The experiments were executed on a machine with eight-core processors running at a base speed of 3.59GHz and 32GB of RAM. To mimic the CPU capabilities of the simulated devices, the containers were configured with CPU allocations of 1.5, 2, and 2.5 cores.

5.7.1 Dataset & Setup

We use the Fall Events and Daily Activities dataset (Ojetola *et al.*, 2015) for our experiments. This dataset includes many labels for different activities that provide helpful information about how participants move and behave. There are 32 participants in the dataset, 28 of whom are male and four are female, with ages ranging from 18 to 32 years old. The sensor data was collected at a rate of 100hz and specifically focused on falling-related activities. The data was collected from chest and thigh sensors, providing a complete view of the participants' movements, orientation, and postures. This allows our fall detection method to utilize complex motion patterns for precise analysis. The dataset contains a diverse array of labels that characterize different activities, including standing, walking, lying, sitting on a bed, sitting on a chair, crouching, ascending/descending staircases, and various types of falls (forward, backward, right, left, real falls, near falls).

For our specific study, we concentrate on three key labels:

- **Activities of Daily Living (ADL) (Label 0):** Label 0 designates routine activities of daily life that do not involve falls or near falls.
- **Falls (Label 1):** This label corresponds to instances where a fall occurs during an activity. We denote this label with the value 1, signifying the occurrence of a fall event.
- **Near Falls (Label 2):** Instances categorized as near falls, where participants experience an almost-fall scenario, are assigned the label value 2.

Our data pre-processing pipeline segments the raw sensor data into uniform windows of three seconds each. We use a majority labelling method to assign labels to these segments, which involves identifying the frequently appearing label within each window. This approach ensures that our data is well-structured, essential for obtaining reliable results in our research. Additionally, we apply filtering techniques to eliminate noise from the sensor data and address gyroscope drift. A high-pass second-order Butterworth filter (Novák, Perfilieva, Holčapek & Kreinovich, 2014) is employed to eradicate low-frequency components from the data. These low-frequency components often represent noise or undesirable drift in the sensor readings. We devised 80% of the data for the training and 20% for testing. We randomly mapped client resources using the resource data for a unique client sourced from (Reiss, Wilkes & Hellerstein, 2011). Algorithm 5.3 provides additional information on the client data distributor. It randomly selects continuous samples to capture the sensor pattern during training and testing while minimizing data size to reduce the client's processing burden. Due to the data distribution, clients may have received partial labels or datasets in non-IID form in some instances. This can lead to lower accuracy of the model for individual users. However, when the models of multiple users are aggregated, a more expansive knowledge base is formed, incorporating input from all participants. In addition, Wearable devices come with essential security considerations, particularly when deploying models on these devices (Arias *et al.*, 2015; Ching & Singh, 2016). To ensure that security is maintained at the device level, it is necessary to implement secure boot processes, utilize secure chipsets, and provide secure data storage to safeguard sensitive information. It is also essential to encrypt data transmissions between the wearable device and servers to prevent unauthorized access.

A regressor machine learning model used to predict the best layer for each client was tested against various efficient models, including Random Forest Regression, Linear Regression, and Support Vector Regression (SVR). The evaluation results showed that Random Forest Regression achieved the highest accuracy of 0.92 and the lowest Mean Absolute Error (MAE) of 22.13. In comparison, Linear Regression had an accuracy of 0.66 with an MAE of 48.31, while SVR

Algorithm 5.3 Percentage Data Distributor Algorithm

```
   Input  : Data Dictionary $\Delta$, Percentage $\alpha$
   Output : New Data Dictionary $\Gamma$
1  for each $\kappa$ in $\Delta$ do
2      $\Gamma_\kappa \leftarrow \Delta[\kappa]$;
3      $\nu \leftarrow$ length of $\Gamma_\kappa.x$;
4      $V \leftarrow \lfloor \nu \times \alpha / 100 \rfloor$;
5      $\varsigma \leftarrow$ random integer between 0 and $\nu - V$;
6      $\Upsilon \leftarrow$ list of integers from $\varsigma$ to $\varsigma + V - 1$;
7      $X_\kappa \leftarrow$ list of elements from $\Gamma_\kappa.x$ at indices in $\Upsilon$;
8      $Y_\kappa \leftarrow$ list of elements from $\Gamma_\kappa.y$ at indices in $\Upsilon$;
9      $\Gamma'_\kappa \leftarrow$ create new DataContainer with $X_\kappa$ and $Y_\kappa$;
10     $\Gamma[\kappa] \leftarrow \Gamma'_\kappa$;
11 end for
```

exhibited an accuracy of 0.13 and an MAE of 79.36. These results indicate that Random Forest Regression provides the most reliable predictions for selecting the optimal layer configuration for each client. The data samples, which are window-shaped and form a data matrix, are processed using a convolutional neural network (CNN) model. This model, comprising four convolutional layers, incorporates four 2D convolutional layers to capture features from the input sequences, progressively increasing the number of output channels. A max-pooling layer succeeds each convolutional layer to reduce spatial dimensions. A dropout mechanism is used to prevent overfitting. After the convolutional and pooling layers, the model flattens the data and employs two fully connected layers for final classification. Using rectified linear unit (ReLU) activation functions for non-linearity and a dropout probability of 0.5 enhance the model's generalization capability. This architecture balances performance and efficiency effectively, making it well-suited for handling window-shaped data samples. Multiple works have used CNN architecture to detect falls and near-fall events (Islam *et al.*, 2020). This model is effective in capturing temporal patterns in sequential data. To optimize performance, we fine-tune hyper-parameters and explore different architectural variations. It is worth mentioning that our framework assessment was carried out on the fall dataset. However, as previously emphasized, our methodology can be utilized for any resource-constrained sensor-based dataset.

5.7.2 Comparison with Baseline Approaches

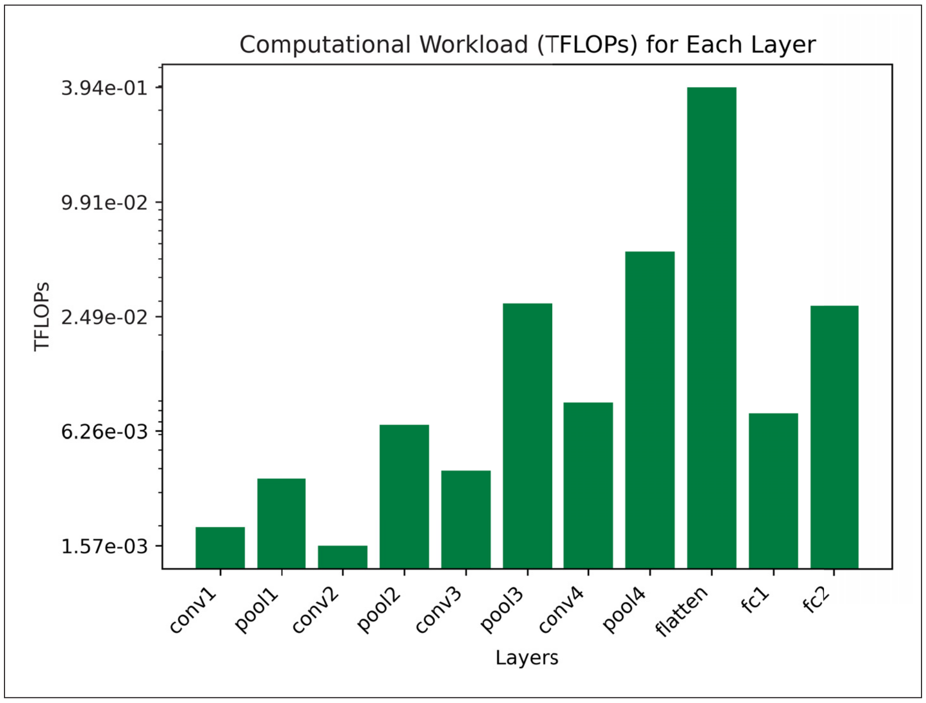

Figure 5.3 Computational Workload (TFLOPs) for Each Model Layer

We conducted comprehensive experiments to demonstrate the effectiveness of our proposed approach. Our experiments encompassed a centralized (Cen) approach as a baseline, SL, RSFL, and DSFL experiments. Additionally, we examined multiple client selection strategies, including random selection, a resource-driven approach based on availability, and a GA selection approach. Our DSFL approach surpassed these methods regarding training efficiency, client communication overhead reduction, and model performance. We employed a neural network model of 11 layers in our experiments and used stochastic gradient descent for optimization. The model layers are illustrated in Figure 5.3, where we present the computational workload for each layer and demonstrate how different layers can have varying processing requirements. In each experiment, the hyperparameter configuration slightly adjusted the learning rate, which was chosen after testing a range of rates from 1e-2 to 3e-6. The data was divided into an 8:2 ratio for training and testing, and the epoch was set to one to minimize the overhead between client and server. The experiments are defined as follows:

Centralized (cen): In the Cen experiment, all client raw data is sent to a central server for aggregation and training of the machine learning model. The model was trained with a learning rate of 3e-4.

Split Learning (SL): For the SL experiments, the first six layers were trained on the client side for one epoch with a learning rate of 1e-5, while the server completed the training of the remaining layers.

Dynamic Split Federated Learning (DSFL): In the DSFL experiments, denoted as DSFL GA when utilizing the Genetic Algorithm for client selection and DSFL Random when selecting clients randomly, participants employed a dynamic layer based on the Layer Predictor. The initial four layers were trained for one epoch, and the remaining layers were trained on the server side with a learning rate of 1e-4.

Resource-aware Split Federated Learning (RSFL): The RSFL experiments followed the same training procedure as DSFL but with a fixed client-model six layer and incorporated a resource-aware client selection strategy (Wazzeh *et al.*, 2024b).

In our approach, we employed cross-entropy loss for loss calculation and implemented weighted averaging to aggregate the models' weights into a unified global model on the federated learning server. We also leveraged the Weights & Biases framework (Biewald, 2020) to visually represent the outcomes of our experiments.

5.7.3 Results and Analysis

This section presents the performance analysis of classification accuracy and model loss for the fall dataset. The figures below provide valuable insights into the effectiveness of various methods.

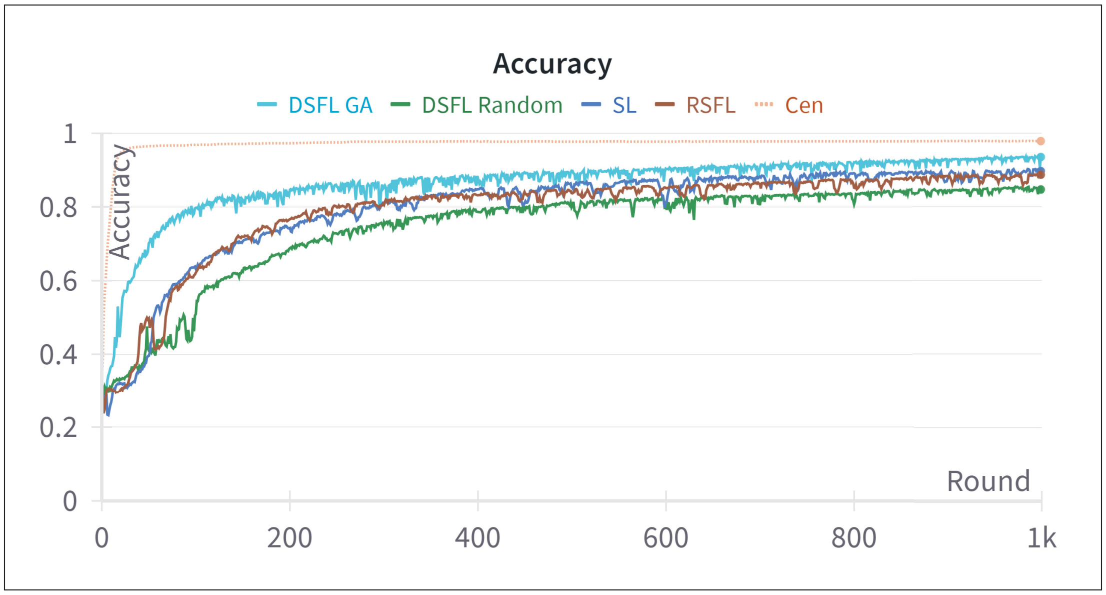

Figure 5.4 Accuracy Analysis per Communication Round

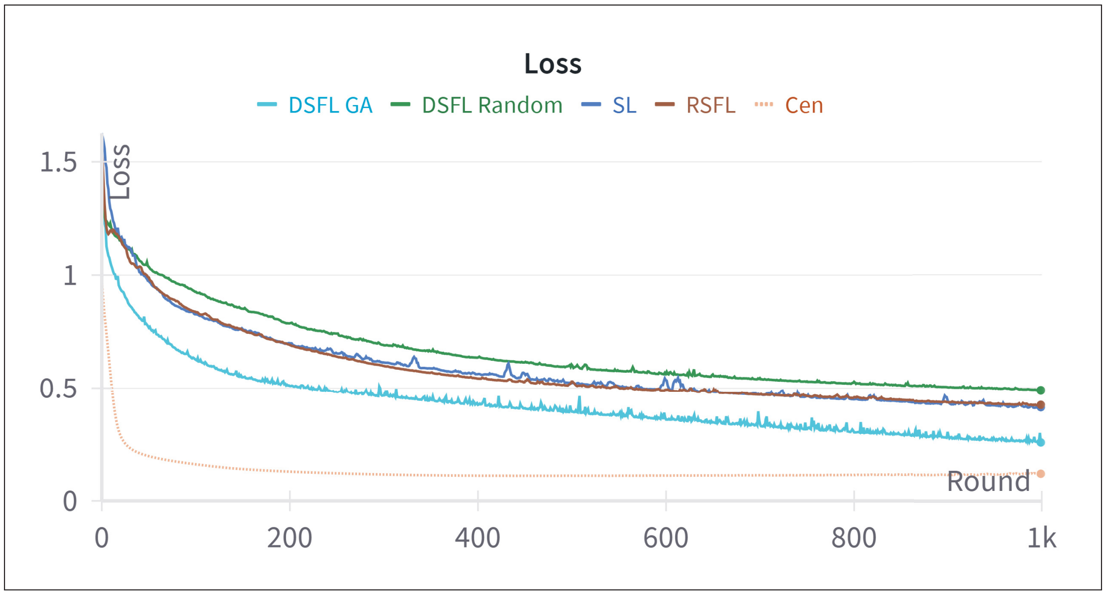

Figure 5.5 Loss Analysis per Communication Round

5.7.3.1 Performance Metrics: Accuracy and Loss

The figures 5.4 and 5.5 illustrate the accuracy and loss of different experiments over 1000 rounds, providing a comprehensive view of their performance. The analyzed methods include DSFL Random, Cen, DSFL GA, SL GA, SL Random, and RSFL.

When it comes to accuracy, Cen and DSFL GA demonstrate superior performance. The cen experiment achieves near-perfect accuracy early on. It maintains its consistency, reaching 0.98, while the DSFL GA experiment climbs to a high accuracy of around 0.95 and has stability with minimal fluctuations. SL and RSFL also perform well, steadily increasing to approximately 0.90 with relative stability. On the other hand, DSFL Random displays more variability, with lower initial accuracies and reaching a maximum of 0.85 accuracy.

Cen and DSFL GA again lead with the best results for the loss metric. Cen starts with the lowest initial loss and quickly stabilizes near 0.12, while DSFL GA drops to below 0.25 and maintains this low loss with minimal fluctuations. SL and RSFL reduce loss steadily to around 0.41, although both experience minor fluctuations. DSFL Random exhibits higher initial losses and more variability, reaching a final loss of 0.48.

Cen and DSFL GA demonstrate robust and stable performance in accuracy and loss metrics, while SL and RSFL perform effectively but with minor variability. DSFL Random shows lower accuracy and higher loss, highlighting its relative instability and less favourable performance.

5.7.3.2 Client Selection & Dropout

Figure 5.6 illustrates the number of clients dropping out due to insufficient resources for completing the training process, with the y-axis representing the number of devices that fail to finish training. Figure 5.7 represents each round's selected clients. The shadow lines represent the original values recorded per round, while the bold plots represent the time-weighted moving average.

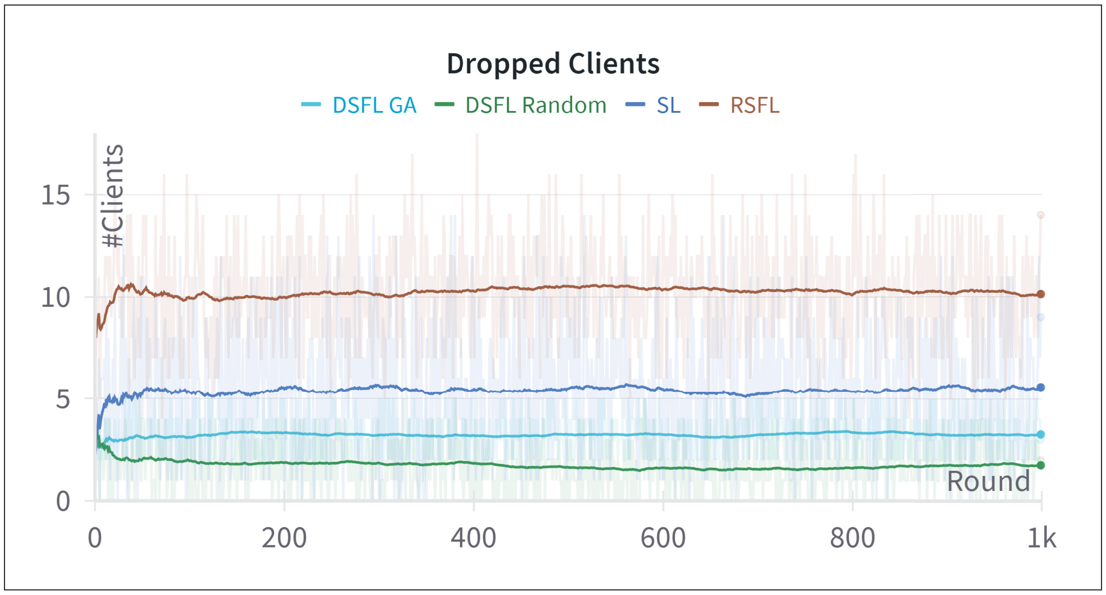

Figure 5.6 Number of Dropped Clients per Round

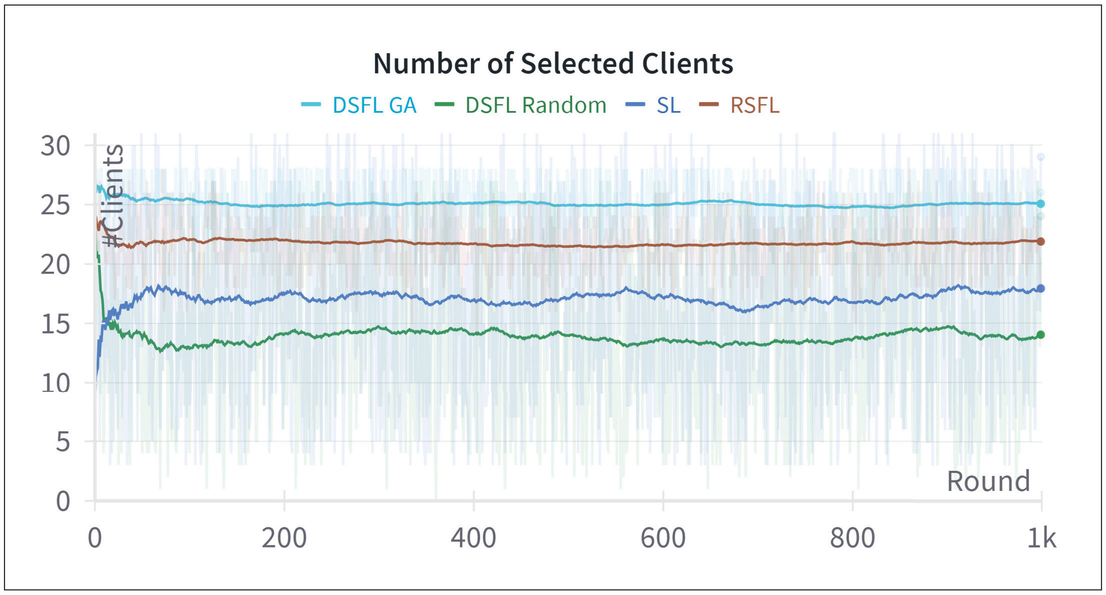

Figure 5.7 Number of Selected Clients per Round

The data indicates that DSFL Random had the lowest average dropout rate, with approximately 1.6 clients dropping out per round. Meanwhile, DSFL GA also exhibited a promising performance, with a low dropout rate, averaging 3.2 clients per round. This success can be attributed to DSFL's dynamic capability to adjust the number of clients being trained, thus meeting the resource requirements of the clients. Additionally, comparing the number of clients selected, DSFL GA showed a much higher participation rate of 28.5 clients per round than DSFL Random's 13.9 clients per round. This higher participation rate indicates the potential of DSFL GA to reduce dropout rates further in the future.

On the other hand, SL, which utilizes a sequential learning process, encountered significant challenges, leading to a higher dropout rate. The data reveals an average of 5.3 clients dropping out per round, a relatively large number. This is primarily attributed to the demanding nature of SL training, which necessitates clients to host all model layers on their side. With an average of 16.6 clients selected per round, this underscores sequential learning processes' difficulties and resource constraints.

RSFL had the highest dropout rate among all the experiments, with an average of 10.1, despite having a high average participation rate of 21.5 clients per round. This high dropout rate is likely due to clients having a fixed number of layers to train, making it challenging to accommodate changing resource availability during training. This mismatch between client capabilities and training requirements emphasizes the importance of flexible and adaptive training strategies in federated learning environments.

5.7.3.3 Traffic Analysis

The data in Figure 5.8 depicts the total client data overhead per round, measured in megabytes (MB), for various experimental configurations. The traffic is calculated by summing all uploads and downloads necessary to complete the training process for all participants in each round. The shaded regions in the plot show the reported data, while the bold lines represent the time-weighted moving average, offering a smoothed trend of the data traffic over time.

The DSFL GA experiment's average data traffic is the highest, at approximately 9 gigabytes (GB) per round. This increase is due to maximizing client selection, which results in greater data exchange during training. Conversely, the RSFL experiment shows a lower average data

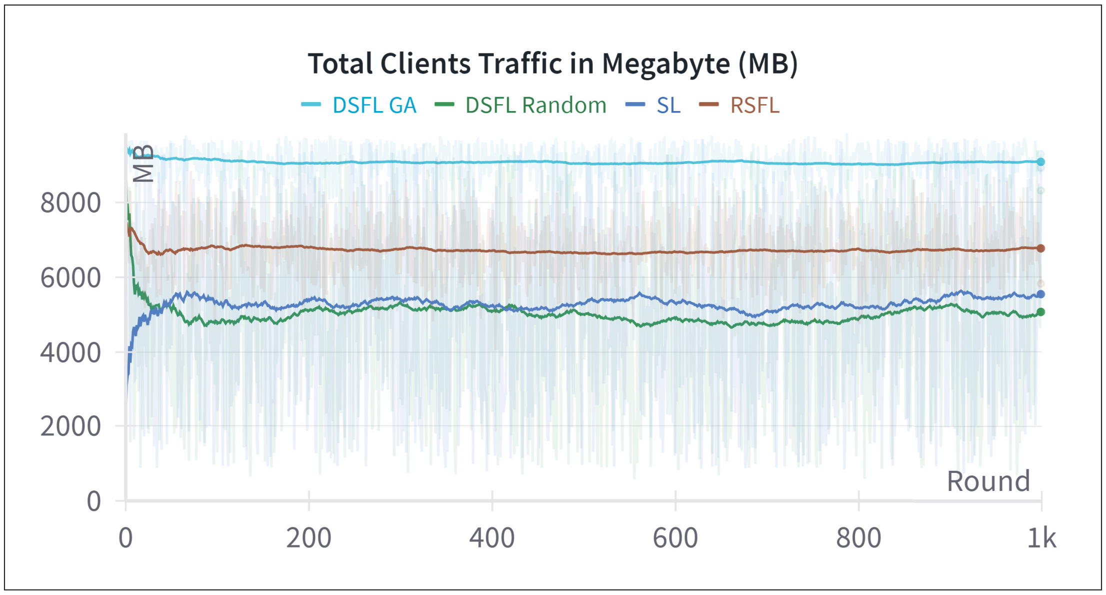

Figure 5.8 Total Clients Traffic per Round in Megabytes (MB)

traffic of around 6.6 GB per round. This reduction in traffic is attributed to a decrease in the number of selected clients, resulting from fixed model layers and resource constraints affecting client participation during training.

The SL and DSFL Random experiments demonstrate similar average data traffic, approximately 5.1 GB per round. The relatively lower traffic in these methods can be attributed to fewer client selections, as random client selection is employed. This random selection leads to significant participant variability from one round to another, with several clients potentially dropping out due to insufficient resources. Ultimately, these factors contribute to the reduced data traffic observed in the SL and DSFL Random experiments.

5.7.3.4 Idle Time

The data in Figure 5.9 illustrates the idle time of clients in natural logarithm milliseconds ln(ms), representing the variance in completion times during model training. This variation, often called the straggler effect, directly impacts the round processing time and overall convergence speed.

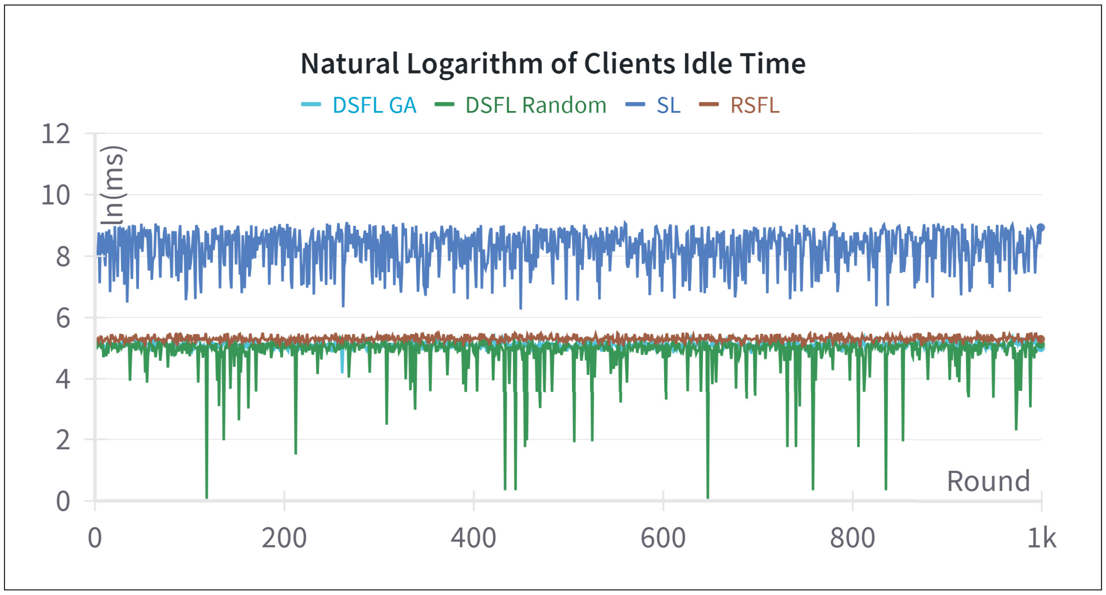

Figure 5.9 Clients Idle time

Due to the dynamic client layer mechanism and parallel training, the DSFL experiments demonstrate an average client idle time of $5.165 \approx 175$ ms. This mechanism strategically selects split points for clients to synchronize their training times, effectively reducing idle time. Similarly, the RSFL experiments show a slightly increased average client idle time of $5.808 \approx 330$ ms. However, this remains relatively low due to the highly efficient client selection and resource allocation strategies, which successfully minimize training time variance. On the other hand, the SL experiments exhibit a significantly higher average client idle time of $8.336 \approx 4400$ ms. This extended idle time results from the sequential training nature, where clients must wait for preceding clients, leading to higher overall idle time.

5.7.3.5 Model Training Time

The data presented in Figure 5.10 depicts the total training time in the natural logarithm of milliseconds ln(ms) across various experiments. In the case of DSFL Random, DSFL GA, and RSFL, both client and server training times demonstrate comparable and efficient performance, indicating well-balanced workload distribution and effective processing for each round.

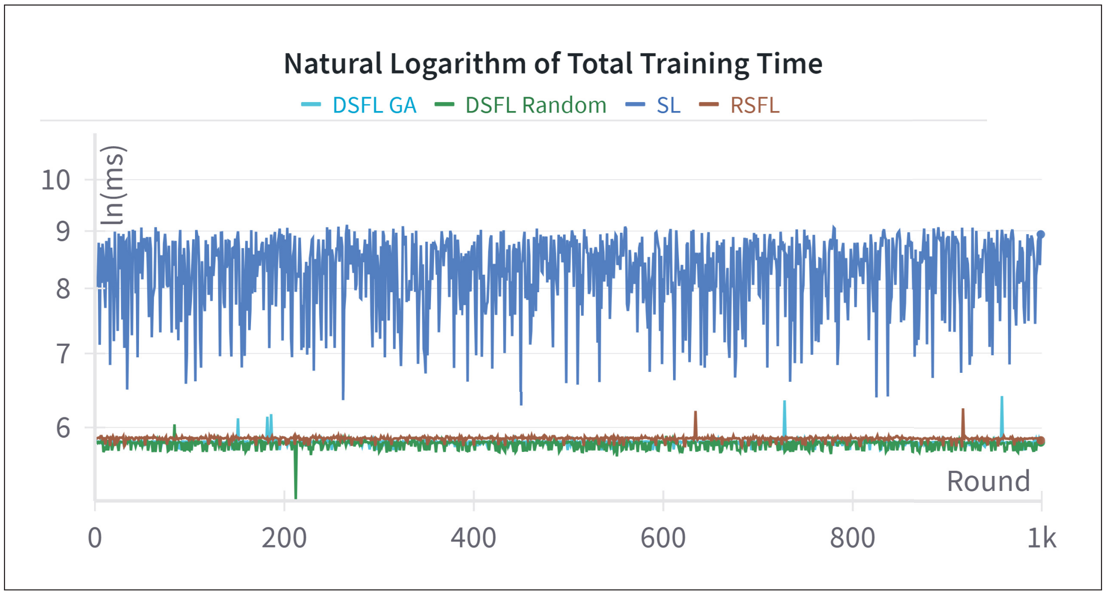

Figure 5.10 Total Training Time in Natural Logarithm Milliseconds ln(ms)

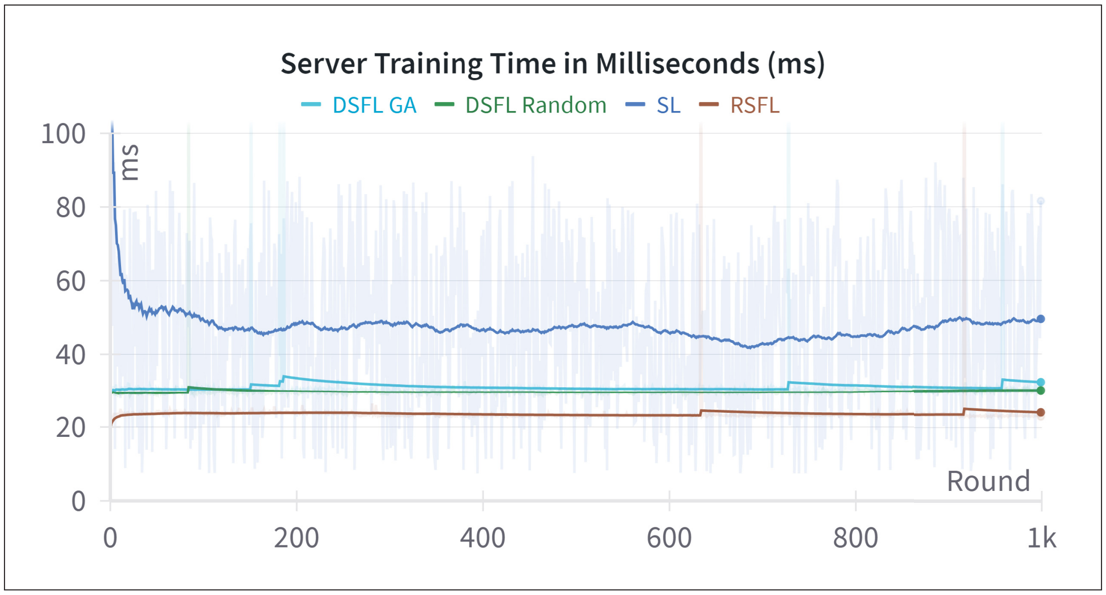

Figure 5.11 Total Server Training Time per Round in Milliseconds (ms)

DSFL Random and DSFL GA exhibit similar averages of approximately 5.808 ≈ 330 ms per round for client training time. This consistency suggests that the parallel training mechanism utilized by these methods effectively distributes the training workload among clients, resulting

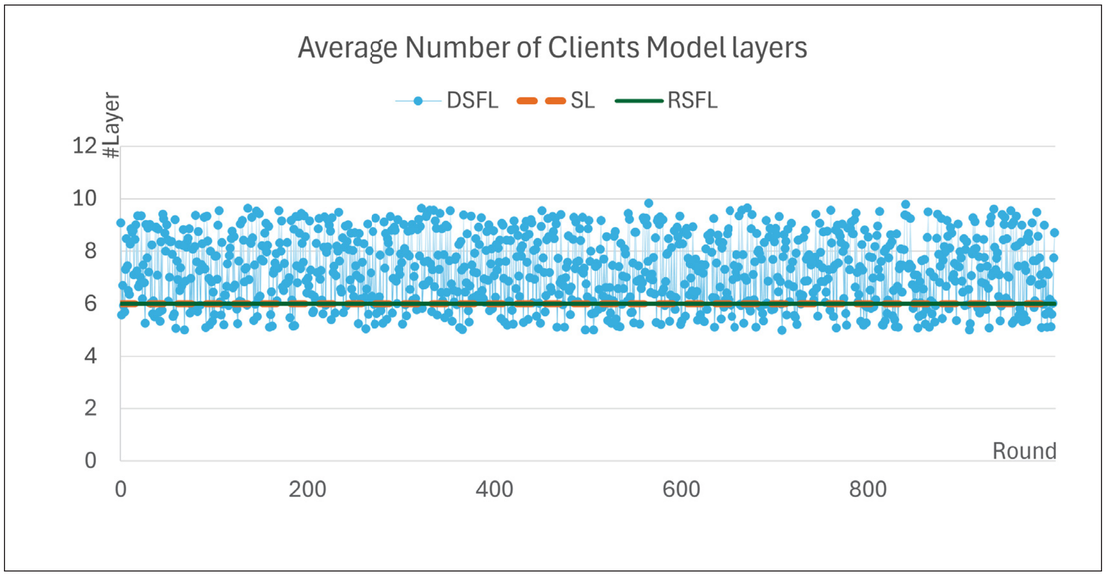

Figure 5.12 Average Number of Clients Model layers

in efficient and uniform training times. RSFL shows a slightly elevated average of 5.857 ≈ 350 ms per round for client training time. However, it maintains efficiency due to its resource-based client selection approach, which minimizes training delays and preserves overall efficiency by selecting only clients with sufficient resources. In contrast, the SL Random method demonstrates significant variability and higher average client training times, approximately 6.144 ≈ 4,650 ms per round. This substantial increase is attributed to the sequential nature of client training in SL, where each client must wait for the preceding one to complete training, resulting in prolonged and more variable training times that significantly impact overall efficiency.

Regarding server training time represented in Figure 5.11. The experiments in DSFL Random, DSFL GA, and RSFL consistently average around 30 ms per round. This uniformity underscores the efficiency of these methods in managing client and server workloads. The minimal delays on the server side suggest that it can effectively handle coordination and aggregation tasks without significant overhead. In contrast, the SL Random method shows a higher average server training time of 53 ms per round. The increased server training time in SL Random is due to the overhead of managing the sequential training process. In this method, the server must coordinate the training order and handle more frequent synchronization tasks, resulting in longer server-side processing times.

Figure 5.12 illustrates each round's average number of client layers. It shows that the DSFL experiment have a higher average number of layers for clients. This is because the Layer Predictor dynamically adjusts the number of trainable layers based on the available resources, varying between 5 and 10 layers. In contrast, the SL and RSFL approaches maintain a constant fixed number of layers, set at 6, regardless of the clients' resource availability.

The DSFL Random, DSFL GA, and RSFL methods consistently demonstrate efficient training times for clients and the server. These methods employ parallel training approaches that allow for a balanced distribution of workload, which is particularly compelling given their ability to adjust the number of client layers dynamically. This adaptability leads to optimized training times and faster convergence. In contrast, the SL Random method's sequential training approach results in significantly higher and more variable training times for clients and the server. This is due to its fixed number of layers for clients, which does not adapt to the available resources and results in less efficient training and slower convergence. The static layer configuration in SL and RSFL highlights the limitations of these approaches in scenarios where rapid convergence and effective resource utilization are critical. In contrast, the DSFL experiments demonstrate the advantages of dynamic layer configurations. By adapting the number of layers based on available resources, DSFL methods optimize training times and enhance convergence efficiency.

5.7.4 Discussion

The Dynamic Split Federated Learning approach is a robust solution, excelling in performance metrics, data traffic management, and model training time. DSFL GA consistently delivers high accuracy and low loss, showcasing its robust model convergence and stability. Its capacity to balance model performance with resource allocation, which distributes computational resources among the clients, ensures reliable and consistent outcomes.

DSFL GA has a significant advantage in efficiently managing client-side and server-side data overhead. Although specific methods have slightly higher client-side data overhead, DSFL GA's server-side data communication costs remain competitive. This compensation underscores DSFL GA's effectiveness in maintaining high performance while efficiently managing data communication, providing reassurance about its efficacy.

When compared to RSFL, DSFL GA remains competitive. While RSFL demonstrates slightly better data efficiency, DSFL GA ensures consistent training times among selected clients. This consistency enhances its efficiency and effectiveness in managing training durations across various clients.

In contrast, the DSFL Random approach reveals significant performance metrics and data overhead variability. While maintaining consistent training times, DSFL Random exhibits higher fluctuations in other performance aspects. This variability suggests that DSFL Random may be less effective in environments where stable and reliable performance is crucial, thus the need for a heuristic client selection approach.

The SL method demonstrates inefficiencies compared to other methods. SL has significantly higher average client and server training times due to its sequential training process, creating a bottleneck effect and leading to longer training times and increased processing delays. In contrast, DSFL GA, DSFL Random, and RSFL have parallel processing capabilities, offering efficient training times. DSFL GA is a strong choice for dynamic resource environments, providing high accuracy, efficient training, and manageable data overhead. Its balanced resource allocation and data communication approach make it a compelling option, especially when performance stability and efficient resource use are critical. DSFL Random, while consistent in training times, lacks the stability and efficiency of DSFL GA. The inefficiencies of the SL method underscore the limitations of sequential training approaches, highlighting the advantages of parallel processing strategies employed by DSFL GA.

5.8 Conclusion and Future Directions

Efficiently training resource-constrained IoT devices presents significant challenges due to privacy concerns and limited resources. We have developed a hybrid approach to address these challenges, optimizing client selection within a dynamic federated and split learning framework. This approach incorporates a client selection optimization algorithm based on GA objectives to select clients efficiently. Additionally, we introduced a machine learning algorithm that predicts the optimal cut layer for each client, enabling dynamic layer training and optimizing the split federated learning architecture. Our contributions include the development of a novel Dynamic Split Federated Learning (DSFL) architecture that dynamically adjusts to client resources, minimizing training overhead and improving resource utilization. We also evaluated key performance metrics to demonstrate the efficiency of our approach. We conducted comprehensive experiments, showing that our method achieves comparable accuracy to centralized approaches while significantly reducing client idle time and enhancing overall system performance. By employing advanced client selection and dynamic layer training, our approach surpasses existing benchmarks, achieving higher accuracy and reducing overhead training time compared to traditional methods.

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This thesis has addressed the critical challenges and constraints in continuous authentication for IoT and mobile systems by leveraging FL and SL frameworks. Throughout our contributions, we focused on achieving robust continuous authentication mechanisms that preserve user privacy, maintain high accuracy, and efficiently manage resources in dynamic and heterogeneous environments.

Our research objectives centered on mitigating key limitations in existing continuous authentication frameworks. To this end, we integrated FL and SL with advanced techniques such as warmup initialization, transfer learning, and resource-aware client selection. We introduced novel architectural enhancements, including the Cluster-Based Resource-aware Split Federated Learning (CRSFL) framework and the Dynamic Split Federated Learning (DSFL) approach, to handle non-IID data distributions, heterogeneity in device capabilities, and communication overhead. Our extensive evaluations demonstrated that these techniques significantly improve model convergence, reduce training overhead, enhance accuracy, and ensure stable participation from a diverse set of devices.

Specifically, the contributions of this thesis can be summarized as follows:

- **Warmup-based Federated Learning for Continuous Authentication (Chapter 2):** In this chapter, we proposed one of the first federated learning (FL) architectures specifically tailored for continuous authentication, addressing the critical challenge of non-Independent and Identically Distributed (non-IID) data. To overcome the limitations posed by data heterogeneity, we introduced a novel warmup phase at the beginning of the model training process. This warmup strategy acts as a stabilizer for the federated learning process by establishing more robust initial conditions for local models, which significantly enhances convergence rates and improves overall authentication accuracy. Unlike traditional approaches that struggle with non-IID data, our method leverages the warmup phase to boost the performance of FL, ensuring better adaptability across clients.

Extensive experiments conducted on real-world authentication datasets demonstrated the effectiveness of the proposed approach. The results showed that conventional FL algorithms, such as FedAvg, are inadequate for handling unique data distributions, while the warmup-based strategy effectively mitigates the adverse effects of data heterogeneity. This approach not only reduced variance across clients but also led to substantial improvements in model accuracy, demonstrating its practical utility for continuous authentication in real-world scenarios.

- **Cluster-Based Resource-aware Split Federated Learning (CRSFL) (Chapter 3):** In this chapter, we introduced a novel hybrid FL-SL architecture called Cluster-Based Resource-aware Split Federated Learning (CRSFL), specifically designed to operate efficiently in heterogeneous IoT environments. CRSFL intelligently forms clusters of devices based on their resource capabilities and data characteristics, enabling more effective distributed training. By grouping devices with similar computational power, CRSFL reduces network overhead and enhances the scalability of split federated learning processes. Additionally, by addressing key challenges such as resource optimization, client dropout mitigation, and efficient client selection, CRSFL provides a comprehensive solution for continuous authentication in IoT environments.

  Central to CRSFL's effectiveness was the integration of a resource-aware client selection heuristic algorithm, leveraging multi-objective optimization techniques (e.g., based on Genetic Algorithms) to determine the most suitable clients for participation in each training round. This heuristic considered multiple factors—such as device reliability, data quality, and event rates—to maximize system performance while minimizing straggler effects. By ensuring that only resource-capable and beneficial clients engaged in the model training, CRSFL maintained balanced workloads across the network, curtailed idle times, and accelerated model convergence.

Through extensive experimentation, we demonstrated that CRSFL significantly reduced communication overhead and training latency, achieving better accuracy-latency trade-offs compared to traditional FL and SL approaches. By allocating resources and employing an advanced client selection mechanism, CRSFL enabled the efficient utilization of heterogeneous IoT devices, ultimately improving the robustness, scalability, and overall quality of continuous authentication in resource-constrained settings.

- **Continuous Authentication for Mobile Crowdsourcing with Federated Learning (Chapter 4):** In this chapter, we pushed the boundaries of continuous authentication by applying FL-based approaches to Mobile Crowdsourcing (MCS) scenarios, incorporating a large-scale facial dataset to authenticate users based on their facial biometrics. This integration allowed us to evaluate the system's performance under realistic and dynamic conditions, where the global model had to adapt to a wide range of user profiles, device types, and environmental factors.

  To further enhance model robustness and efficiency, we introduced transfer learning techniques, leveraging pre-trained feature extractors and domain-adaptive approaches that could rapidly align client models with the global model parameters. These transfer learning strategies reduced convergence time and improved accuracy, even when dealing with skewed and non-IID data distributions. The resulting FL framework maintained strong privacy guarantees through decentralized data processing and secure model aggregation, ensuring that sensitive facial information remained on the client devices.

  Our results demonstrated that, when properly configured, FL combined with transfer learning could effectively support continuous authentication in large-scale, real-world MCS environments. The global model achieved robust performance and stable authentication accuracy, despite the evolving and complex data landscapes inherent to facial data. This work validated the feasibility and benefits of employing FL and transfer learning to achieve high-quality, privacy-preserving continuous authentication in resource-constrained, heterogeneous IoT ecosystems.

- **Dynamic Split Federated Learning (DSFL) for Resource-Constrained IoT Systems (Chapter 5):** In this chapter, we introduced Dynamic Split Federated Learning (DSFL), a novel FL-SL hybrid framework designed to adapt model partitioning dynamically in response to clients' current resource conditions. Rather than relying on a fixed split point, DSFL continuously determines the extent of local layer training each client undertakes, taking into account their real-time computational and communication capabilities. This adaptability enables devices with constrained resources to contribute effectively, mitigating the risks of straggling clients and ensuring that training progresses smoothly across a diverse network of IoT devices.

  A key element of DSFL is its heuristic-driven client selection mechanism, which utilizes multi-objective optimization strategies and approaches to evaluate and choose participants best suited for each training round. By considering factors like device reliability, past performance, and predicted resource availability, DSFL achieves a well-balanced workload distribution and timely model updates. This careful alignment of resource allocations and training responsibilities not only enhances convergence speed but also improves the stability and fairness of the entire training process.

  Our experimental results demonstrate that DSFL effectively accommodates heterogeneous device landscapes and dynamic conditions. By continually adjusting model partitioning and leveraging heuristic client selection, DSFL realizes more robust, scalable, and efficient continuous authentication frameworks that maintain high performance even in resource-constrained and rapidly evolving IoT environments.

Collectively, these contributions establish a solid foundation for continuous authentication solutions in real-world IoT and MCS scenarios. By maintaining user data locality, adapting to resource variability, and leveraging advanced optimization and privacy-preserving techniques, we have demonstrated meaningful progress towards operational, robust, and secure split FL solutions for continuous authentication systems.

6.2 Recommendations for Future Work

While our research has advanced the state of the art in continuous authentication via FL and SL, several directions remain open for further investigation and refinement:

- **Enhanced Privacy and Security Mechanisms:** Although privacy-preserving techniques—such as differential privacy, secure aggregation, and homomorphic encryption—have been explored in FL settings (AbdulRahman *et al.*, 2020b; Wahab *et al.*, 2021; Truong *et al.*, 2021), more adaptive protocols tailored specifically to continuous authentication are needed. Future work might involve developing privacy solutions that combine cryptographic techniques and model obfuscation to counteract sophisticated inference or backdoor attacks. Recent efforts have begun to examine privacy-preserving continuous authentication approaches (Wazzeh *et al.*, 2022a; Feng *et al.*, 2024), but further innovations are required to strengthen guarantees and ensure minimal impact on model accuracy and latency.
- **Hierarchical and Multi-Tier Architectures:** Emerging hierarchical FL architectures, potentially combined with SL, can improve responsiveness and scalability in large-scale IoT deployments (Duan *et al.*, 2022; Wu *et al.*, 2023). Incorporating intermediate-edge servers and cluster-level aggregations could reduce communication overhead and latency, adapt to fluctuating resource conditions, and enable efficient load-balancing strategies. Investigating hierarchical optimization frameworks or modular designs (Arafeh *et al.*, 2023a) that dynamically allocate tasks across multiple tiers of devices is a promising avenue for future research.
- **Adaptive and Autonomous Management:** As IoT devices vary widely in capabilities and operate under dynamic conditions, autonomous and context-aware decision-making is

essential. Reinforcement learning or multi-agent systems could be leveraged to optimize client selection, model partitioning, and aggregation strategies in real time, building on existing client selection heuristics and multi-criteria approaches (AbdulRahman *et al.*, 2020a; Chahoud *et al.*, 2023). Such adaptive mechanisms can ensure that continuous authentication frameworks maintain robustness and efficiency despite evolving device profiles, changing network states, and heterogeneous data distributions.

- **Integration with Emerging IoT Paradigms:** Beyond conventional IoT and MCS scenarios, continuous authentication solutions can be integrated into diverse verticals, including drone-based systems, autonomous vehicles, and Industry 4.0 applications (Yazdinejad *et al.*, 2021; Sami & Mourad, 2020). Future research may consider novel sensing modalities, interaction patterns, and mobility models to refine authentication strategies that handle even more complex environments. Investigating the interplay with 5G-enabled networks, metaverse applications (Sami *et al.*, 2023), and resource-intensive distributed systems can further broaden the applicability and resilience of FL/SL-based authentication solutions.
- **Real-World Deployments and Standards:** Finally, translating these advancements into large-scale, real-world deployments remains a crucial step. Ensuring interoperability, defining standardized benchmarks, and establishing common datasets or testbeds (Caldas *et al.*, 2018) would enable fair comparisons and facilitate industry adoption. Collaborations between academia and industry stakeholders can help identify practical constraints, guide the development of user-friendly frameworks, and validate proposed solutions under realistic conditions. Such efforts would accelerate the transition from research prototypes to fully operational split federated continuous authentication systems.

In conclusion, this thesis has laid a robust conceptual and practical foundation for FL and SL-based continuous authentication in resource-constrained, privacy-sensitive IoT and MCS domains. By building upon our findings, future researchers and practitioners can further refine, adapt, and extend these solutions, ultimately shaping a new era of secure, scalable, and user-centric distributed learning systems.

LIST OF REFERENCES

AbdulRahman, S., Tout, H., Mourad, A. & Talhi, C. (2020a). FedMCCS: Multicriteria client selection model for optimal IoT federated learning. *IEEE Internet of Things Journal*, 8(6), 4723–4735.

AbdulRahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C. & Guizani, M. (2020b). A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7), 5476–5497.

Abuhamad, M., Abuhmed, T., Mohaisen, D. & Nyang, D. (2020). AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors. *IEEE Internet of Things Journal*, 7(6), 5008–5020.

Arafeh, M., Otrok, H., Ould-Slimane, H., Mourad, A., Talhi, C. & Damiani, E. (2023a). ModularFed: Leveraging modularity in federated learning frameworks. *Internet of Things*, 100694.

Arafeh, M., Wazzeh, M., Ould-Slimane, H., Talhi, C., Mourad, A. & Otrok, H. (2023b). Efficient privacy-preserving ML for IoT: Cluster-based split federated learning scheme for non-IID data. *2023 7th Cyber Security in Networking Conference (CSNet)*, pp. 143–147.

Arias, O., Wurm, J., Hoang, K. & Jin, Y. (2015). Privacy and security in internet of things and wearable devices. *IEEE transactions on multi-scale computing systems*, 1(2), 99–109.

Bansal, R., Raj, G. & Choudhury, T. (2016). Blur image detection using Laplacian operator and Open-CV. *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, pp. 63–67.

Bazgan, C., Hugot, H. & Vanderpooten, D. (2009). Solving efficiently the 0–1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1), 260–279.

Bharathkumar, K., Paolini, C. & Sarkar, M. (2020). FPGA-based edge inferencing for fall detection. *2020 IEEE Global Humanitarian Technology Conference (GHTC)*, pp. 1–8.

Biewald, L. [Software available from wandb.com]. (2020). Experiment Tracking with Weights and Biases. Retrieved from: https://www.wandb.com/.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V. & Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M. & Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pp. 67–74.

Chahoud, M., Sami, H., Mourad, A., Otoum, S., Otrok, H., Bentahar, J. & Guizani, M. (2023). On-demand-fl: A dynamic and efficient multi-criteria federated learning client deployment scheme. *IEEE Internet of Things Journal*.

Chelli, A. & Pätzold, M. (2019). A machine learning approach for fall detection and daily living activity recognition. *IEEE Access*, 7, 38670–38687.

Ching, K. W. & Singh, M. M. (2016). Wearable technology devices security and privacy vulnerability analysis. *International Journal of Network Security & Its Applications*, 8(3), 19–30.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6), 141–142.

Duan, Q., Hu, S., Deng, R. & Lu, Z. (2022). Combined federated and split learning in edge computing for ubiquitous intelligence in internet of things: State-of-the-art and future directions. *Sensors*, 22(16), 5983.

Elayan, H., Aloqaily, M. & Guizani, M. (2021). Sustainability of healthcare data analysis IoT-based systems using deep federated learning. *IEEE Internet of Things Journal*, 9(10), 7338–7346.

Feng, W., Yan, Z., Zhang, H., Zeng, K., Xiao, Y. & Hou, Y. T. (2017). A survey on security, privacy, and trust in mobile crowdsourcing. *IEEE Internet of Things Journal*, 5(4), 2971–2992.

Feng, X., Wang, X., Liu, H., Yang, H. & Wang, L. (2024). A Privacy-preserving Aggregation Scheme with Continuous Authentication for Federated Learning in VANETs. *IEEE Transactions on Vehicular Technology*.

Gao, Y., Kim, M., Abuadbba, S., Kim, Y., Thapa, C., Kim, K., Camtepe, S. A., Kim, H. & Nepal, S. (2020). End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*.

Gao, Y., Kim, M., Thapa, C., Abuadbba, A., Zhang, Z., Camtepe, S., Kim, H. & Nepal, S. (2021). Evaluation and optimization of distributed machine learning techniques for internet of things. *IEEE Transactions on Computers*, 71(10), 2538–2552.

Gascon, H., Uellenbeck, S., Wolf, C. & Rieck, K. (2014). Continuous authentication on mobile devices by analysis of typing motion behavior. *Sicherheit 2014–Sicherheit, Schutz und Zuverlässigkeit*.

Gupta, O. & Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116, 1–8.

Hammoud, A., Otrok, H., Mourad, A., Wahab, O. A. & Bentahar, J. (2018). On the detection of passive malicious providers in cloud federations. *IEEE Communications Letters*, 23(1), 64–67.

Han, D.-J., Bhatti, H. I., Lee, J. & Moon, J. (2021). Accelerating federated learning with split learning on locally generated losses. *ICML 2021 workshop on federated learning for user privacy and data confidentiality. ICML Board*.

He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M. & Avestimehr, S. (2020). FedML: A Research Library and Benchmark for Federated Machine Learning. *arXiv preprint arXiv:2007.13518*.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, L., Ma, C., Tu, C. & Zhang, Y. (2022). Gait2Vec: Continuous Authentication of Smartphone Users Based on Gait Behavior. *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 280–285.

Hemmatpour, M., Ferrero, R., Montrucchio, B. & Rebaudengo, M. (2019). A review on fall prediction and prevention system for personal devices: evaluation and experimental results. *Advances in human-computer interaction*, 2019.

Hosseini, H., Park, H., Yun, S., Louizos, C., Soriaga, J. & Welling, M. (2021). Federated learning of user verification models without sharing embeddings. *International Conference on Machine Learning*, pp. 4328–4336.

Hussain, F., Umair, M. B., Ehatisham-ul Haq, M., Pires, I. M., Valente, T., Garcia, N. M. & Pombo, N. (2019). An efficient machine learning-based elderly fall detection algorithm. *arXiv preprint arXiv:1911.11976*.

Islam, M. M., Tayan, O., Islam, M. R., Islam, M. S., Nooruddin, S., Kabir, M. N. & Islam, M. R. (2020). Deep learning based systems developed for fall detection: a review. *IEEE Access*, 8, 166117–166137.

Jebbaoui, H., Mourad, A., Otrok, H. & Haraty, R. (2015). Semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies. *Computers & Electrical Engineering*, 44, 91–103.

Jorquera Valero, J. M., Sanchez Sanchez, P. M., Fernández Maimó, L., Huertas Celdran, A., Arjona Fernandez, M., De Los Santos Vílchez, S. & Martínez Pérez, G. (2018). Improving the security and QoE in mobile devices through an intelligent and adaptive continuous authentication system. *Sensors*, 18(11), 3769.

Konak, A., Coit, D. W. & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91(9), 992–1007.

Kong, H., Lu, L., Yu, J., Chen, Y. & Tang, F. (2020). Continuous authentication through finger gesture interaction for smart homes using WiFi. *IEEE Transactions on Mobile Computing*, 20(11), 3148–3162.

Krizhevsky, A., Nair, V. & Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5).

Lee, W.-H. & Lee, R. B. (2017). Implicit smartphone user authentication with sensors and contextual machine learning. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 297–308.

Li, A., Sun, J., Zeng, X., Zhang, M., Li, H. & Chen, Y. (2021a). Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 42–55.

Li, J., Chen, T. & Teng, S. (2024). A comprehensive survey on client selection strategies in federated learning. *Computer Networks*, 110663.

Li, Q., Diao, Y., Chen, Q. & He, B. (2021b). Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*.

Li, T., Xia, T., Wang, H., Tu, Z., Tarkoma, S., Han, Z. & Hui, P. (2022). Smartphone app usage analysis: Datasets, methods, and applications. *IEEE Communications Surveys & Tutorials*, 24(2), 937–966.

Li, Y. & Lyu, X. (2023). Convergence Analysis of Split Learning on Non-IID Data. *arXiv preprint arXiv:2302.01633*.

Liang, Y., Samtani, S., Guo, B. & Yu, Z. (2020). Behavioral biometrics for continuous authentication in the internet-of-things era: An artificial intelligence perspective. *IEEE Internet of Things Journal*, 7(9), 9128–9143.

Liu, W., Wang, X. & Peng, W. (2019). Secure remote multi-factor authentication scheme based on chaotic map zero-knowledge proof for crowdsourcing internet of things. *IEEE Access*, 8, 8754–8767.

Lu, C. X., Du, B., Zhao, P., Wen, H., Shen, Y., Markham, A. & Trigoni, N. (2018). Deepauth: in-situ authentication for smartwatches via deeply learned behavioural biometrics. *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pp. 204–207.

Lust, T. & Teghem, J. (2012). The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4), 495–520.

Mahbub, U., Sarkar, S., Patel, V. M. & Chellappa, R. (2016a, Sept). Active user authentication for smartphones: A challenge data set and benchmark results. *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1-8. doi: 10.1109/BTAS.2016.7791155.

Mahbub, U., Sarkar, S., Patel, V. M. & Chellappa, R. (2016b). Active user authentication for smartphones: A challenge data set and benchmark results. *2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS)*, pp. 1–8.

McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. (2017a). Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*, pp. 1273–1282.

McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. (2017b). Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*, pp. 1273–1282.

Monschein, D., Pérez, J. A. P., Piotrowski, T., Nochta, Z., Waldhorst, O. P. & Zirpins, C. (2021). Towards a peer-to-peer federated machine learning environment for continuous authentication. *2021 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6.

Mourad, A. & Jebbaoui, H. (2015). SBA-XACML: Set-based approach providing efficient policy decision process for accessing Web services. *Expert systems with applications*, 42(1), 165–178.

Mourad, A., Tout, H., Wahab, O. A., Otrok, H. & Dbouk, T. (2020). Ad hoc vehicular fog enabling cooperative low-latency intrusion detection. *IEEE Internet of Things Journal*, 8(2), 829–843.

Murata, T., Ishibuchi, H. et al. (1995). MOGA: multi-objective genetic algorithms. *IEEE international conference on evolutionary computation*, 1, 289–294.

Novák, V., Perfilieva, I., Holčapek, M. & Kreinovich, V. (2014). Filtering out high frequencies in time series using F-transform. *Information Sciences*, 274, 192–209.

Ojetola, O., Gaura, E. & Brusey, J. (2015). Data set for fall events and daily activities from inertial sensors. *Proceedings of the 6th ACM multimedia systems conference*, pp. 243–248.

Oza, P. & Patel, V. M. (2021a). Federated Learning-based Active Authentication on Mobile Devices. *arXiv preprint arXiv:2104.07158*.

Oza, P. & Patel, V. M. (2021b). Federated learning-based active authentication on mobile devices. *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–8.

Pardalos, P. M., Steponavičė, I. & Žilinskas, A. (2012). Pareto set approximation by the method of adjustable weights and successive lexicographic goal programming. *Optimization Letters*, 6, 665–678.

Pritee, Z. T., Anik, M. H., Alam, S. B., Jim, J. R., Kabir, M. M. & Mridha, M. (2024). Machine learning and deep learning for user authentication and authorization in cybersecurity: A state-of-the-art review. *Computers & Security*, 103747.

Rahman, S. A., Tout, H., Talhi, C. & Mourad, A. (2020). Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6), 310–317.

Reiss, C., Wilkes, J. & Hellerstein, J. L. (2011). *Google cluster-usage traces: format + schema*. Mountain View, CA, USA.

Salazar, A., Safont, G., Vergara, L. & Vidal, E. (2023a). Graph Regularization Methods in Soft Detector Fusion. *IEEE Access*.

Salazar, A., Vergara, L. & Vidal, E. (2023b). A proxy learning curve for the Bayes classifier. *Pattern Recognition*, 136, 109240.

Samet, S., Ishraque, M. T., Ghadamyari, M., Kakadiya, K., Mistry, Y. & Nakkabi, Y. (2019). TouchMetric: a machine learning based continuous authentication feature testing mobile application. *International Journal of Information Technology*, 11(4), 625–631.

Sami, H. & Mourad, A. (2020). Dynamic on-demand fog formation offering on-the-fly IoT service deployment. *IEEE Transactions on Network and Service Management*, 17(2), 1026–1039.

Sami, H., Hammoud, A., Arafeh, M., Wazzeh, M., Arisdakessian, S., Chahoud, M., Wehbi, O., Ajaj, M., Mourad, A., Otrok, H. et al. (2023). The Metaverse: Survey, Trends, Novel Pipeline Ecosystem & Future Directions. *arXiv preprint arXiv:2304.09240*.

Samikwa, E., Di Maio, A. & Braun, T. (2022). Ares: Adaptive resource-aware split learning for internet of things. *Computer Networks*, 218, 109380.

Samikwa, E., Di Maio, A. & Braun, T. (2024). DFL: Dynamic Federated Split Learning in Heterogeneous IoT. *IEEE transactions on machine learning in communications and networking*.

Sánchez, P. M. S., Maimó, L. F., Celdrán, A. H. & Pérez, G. M. (2021). AuthCODE: A privacy-preserving and multi-device continuous authentication architecture based on machine and deep learning. *Computers & Security*, 103, 102168.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression.

Sinaga, K. P. & Yang, M.-S. (2020). Unsupervised K-means clustering algorithm. *IEEE access*, 8, 80716–80727.

Singh, A., Dhanaraj, R. K. & Sharma, A. K. (2024). Personalized device authentication scheme using Q-learning-based decision-making with the aid of transfer fuzzy learning for IIoT devices in zero trust network (PDA-QLTFL). *Computers and Electrical Engineering*, 118, 109435.

Siraj, M. S. & et al. (2022). Incentives to learn: A location-based federated learning model. *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 40–45.

Sivaram, M., Rathee, G., Rastogi, R., Quasim, M. T. & Saini, H. (2020). A resilient and secure two-stage ITA and blockchain mechanism in mobile crowd sourcing. *Journal of Ambient Intelligence and Humanized Computing*, 11, 5003–5016.

Sun, L., Cao, B., Wang, J., Srisa-an, W., Philip, S. Y., Leow, A. D. & Checkoway, S. (2020). Kollector: Detecting fraudulent activities on mobile devices using deep learning. *IEEE Transactions on Mobile Computing*, 20(4), 1465–1476.

Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-first AAAI conference on artificial intelligence*.

Thapa, C., Arachchige, P. C. M., Camtepe, S. & Sun, L. (2022). Splitfed: When federated learning meets split learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36, 8485–8493.

Truong, N., Sun, K., Wang, S., Guitton, F. & Guo, Y. (2021). Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Computers & Security*, 110, 102402.

Tu, L., Ouyang, X., Zhou, J., He, Y. & Xing, G. (2021). Feddl: Federated learning via dynamic layer sharing for human activity recognition. *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 15–28.

Valero, J. M. J., Sánchez, P. M. S., Celdran, A. H. & Pérez, G. M. (2020). Machine Learning as an Enabler of Continuous and Adaptive Authentication in Multimedia Mobile Devices. In *Handbook of Research on Multimedia Cyber Security* (pp. 21–47). IGI Global.

Vepakomma, P., Gupta, O., Swedish, T. & Raskar, R. (2018). Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Wahab, O. A., Mourad, A., Otrok, H. & Taleb, T. (2021). Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. *IEEE Communications Surveys & Tutorials*, 23(2), 1342–1397.

Wang, W., Wang, Y., Huang, Y., Mu, C., Sun, Z., Tong, X. & Cai, Z. (2022). Privacy protection federated learning system based on blockchain and edge computing in mobile crowdsourcing. *Computer Networks*, 215, 109206.

Wazzeh, M., Ould-Slimane, H., Talhi, C., Mourad, A. & Guizani, M. (2022a). Privacy-preserving continuous authentication for mobile and iot systems using warmup-based federated learning. *IEEE Network*.

Wazzeh, M., Ould-Slimane, H., Talhi, C., Mourad, A. & Guizani, M. (2022b). Warmup and Transfer Knowledge-Based Federated Learning Approach for IoT Continuous Authentication. *arXiv preprint arXiv:2211.05662*.

Wazzeh, M., Arafeh, M., Ould-Slimane, H., Talhi, C., Mourad, A. & Otrok, H. (2023). Towards cluster-based split federated learning approach for continuous user authentication. *2023 7th Cyber Security in Networking Conference (CSNet)*, pp. 114–118.

Wazzeh, M., Arafeh, M., Sami, H., Ould-Slimane, H., Talhi, C., Mourad, A. & Otrok, H. (2024a). CRSFL: Cluster-based Resource-aware Split Federated Learning for Continuous Authentication. *Journal of Network and Computer Applications*, 231, 103987. doi: https://doi.org/10.1016/j.jnca.2024.103987.

Wazzeh, M., Hammoud, A., Guizani, M., Mourad, A., Otrok, H., Talhi, C., Dziong, Z. & Chang-Dong, W. (2024b). 'Resource-aware Split Federated Learning for Fall Detection in the Metaverse. *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).*

Weiss, K., Khoshgoftaar, T. M. & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1), 1–40.

Wu, W., Li, M., Qu, K., Zhou, C., Shen, X., Zhuang, W., Li, X. & Shi, W. (2023). Split learning over wireless networks: Parallel design and resource management. *IEEE Journal on Selected Areas in Communications*.

Yang, K., Zhang, K., Ren, J. & Shen, X. (2015). Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE communications magazine*, 53(8), 75–81.

Yang, M., Wang, X., Qian, H., Zhu, Y., Zhu, H., Guizani, M. & Chang, V. (2022). An improved federated learning algorithm for privacy preserving in cybertwin-driven 6G system. *IEEE Transactions on Industrial Informatics*, 18(10), 6733–6742.

Yazdinejad, A., Parizi, R. M., Dehghantanha, A. & Karimipour, H. (2021). Federated learning for drone authentication. *Ad Hoc Networks*, 120, 102574.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. & Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhu, G., Deng, Y., Chen, X., Zhang, H., Fang, Y. & Wong, T. F. (2024). ESFL: Efficient Split Federated Learning over Resource-Constrained Heterogeneous Wireless Devices. *IEEE Internet of Things Journal*.

Zhu, T., Qu, Z., Xu, H., Zhang, J., Shao, Z., Chen, Y., Prabhakar, S. & Yang, J. (2019). RiskCog: Unobtrusive real-time user authentication on mobile devices in the wild. *IEEE Transactions on Mobile Computing*, 19(2), 466–483.