

Privacy-Preserving Fall Detection using Federated Machine Learning in IoT-based Applications

AXEL SVEGERUD & BJÖRN DAHLSTRÖM

MASTER'S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY



Privacy-Preserving Fall Detection using Federated Machine Learning in IoT-based Applications

Axel Svegerud
ax5122sv-s@student.lu.se

Björn Dahlström
bj3013da-s@student.lu.se

Department of Electrical and Information Technology
Faculty of Engineering | LTH | Lund University

Supervisor: Kaan Bür
Company Supervisors: Peter Bärmann & Fredrik Westman

Examiner: Maria Kihl

April 23, 2025



© 2025 Axel Svegerud & Björn Dahlström
Printed in Sweden
Tryckeriet i E-huset, Lund

Acknowledgements

We are deeply grateful to Sensative and RISE for their support throughout this project. In particular, we would like to thank our supervisors at Sensative, Peter Bärmann and Fredrik Westman, as well as our colleagues at Sensative, for their valuable insights, continuous encouragement, and engaging discussions. Their expertise and willingness to share their knowledge have been instrumental in shaping this work. Additionally, we extend our appreciation to RISE for their technical expertise, constructive feedback, and invaluable support, all of which have significantly influenced the direction and outcomes of this thesis.

Finally, we would like to express our sincere gratitude to our supervisor at LTH, Kaan Bür, for his guidance and support throughout this process. His feedback and expertise have been essential in refining and strengthening our work.

Declaration

We hereby affirm that this Master thesis was composed by ourselves, that the work herein is our own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified; nor has it been published. Where other people's work has been used (either from a printed source, internet, or any other source), this has been carefully acknowledged and referenced.

During the preparation of this thesis, we have used Writefull and ChatGPT-4 to assist us in the writing process to improve language, flow, and readability. After using this tool/service, we have reviewed and edited the content as needed and we take full responsibility for the content of the whole thesis.

Abstract

Falls are one of the leading causes of injury and death among the elderly, creating a growing need for efficient and privacy-preserving fall detection systems. Traditional machine learning approaches for fall detection often rely on centralized data collection, where sensitive user data is collected and stored on a central server. This poses significant privacy risks, especially in the healthcare sector.

This work explores the possibility of using federated machine learning for fall detection as an alternative to traditional centralized machine learning. Federated learning enables model training directly on user devices, such as smartphones and wearable sensors, without the need to transfer raw data to a central server. The thesis compares the performance and privacy protection of centralized and federated learning through experiments based on the publicly available MobiAct dataset.

The results show that federated machine learning can reduce privacy risks while maintaining competitive accuracy compared to a centralized model. Furthermore, potential challenges and opportunities of integrating federated learning into Internet-of-Things-based systems are discussed. The study thus contributes insights into how federated machine learning can be used to create more secure and privacy-preserving solutions in this area.

Keywords: Fall Detection, Federated Learning, Machine Learning, Data Privacy, Privacy Preservation, IoT, Sensor Data

Popular Science Summary

Privacy-Preserving Fall Detection using Federated Machine Learning in IoT-based Applications

Axel Svegerud & Björn Dahlström

Fall incidents are one of the most common causes of injuries and fatalities among the elderly. Modern technology can help detect falls, but this often comes at the expense of personal privacy. In our thesis, we explore how federated machine learning can be used to detect falls without sharing any raw motion data from users' devices, thereby protecting personal privacy.

Each year, millions of elderly individuals suffer from fall incidents, leading to extensive hospital stays and, in the worst cases, fatal outcomes. Modern technologies can assist in detecting and responding to fall incidents more quickly, but many of these systems are based on centralized machine learning, where all data is collected and analyzed on a central server. This poses a significant privacy risk, as sensitive health data can be exposed to leaks or misuse.

In our thesis, we investigate how the federated machine learning approach can provide a more secure alternative for fall detection. Federated learning allows the ML model to be trained directly on the users' own devices, such as smartphones and wearable sensors, without requiring any sensitive motion data to be sent to a central server. Instead, only model updates are sent and aggregated to improve the system's ability to detect falls.

To assess how this method performs in practice, we conducted an experimental study using the publicly available MobiAct dataset, which contains mo-

tion data from both daily activities and falls. We compared the performance of a federated model with that of a traditional centralized model and found that the federated solution offers nearly the same level of accuracy while significantly reducing privacy risks.

The results of our study show that federated machine learning can be an effective way to protect user privacy without compromising model performance in fall detection. This insight is crucial for the future development of health technology, particularly in elderly care and personal health monitoring where data often can be seen as very privacy sensitive. The next step in research could be to test the model in real-world environments and explore how it can be integrated into existing IoT-based healthcare platforms.

By combining advanced technology with secure data management, we can create the next generation of fall detection systems that are both effective and safe for its users.

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Project Aim | 1 |
| 1.1.1 | Research Questions | 2 |
| 1.2 | Project Limitations | 2 |
| 1.3 | Thesis Outline | 3 |
| 2 | Theoretical Background and Related Work | 5 |
| 2.1 | Data Privacy in IoT | 5 |
| 2.2 | Machine Learning | 6 |
| 2.2.1 | Learning Methods | 6 |
| 2.2.2 | ML Architectures: Centralized and Federated Learning | 7 |
| 2.3 | Related Work | 10 |
| 3 | Research and Evaluation Approach | 13 |
| 3.1 | Research Approach | 13 |
| 3.2 | Research Evaluation Strategy | 14 |
| 3.2.1 | Performance Evaluation | 14 |
| 3.2.2 | Privacy Evaluation | 15 |
| 3.3 | Dataset | 16 |
| 3.3.1 | Public Datasets | 16 |
| 3.3.2 | The MobiAct Dataset | 18 |
| 3.3.3 | Limitations with Public Datasets | 19 |
| 3.4 | Tools | 20 |
| 4 | Implementation | 21 |
| 4.1 | Dataset Selection | 21 |
| 4.2 | Framework Selection | 23 |
| 4.3 | Data Preprocessing | 25 |
| 4.4 | Model Architecture | 29 |
| 4.5 | Centralized Implementation | 30 |
| 4.6 | Federated Implementation | 31 |
| 5 | Results | 33 |
| 5.1 | Performance Evaluation | 33 |

| | | |
|----------|--|-----------|
| 5.1.1 | Performance of the Centralized Implementation | 33 |
| 5.1.2 | Performance of the Federated Implementation | 36 |
| 5.1.3 | Performance Comparison | 40 |
| 5.2 | Privacy Evaluation | 41 |
| 5.2.1 | Privacy Evaluation of the Centralized Implementation | 41 |
| 5.2.2 | Privacy Evaluation of the Federated Implementation | 42 |
| 5.2.3 | Privacy Comparison | 42 |
| 6 | Discussion and Future Work _____ | 43 |
| 6.1 | Privacy Impacts of Federated vs. Centralized Learning for Fall Detection | 43 |
| 6.2 | Performance and Accuracy Implications of Federated vs. Centralized Learning for Fall Detection | 45 |
| 6.3 | Suggested Federated Learning Integration for IoT Fall Detection: Feasibility and Challenges | 47 |
| 6.4 | Future Work | 50 |
| 6.4.1 | Privacy-Enhancing Techniques and Performance Optimization | 50 |
| 6.4.2 | Scalability and Handling of Real-Time Data Streaming | 50 |
| 6.4.3 | Economic and Energy Aspects of Federated Fall Detection | 51 |
| 7 | Conclusions _____ | 53 |

List of Figures

| | | |
|-----|--|----|
| 2.1 | Centralized learning data flow. Adaptation from [12] | 8 |
| 2.2 | Federated learning data flow. Adaptation from [12] | 8 |
| 4.1 | Evaluation of federated learning frameworks based on key selection criteria. | 24 |
| 4.2 | An illustration of window creation from the data. | 26 |
| 4.3 | (a) Label distribution of activity categories before resampling, showing a significant imbalance. (b) Label distribution after resampling. | 28 |
| 4.4 | Illustration of the CNN network. | 29 |
| 5.1 | Accuracy and loss for the centralized training process over 100 epochs using different window sizes. | 34 |
| 5.2 | Confusion matrices for the centralized model on test data using different window sizes: (a) 1.5s, (b) 2.0s, (c) 3.0s, (d) 4.0s, and (e) 5.0s. | 35 |
| 5.3 | Accuracy and loss for the federated training process across different configurations: (a) Conf. 1, (b) Conf. 2, (c) Conf. 3, (d) Conf. 4, and (e) Conf. 5. | 37 |
| 5.4 | Confusion matrices for the federated model on test data using different configurations: (a) Conf. 1, (b) Conf. 2, (c) Conf. 3, (d) Conf. 4, and (e) Conf. 5. | 38 |
| 5.5 | Accuracy and loss for the federated training process using Conf. 4 and no subject grouping | 39 |
| 5.6 | Confusion matrix for FL using Conf. 4 and no subject grouping | 39 |
| 6.1 | Fall detection dashboard in Yggio | 48 |

List of Tables

| | | |
|------|---|----|
| 2.1 | Summary of related work | 11 |
| 3.1 | Confusion matrix layout for fall detection | 15 |
| 3.2 | Evaluation metric formulas | 15 |
| 3.3 | Summary of privacy evaluation metrics | 16 |
| 3.4 | Falls recorded in the MobiAct dataset | 18 |
| 3.5 | Activities of daily living recorded in the MobiAct dataset | 19 |
| 3.6 | Summary of subjects in the MobiAct dataset | 19 |
| 4.1 | Requirements for suitable dataset | 22 |
| 4.2 | Datasets in relation to fulfilled requirements | 22 |
| 4.3 | Computed features capturing key phases of falls. | 27 |
| 4.4 | Training model parameters for the centralized implementation | 30 |
| 4.5 | Training model parameters for the federated implementation | 31 |
| 5.1 | Window size configurations | 34 |
| 5.2 | Performance of the centralized model for different window sizes with a 50% overlap. | 36 |
| 5.3 | Training configurations for federated learning | 36 |
| 5.4 | Performance of the federated model for different training configurations. Bold values highlight results from Configuration 4 (30 server rounds, 5 local epochs). | 39 |
| 5.5 | Performance metrics of the federated model using 30 server rounds, 5 local epochs and no subject grouping. | 40 |
| 5.6 | Comparison of performance results of the centralized and federated model with subject grouping. | 41 |
| 5.7 | Comparison of performance results of the centralized and federated model without subject grouping. | 41 |
| 5.8 | Privacy metrics for the centralized model. | 41 |
| 5.9 | Privacy metrics for the federated model. | 42 |
| 5.10 | Comparison of data sharing between centralized and federated implementations, showing the type and size of data transmitted during model training. The significant reduction in data size (27.68 MB) for the federated approach represents enhanced privacy protection. | 42 |

Abbreviations

- ADL** – Activities of Daily Living
AI – Artificial Intelligence
CNN – Convolutional Neural Network
FedAvg – Federated Averaging
FL – Federated Learning
IoT – Internet of Things
ML – Machine Learning
RISE – Research Institutes of Sweden

Introduction

Fall detection has become an increasingly critical area of research in response to the heightened need for healthcare solutions tailored for the aging population. Globally, falls are among the leading causes of injury, disability, and even death, especially among the elderly. According to the World Health Organization (WHO) [1], falls are the second leading cause of accidental injury deaths worldwide, with older adults particularly vulnerable due to decreased mobility and other age-related factors.

Fall incidents not only impact the physical well-being of the elderly, but also lead to substantial healthcare costs and reduce their quality of life due to a prevalent fear of falling again, which often restricts activity and independence.

Despite the need for fall detection solutions, developing an effective and privacy-conscious system remains challenging. Traditional machine learning techniques rely on centralized learning, where data from multiple users is collected in a central repository for training and analysis. This centralized approach, while effective in achieving high accuracy, comes with inherent privacy risks, as it requires the transfer and storage of potentially sensitive data in one location. In fields like healthcare, where data sensitivity and regulatory requirements are important, centralized models may not be practical due to privacy concerns and the potential for data breaches.

Federated learning offers a promising framework for achieving these goals by enabling collaborative machine learning across multiple devices while maintaining data privacy. Unlike traditional centralized machine learning methods, which require gathering and storing data in a single location, federated learning allows individual devices such as smartphones and wearable internet of things (IoT) devices to train the model locally. Rather than transmitting raw data, these devices share only model updates, which are aggregated to improve the global model without exposing individual's sensitive information [2].

1.1 Project Aim

This master's thesis includes both a primary and a secondary aim and is conducted in collaboration with Sensative AB and the Research Institutes of Sweden (RISE) as part of the project SID: Secure data sharing for Industrial Digitalization [3] funded by Vinnova within the program Advanced Digitalization.

The thesis primarily aims to develop and analyze machine learning for fall detection using federated and centralized machine learning. The primary objective is to compare and analyze how federated machine learning performs in comparison to centralized machine learning from a privacy-preserving and accuracy perspective. Furthermore, the thesis project also aims, as its secondary objective, to explore the potential of integrating federated machine learning for fall detection within Sensative's Internet of Things (IoT) platform, Yggio.

Based on the project's primary objective, the thesis project will be substantiated by addressing the following problem statement:

What is the impact of federated machine learning for fall detection from a privacy preserving and accuracy perspective compared to centralized machine learning?

1.1.1 Research Questions

To address the project's primary problem statement and fulfill its purpose, a number of research questions have been formulated. These questions align with the project's aims and help analyze different aspects of the problem statement.

Research questions 1 and 2 are directly related to the primary aim of comparing federated and centralized machine learning for fall detection in terms of privacy and accuracy. Research question 3, on the other hand, aligns with the secondary aim of exploring the integration of federated learning within an IoT platform.

RQ 1: How does federated machine learning for fall detection impact privacy preservation compared to centralized machine learning?

RQ 2: Is there any significant difference in performance and accuracy of detecting falls when comparing federated and centralized learning?

RQ 3: How can a federated machine learning approach be integrated with an IoT platform and what benefits does this hold?

1.2 Project Limitations

The scope of this project is defined by both the available resources and the constraints inherent to the chosen methodologies. A key limitation lies in the use of a publicly available dataset, which comprises activity recordings from 67 subjects with a relatively small number of fall trials. While this dataset provides a valuable foundation for developing and evaluating the fall detection framework, its size and composition pose challenges in achieving broader generalizability and robustness. The inclusion of additional data, potentially obtained from hospitals or controlled studies, would have been ideal to address these limitations. However, due to the time constraints and complexities associated with collecting such data, the creation of a new dataset is beyond the scope of this work.

Furthermore, while federated learning offers significant potential for privacy-preserving fall detection systems, the experiments in this project are confined

to simulations. The model will not be deployed on actual edge devices, such as smartphones or wearables, due to the additional development and testing efforts required for such deployment. These constraints reflect a deliberate focus on validating the core concepts and methodologies within a controlled environment, laying the groundwork for potential future work to expand and operationalize the system in real-world scenarios.

1.3 Thesis Outline

This thesis is structured to provide a clear and logical flow of information, guiding the reader through the background, methodology, implementation, and results of the study. Each chapter builds upon the previous one, ensuring a comprehensive understanding of the research process and findings. The structure is as follows:

Chapter 2: Theoretical Background and Related Work

A foundational overview of data privacy in IoT, machine learning techniques, and the concepts of centralized and federated learning. The chapter also reviews previous research relevant to the thesis.

Chapter 3: Research and Evaluation Approach

Describes the research approach, including the experimental study of the project. The evaluation framework, focusing on both performance and privacy metrics, is also presented along with the datasets considered for the project, particularly emphasizing the MobiAct dataset, as well as the tools used in the thesis.

Chapter 4: Implementation

Explains the technical aspects of the study, including dataset selection, federated framework selection, preprocessing of data, and model architecture. The implementation of both the centralized and federated learning approaches, is detailed.

Chapter 5: Results

Presents the findings from the experimental study, comparing the centralized and federated implementations. The performance and privacy aspects for the two implementations are presented.

Chapter 6: Discussion and Future Work

Reflects and discusses the results in relation to existing research and the research questions of the project. The trade-offs between performance and privacy are examined, and the benefits and challenges of integrating federated machine learning with an IoT platform such as Yggio are discussed. Additionally, a possible integration approach is presented, and suggestions for future work are proposed.

Chapter 7: Conclusions

Summarizes the study's key contributions, emphasizing its significance and potential impact. The final remarks highlight insights gained and areas for future exploration.

Theoretical Background and Related Work

This chapter provides the foundational knowledge necessary to understand the key concepts explored in this thesis. It begins with an overview of data privacy in IoT, discussing the challenges and considerations associated with handling sensitive information in connected environments. The chapter then introduces machine learning in general, outlining various learning methods before distinguishing between centralized and federated machine learning, highlighting their respective advantages and trade-offs. Finally, the chapter reviews related work, examining previous studies relevant to the topic and positioning this thesis within the broader academic and industrial landscape.

2.1 Data Privacy in IoT

In recent years, connected devices have become an increasingly large part of our everyday lives. One of the fastest growing categories of connected devices is IoT devices used for health and fitness tracking. These devices appear today to an increasing extent in our everyday life in the form of wearable devices such as smartwatches and fitness trackers or applications in our smartphones. These devices increasingly appear in our everyday lives as wearable technology like smartwatches and fitness trackers, or as applications in our smartphones [4].

While these devices can help users in a beneficial way, they also pose critical challenges when it comes to data privacy and privacy preservation. The units in question usually collect and manage large amounts of personal data linked to the users. It is becoming increasingly important to handle data in a secure way when it is shared between different organizations and services. In addition to this, challenges are also experienced when it comes to unwanted public profiles and eavesdropping. Both of these factors are a consequence of IoT devices increasingly collecting and sharing data. A big problem for the users is that, in many cases they accept the companies' terms of service without actually reading them, which in turn can lead to a range of privacy-related concerns [4][5].

A recent indicator of users' privacy concerns is the low opt-in rate for Apple's App Tracking Transparency feature. After the introduction of the new opt-out option for third-party tracking, the volume of personal data shared by third-party applications significantly decreased, highlighting a clear concern for personal data protection. The feature, introduced in iOS 14.5, requires apps to ask for permission

before tracking user data across apps and websites owned by other companies. This feature has sparked some public awareness of data privacy, with many users opting out of tracking [6].

An area that is currently growing within IoT is Internet of Medical Things. As the name suggests, this category of IoT is characterized by IoT solutions intended for healthcare. Through the use of these solutions, healthcare services can expand the capacity within traditional IoT. Common use cases include medical devices, wearables, and sensors to improve care and monitor health in real time. Within this area, privacy preservation plays a significant role as the devices handle patient data that is considered highly sensitive. It is therefore of great importance to handle privacy protection with a high priority and ensure that no risks are introduced in how the data is handled. In recent times, Internet of Medical Things solutions have also started to incorporate artificial intelligence (AI) and machine learning (ML), which can be seen as partly an advantage for healthcare but also as an increased risk when it comes to data management. To improve the privacy aspect in the area, they have therefore started using different types of machine learning techniques to preserve data privacy [7].

2.2 Machine Learning

Machine learning is very much to be considered a subfield of artificial intelligence. Broadly speaking, AI can be defined as computers' way of imitating intelligent human thinking and problem-solving abilities. That is, AI-based systems are developed to solve problems of complex nature in a way that can be compared to human solving abilities [8].

The main advantage of machine learning is that it allows us to tackle and solve problems that are too complex in nature to solve with fixed programs. Solving these problems requires a machine learning model that is created by letting the system take in large amounts of data and learn from this data [9]. The foundation to a machine learning model is that the system processes data points generated from some selected domain. These data points are advantageously distributed within two categories, features and labels. Labels correspond to characteristics that cannot be measured and identified in any simple way and usually require human assessment. Features, on the other hand, correspond to properties that a system can easily detect and calculate in an automated way. With the help of this, machine learning intends to learn to identify the label of a data point based on the features that are related to the data point. Furthermore, learning can in turn be classified into three main learning methods: supervised, unsupervised and reinforcement learning [10].

2.2.1 Learning Methods

Supervised learning uses data that has already been labeled. This means that a human expert has processed the data in advance and annotated the data with labels. With this as a starting point, supervised machine learning aims to let the system try to create a model that mimics the human annotation and predict labels based on the features of the data. This is usually achieved by fitting a curve to

the training data, i.e. the labeled data, by using a loss function to measure errors and thereby create a model that can identify both expected and deviant features. During the training process, it is also important to use validation of the model to avoid overfitting, a problem where the model becomes too specialized to the training data and performs poorly on new data. This is done by withholding part of the data to later be used as validation data. In this way, the model can be tested on data that it has not previously seen. Through this validation process, it is possible to avoid the model being overfitted [10][11].

Unlike supervised learning, unsupervised learning does not use data that has already been labeled and annotated by a human, thereby the name unsupervised learning. The learning thus takes place by letting the system see and analyze the structure and relationships within the unlabeled dataset. This type of machine learning can thus be very useful for generating a model that can identify hidden patterns or characteristics within the data. When talking about unsupervised learning, one usually refers to two unsupervised learning methods, clustering and feature learning. Clustering involves breaking down the dataset into smaller subsets of data where the data points are more similar to each other than the rest of the data. In other words, a cluster with similar data points is created. Feature learning, on the other hand, involves automatically identifying specific functions within the dataset. With the help of these features, the data points can then be efficiently analyzed and processed [10][11].

Another learning method is reinforcement learning which differs from both supervised and unsupervised learning. Reinforcement learning can in many ways be seen as a behavior based learning method. The method is based on the algorithm receiving feedback on the decisions made as the learning process takes place. In this way, the method will be strengthened and increase accuracy over time. This method differs from supervised and unsupervised learning as the method does not use labeled data for the learning process. Instead, reinforcement learning uses a trial-based system in which good decisions are rewarded and bad decisions are punished. In this way, the model is designed to achieve good results and enables improvement over time as the model is exposed to more decisions [10].

2.2.2 ML Architectures: Centralized and Federated Learning

Independently from which learning method is used, machine learning systems can be implemented using different machine learning architectures. These architectures primarily define how data is handled throughout the system and during model training. Two such machine learning architectures are the traditional centralized architecture, where data is collected and processed in a single location, and the federated architecture, where training is distributed across multiple devices without centralizing the data [12]. The following sections introduce these two architectures and their fundamental differences.

Centralized Machine Learning

Centralized learning is a conventional approach in machine learning where all training data is collected, stored, and processed in a single centralized location,

such as a cloud server or a dedicated data center. Data from various sources (e.g. user devices or databases) is aggregated into one repository, where a model is trained on the combined dataset. This process allows for high computational power, efficient data management, and straightforward model updates, as all data is accessible in a unified environment [13][12]. The information flow of this process is illustrated in Figure 2.1.

The centralized learning approach has several advantages, particularly in terms of data volume and computational efficiency. By centralizing data, machine learning models benefit from having a large and diverse dataset to train on, which often results in higher accuracy and robustness [14]. However, centralized learning poses significant challenges, particularly in domains that involve sensitive or personal data, such as healthcare, finance, and social media. The transfer of data to a central location raises substantial privacy and security concerns, as large volumes of potentially sensitive information are vulnerable to breaches and misuse [13]. Furthermore, in highly regulated fields like healthcare, data-sharing restrictions may prevent institutions from pooling data, limiting the scope of centralized datasets [15][16]. Additionally, centralized learning requires extensive data storage and computational resources, which can be cost-prohibitive.

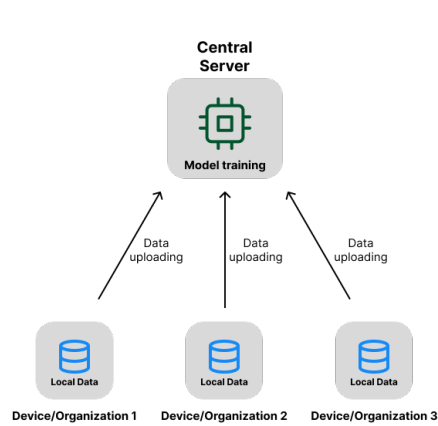


Figure 2.1: Centralized learning data flow. Adaptation from [12]

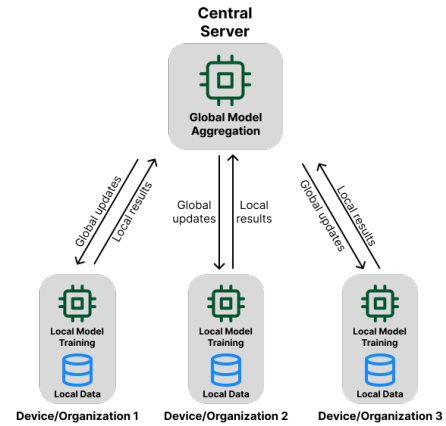


Figure 2.2: Federated learning data flow. Adaptation from [12]

Federated Machine Learning

Federated learning (FL) is a decentralized machine learning framework that allows a model to be trained across multiple devices or servers, often referred to as 'clients', without aggregating raw data into a central location. Instead, each client locally processes its data and only shares model updates, such as parameters or gradients, with a central server. This process ensures that personal data remains on the original device, thereby minimizing eventual privacy risks [12][17]. Figure 2.2 illustrates the information flow of the federated process. In contexts such as fall detection, where sensitive data about individuals' activities or health infor-

mation is frequently collected, federated learning offers a promising approach to safeguarding user privacy while striving to maintain the same performance of a centralized model.

The process of federated learning typically involves three main steps:

- **Model initialization:** The central server initializes the model with standard parameters, creating a base for training.
- **Local training:** Each device receives a copy of the global model and trains it using its own data for several local epochs (training passes on local data). This step ensures that sensitive raw data never leaves the device, maintaining privacy and reducing the risk of data exposure.
- **Model aggregation:** Once local training is complete, the devices send only the updated model parameters back to the central server. The server then aggregates these updates to form a refined global model, completing one server round (communication cycle). This process repeats for multiple server rounds to gradually improve the model [18].

Federated Averaging (FedAvg) is a key algorithm in federated learning that addresses how updates from distributed devices are aggregated to refine a global model. FedAvg combines local model updates by averaging their parameters, which are then used to update the global model [17][18].

Algorithm 1 FederatedAveraging (FedAvg). The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

```

initialize  $\omega_0$ 
for each round  $t = 1, 2, \dots$  do
   $m \leftarrow \max(C \cdot K, 1)$ 
   $S_t \leftarrow$  (random set of  $m$  clients)
  for each client  $k \in S_t$  in parallel do
     $\omega_{t+1}^k \leftarrow$  ClientUpdate( $k, \omega_t$ )
     $\omega_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} \omega_{t+1}^k$ 
  end for
end for

```

ClientUpdate(k, ω):

▷ Run on client k

```

 $B \leftarrow$  split  $\rho_k$  into batches of size  $B$ 
for each local epoch  $i$  from 1 to  $E$  do
  for batch  $b \in B$  do
     $\omega \leftarrow \omega - \eta \nabla l(w; b)$ 
  end for
end for

```

FedAvg, as described in Algorithm 1, is widely used in federated learning due to its efficiency in aggregating model updates while maintaining privacy. By averaging the locally trained model parameters, it effectively reduces communication costs and ensures convergence towards an optimal global model. However, FedAvg has limitations, such as susceptibility to client heterogeneity, where variations in data distribution across different clients can lead to biased updates and slower convergence. These challenges are particularly relevant in fall detection, where users may exhibit unique movement patterns, resulting in non-IID (independent and identically distributed) data [19].

Moreover, privacy-preserving mechanisms such as differential privacy, where random noise is introduced into the data or model parameters, can be integrated into federated learning to enhance security while maintaining model performance. These techniques ensure that even if transmitted updates are intercepted, the individual contributions of clients remain obscured. While this work does not implement these privacy-enhancing methods, they are crucial considerations for real-world federated learning applications, particularly in health-related domains where data confidentiality is paramount [18].

2.3 Related Work

Within the field of fall detection, a lot of research has been done on detecting falls using machine learning. Some examples of articles investigating this are [20], [21], [22] and [23]. Just as in the papers mentioned, it is common to use public datasets and take measurements from IoT devices of various kinds to improve the accuracy of fall detection. However, the focus usually lies on improving accuracy by the use of different machine learning techniques rather than privacy preservation.

In contrast to the previously mentioned papers, some research has also been carried out within the field of federated learning. This research focuses more on exploring federated learning as a privacy-preserving technique across various applications. One such being fatigue detection for driver safety. A study by Mohammadi et al. (2023) presents a federated learning solution aimed at monitoring driver fatigue in a way that ensures data privacy. The study addresses challenges such as data privacy risks associated with centralizing personal driver data, proposing a privacy-preserving federated learning system where local model parameters are aggregated rather than raw data [18].

The federated based fatigue detection model was able to achieve promising results, with accuracy exceeding 70%. However, the study also showed a slight decrease in accuracy when using differential privacy compared to not using the technique. This outcome demonstrates that federated approaches may hold substantial value for privacy-sensitive applications but the addition of differential privacy may affect the accuracy of the model.

Federated learning has also been applied in other use cases similar in nature to fall detection. One such being human activity recognition where federated learning was explored for its privacy-preserving capabilities. In a study by Sozinov et al. (2018), the authors applied federated learning to classify human activities, such as walking, sitting, and cycling, using sensor data from smartphones and wearable

devices. In this study they implemented both a centralized and a federated model, comparing their effectiveness across various configurations. The centralized model reached an accuracy of 93%, while the federated model achieved a slightly lower accuracy of 89%. This slight reduction in accuracy was offset by the federated model’s ability to keep data on-device, thus ensuring greater privacy for users. By employing federated learning, they were able to significantly lower the needed communication for data transfers between devices and the cloud [17].

Similarly to the human activity recognition use case, a study by Fauzi et al. (2022) explored federated learning in the context of stress detection using smart-watch data. Their study compared individual, centralized, and federated learning approaches, evaluating the trade-offs between privacy and model performance. While centralized learning achieved the highest accuracy, federated learning provided a strong balance between accuracy and data privacy, reinforcing its potential in wearable health monitoring. The findings suggest that federated learning can be an effective method for privacy-preserving data analysis in real-time health applications [24].

While federated learning may come with a slight trade-off in accuracy, it remains a viable option for applications, like fall detection, where data privacy is paramount. The comparative success of the federated human activity recognition model and stress detection demonstrates that federated solutions could achieve similarly effective results in our context, enabling sensitive health data processing without centralizing personal data.

| Papers: | Data type: | FL: | Description: |
|------------------|-------------------|------------|----------------------|
| [20] | Visual | ✗ | Fall Detection |
| [21], [22], [23] | Time series | ✗ | Fall Detection |
| [18] | Visual | ✓ | Fatigue Detection |
| [17] | Time series | ✓ | Activity Recognition |
| [24] | Time series | ✓ | Stress Detection |

Table 2.1: Summary of related work

Research and Evaluation Approach

This chapter outlines the research approach used in the thesis. It begins by describing the chosen research methodology, which primarily consists of an experimental study designed to enable a direct comparison of centralized and federated machine learning for fall detection in terms of accuracy and privacy preservation. The chapter then presents the research evaluation strategy, focusing on two key aspects: performance evaluation and privacy evaluation. The chapter also introduces the public datasets considered for the study, with a particular emphasis put on the MobiAct dataset, which plays a central role in the thesis. A presentation of the limitations with using a public dataset is also included, highlighting the main challenges and trade-offs in using a public dataset. Finally, the chapter details the tools used throughout the thesis, such as software and frameworks that supported the research and implementation of the experimental study.

3.1 Research Approach

The research approach for this thesis consists of an experimental study designed to empirically compare centralized and federated machine learning models for fall detection. This approach was chosen to ensure a systematic and controlled evaluation, aligning with the thesis main purpose and objectives.

The experimental study aims to conduct a direct empirical analysis and comparison of centralized and federated machine learning in terms of accuracy and privacy preservation. This comparison is essential to fulfill the research objectives and provide meaningful insights into the trade-offs between the two approaches. To facilitate this analysis, two implementations were implemented:

- **A centralized implementation**, where all data is collected and processed in a single location.
- **A federated implementation**, where data remains distributed between clients, and only model updates are shared.

These implementations are described in detail in Chapter 4. The decision to implement both a centralized and a federated learning model was made to fulfill the purpose and objectives of the thesis by enabling a controlled, side-by-side comparison of performance and privacy preservation. To ensure a fair evaluation, both

implementations were designed to use the same model architecture and training data, minimizing the influence of external factors such as dataset variations. This consistency ensures that any observed differences can be attributed solely to the machine learning architecture, aligning with the overall research goals.

In addition to fulfilling the thesis objectives, the choice of implementing two implementations is also justified by existing research in the field. This dual-implementation strategy aligns with established research frameworks, reinforcing its relevance and making it a well-grounded approach for this project.

3.2 Research Evaluation Strategy

The experimental study aims to assess two aspects of the fall detection system: performance and privacy preservation. To achieve this, two distinct strategies have been devised to facilitate the analysis of these aspects. One strategy is used to evaluate performance, while the other is used to assess privacy preservation. Presented below is a detailed description of how these aspects are investigated within the project.

3.2.1 Performance Evaluation

To assess the performance of the models developed for fall detection in both centralized and federated learning scenarios, we adopt a comprehensive evaluation strategy. The effectiveness of these models is analyzed through empirical results, focusing on the trade-off between predictive accuracy and the privacy benefits offered by federated learning. This section outlines the metrics and methods used for performance evaluation.

The performance metrics employed include accuracy, precision, recall, and F1-score, which are widely used in related work for classification tasks [25]. These metrics provide a holistic view of the model's ability to correctly identify falls while minimizing misclassifications. The definitions and interpretations of these metrics are based on the confusion matrix, which encapsulates the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) counts. True Positives (TP) refer to instances where the model correctly identifies a fall as a fall, while True Negatives (TN) represent cases where the model accurately identifies a non-fall as not being a fall. Conversely, False Positives (FP) occur when the model incorrectly classifies a non-fall as a fall, leading to unnecessary alarms. False Negatives (FN), on the other hand, are instances where the model fails to identify a fall, incorrectly classifying it as a non-fall, which is critical to minimize in fall detection systems [26][25].

A confusion matrix is a tabular representation of the model's predictions against the actual labels. The following components of the confusion matrix are defined:

| True Values | Predicted Values | |
|-------------|---------------------|---------------------|
| | Fall | Not Fall |
| Fall | True Positive (TP) | False Negative (FN) |
| Not Fall | False Positive (FP) | True Negative (TN) |

Table 3.1: Confusion matrix layout for fall detection

Based on the confusion matrix, the following evaluation metrics are calculated:

- Accuracy: The ratio of correctly classified instances (both falls and non-falls) to the total number of instances.
- Precision: The ratio of correctly identified falls to all instances predicted as falls.
- Recall: The ratio of correctly identified falls to all actual falls.
- F1-Score: The harmonic mean of precision and recall, providing a balanced metric that considers both false positives and false negatives.

| Metric | Formula |
|-----------|---|
| Accuracy | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Precision | $\frac{TP}{TP+FP}$ |
| Recall | $\frac{TP}{TP+FN}$ |
| F1-Score | $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$ |

Table 3.2: Evaluation metric formulas

This systematic approach ensures a thorough and reliable evaluation of the models' performance, highlighting their strengths and limitations in the context of fall detection [26][25].

3.2.2 Privacy Evaluation

Comparing privacy preservation in centralized and federated machine learning setups comes with several challenges, especially when it comes to choosing the right evaluation metrics. The way data is handled and what is to be considered a privacy risk can vary depending on the approach and use case. Previous research has explored different ways to assess privacy risks in machine learning, often using direct privacy attacks or analyzing how data is shared [27]. However, implementing attack methods is outside the scope of this project. Instead, the thesis focuses the privacy evaluation on what type of data is being shared between client and server as well as the size of the data that is transferred.

The type of data being shared is a crucial factor in protecting privacy. Shared data can contain sensitive information, so understanding what is actually being transmitted helps identify potential risks. Since different machine learning architectures handle data differently, looking at this aspect provides a broader perspective on privacy concerns [28][27]. Instead of focusing on specific attack techniques, this approach makes it possible to compare centralized and federated implementations in a way that is relevant to real-world privacy challenges.

The amount of data transferred between client and server also plays a big role in privacy risks. The more data that gets sent, the higher the chances of accidental leaks, no matter what type of data it is [28]. By measuring and comparing the volume of transferred data, it can be conceivable that privacy trade-offs can be assessed objectively without depending on specific attack methods. This also makes the evaluation easier to reproduce and apply in different scenarios.

By focusing on these two aspects, this evaluation provides a structured yet practical way to compare privacy risks between centralized and federated machine learning implementations. The selected factors are fundamental in understanding data exposure while keeping the analysis broad enough to be useful across different contexts. This method allows for a general and reproducible privacy assessment without getting tied down to specific attack strategies.

The evaluation metrics are presented in a summarized way in Table 3.3 below.

| Aspect | Metric |
|--------------------------|---|
| Type of shared data | Data type/category (e.g., raw data, gradients, model updates) |
| Size of transferred data | Data size in bytes |

Table 3.3: Summary of privacy evaluation metrics

3.3 Dataset

A crucial component of the experimental study is the dataset used for training and evaluation of the fall detection models. In machine learning research, publicly available datasets are often utilized to ensure reproducibility and comparability with previous studies. To support the experimental approach chosen in this thesis, several public datasets were evaluated before selecting the most suitable one for this project. The following section presents the datasets considered and provides details on the dataset that was ultimately chosen for this study.

3.3.1 Public Datasets

Machine learning research in fall detection often relies on publicly available datasets, as seen in studies such as [20], [21], [22] and [23], which all use public datasets for research purposes. These datasets provide pre-recorded data for both activities of daily living (ADL) and falls, enabling model training without the need for

new data collection. Several datasets were evaluated for this project, each offering different characteristics in terms of sensor data, activity types, and participant demographics. Below is an overview of the datasets considered for the experimental study.

SisFall - The SisFall dataset is intended for fall detection research, designed to address some critical requirements in this field. It is a publicly available dataset that offers data for researchers working on solutions for fall and activity detection. SisFall includes data from 38 participants across different age groups, thus ensuring age diversity among participants.

The dataset records accelerometer data alongside gyroscope data, capturing both falls and ADLs across 34 distinct activities. The acceleration data was gathered with a self-developed device, fixed to participants' waists, which included triaxial accelerometers.

Furthermore, the data was gathered under controlled conditions using mats to ensure safety while striving to maintain realistic fall dynamics. The dataset notably includes both simulated falls from young adults and one elderly participant trained in Judo. The other elderly participants only participated in the ADL data recordings [29].

MobiAct - MobiAct is a publicly available dataset developed by The Biomedical Informatics & eHealth Laboratory (BMI Lab) in Greece to identify different ADLs as well as falls. The dataset is an extended version of the previously released MobiFall dataset, which appears in several studies in the field of fall detection and activity recognition. MobiAct contains collected data for several ADLs and four different types of falls. The participants in the dataset come from different ages, genders, weights and heights. The collected data consists of accelerometer and gyroscope data collected from the built-in sensors of a smartphone. In an attempt to imitate real-world conditions, the phone was carried in the trouser pocket of the participants when the data collection took place. In order to further try to imitate real situations, the data set also contains several scenarios that can arise in everyday life [30][31].

WEDA Fall - The WEDA Fall dataset is a dataset created with data collected from the Fitbit Sense wrist device. WEDA Fall is publicly available and the collected data consists of both accelerometer data and gyroscope data from the wrist device. The dataset includes data for 11 ADLs and 8 different types of falls collected from a diverse group of participants of different ages in a controlled setting. 14 of the participants were classified as young people and 11 participants were classified as elderly people with an age of over 80 years old. However, participants in the elderly group was not allowed to participate in the collection of fall data [32][33].

3.3.2 The MobiAct Dataset

Among the datasets considered, MobiAct was selected as the most suitable datasets for this thesis project. This section presents the MobiAct dataset in more detail whereas the evaluation and selection process for the dataset are described in Section 4.1.

The MobiAct dataset includes records from four different types of simulated falls, presented in Table 3.4, as well as 11 types of ADLs, presented in Table 3.5. In addition to the 11 ADLs, the dataset also includes one activity for lying (LYI) which is a consequence of the inactivity period after a fall by the participants. In total the dataset includes 67 subjects, both male and female, with varying age, height and weight. A summary of the participating subjects is presented in Table 3.6. The falls were performed by all participants, 19 of the participants performed all 11 ADLs and 59 participants performed nine of the ADLs. In addition to the falls and ADLs, the dataset includes five sub-scenarios of daily living which consists of a sequence of 50 activities performed by 19 of the participants.

To simulate real-world conditions, the smartphone was placed loosely in a participant's trouser pocket, with random orientation to reflect typical daily use. Each activity and fall event was captured under controlled conditions, utilizing a cushioned surface to minimize injury risk during falls. The dataset's structure supports testing and comparison of different fall detection models, as it includes diverse sensor readings across activities that may trigger false positives in fall detection models [30].

| Label: | Activity: | Trials: | Duration: | Description: |
|--------|--------------------|---------|-----------|---|
| FOL | Forward-lying | 3 | 10s | Fall forward from standing, use of hands to dampen fall |
| FKL | Front-knees-lying | 3 | 10s | Fall forward from standing, first impact on knees |
| BSC | Back-sitting-chair | 3 | 10s | Fall backward while trying to sit on a chair |
| SDL | Sideward-lying | 3 | 10s | Fall sideways from standing, bending legs |

Table 3.4: Falls recorded in the MobiAct dataset

| Label: | Activity: | Trials: | Duration: | Description: |
|---------------|--------------------------------|----------------|------------------|---|
| STD | Standing | 1 | 5min | Standing with subtle movements |
| WAL | Walking | 1 | 5min | Normal walking |
| JOG | Jogging | 3 | 30s | Jogging |
| JUM | Jumping | 3 | 30s | Continuous jumping |
| STU | Stairs up | 6 | 10s | Stairs up (10 stairs) |
| STN | Stairs down | 6 | 10s | Stairs down (10 stairs) |
| SCH | Stand to sit (sit on chair) | 6 | 6s | Transition from standing to sitting |
| SIT | Sitting on chair | 1 | 1min | Sitting on a chair with subtle movements |
| CHU | Sit to stand (chair up) | 6 | 6s | Transition from sitting to standing |
| CSI | Car-step in | 6 | 6s | Step in a car |
| CSO | Car-step out | 6 | 6s | Step out a car |
| LYI | Lying | 12 | - | Activity taken from the lying period after a fall |

Table 3.5: Activities of daily living recorded in the MobiAct dataset

| Gender | Number of subjects: | Age (years) | Height (cm) | Weight (kg) |
|---------------|----------------------------|--------------------|--------------------|--------------------|
| Male | 48 | 20 - 47 | 169 - 193 | 62 - 120 |
| Female | 16 | 20 - 36 | 158 - 175 | 50 - 90 |
| Unknown | 3 | 21 - 24 | 175 - 187 | 70 - 85 |

Table 3.6: Summary of subjects in the MobiAct dataset

3.3.3 Limitations with Public Datasets

In this project, one key limitation in the choice of dataset stems from the fact that, like many studies in this field [20], [21], [22] and [23], the implementation in this project rely on simulated falls recorded in controlled settings rather than data from real-world incidents. Falls, by nature, are unexpected and involuntary, making them difficult to capture authentically, especially for vulnerable populations who experience these events most frequently [34].

As the researches point out in [34], simulations cannot fully replicate the spontaneous nature of real falls, often leading to differences in movement dynamics and landing impacts that may affect the efficacy of detection solutions.

In a study by Fabio Bagalà et al., it was found that the accuracy of several threshold-based fall detection algorithms decreased considerably when applied to real falls, as opposed to simulated ones. This discrepancy likely stems from the fact that these algorithms were developed using simulated datasets, which cannot fully capture the spontaneous dynamics of real-world falls. However, it is important to note that Bagalà et al. did not employ machine learning, and it remains possible that machine learning models, trained on a mix of data types, could perform better in detecting real falls [35].

This limitation, though common in current datasets, potentially reduces the real-world applicability of our model. Nonetheless, public datasets allow us to address other essential requirements such as including a diversity of fall types and actions of daily living, meeting the project's scope and timeline while providing a foundational step towards real-world application in fall detection.

3.4 Tools

The selection of tools for this study was guided by the specific requirements of fall detection using centralized and federated learning, including the need for flexibility. Several tools were chosen to facilitate the implementation and effectively meeting these needs.

Python (3.9) was chosen as the primary language due to its extensive ecosystem of libraries and frameworks, making it an industry/standard choice for data analysis and machine learning. Together with python it was also chosen to use PyTorch (2.2.2) for implementing and training the machine learning models. Its intuitive interface and dynamic computation graph capabilities, and strong industry backing makes PyTorch a standard choice of framework for both research and production environments and hence a suiting choice for this project as well.

Flower (1.13.1) was used as the federated learning framework for the thesis project. Flower offers ease of use, minimal setup overhead as well as compatibility with the requirements of the fall detection system. This decision is described in more detail in Section 4.2.

Implementation

This chapter details the implementation and the technical aspects of the experimental study, beginning with the dataset selection and the choice of a suitable federated learning framework. It then describes the data preprocessing steps required to prepare the dataset for training. The chapter further outlines the model architecture, followed by the implementation of both centralized and federated learning approaches.

4.1 Dataset Selection

A common and essential factor of both the centralized and federated implementation in this project is the dataset on which the learning process is based. There are several ways to acquire data for the learning processes presented in this project. Initially, it was discussed to acquire and compile a dataset using sensors or a motion tracking app. However, it was noted that this method would be far too time-consuming and unsuited for the project's time frame and scope. Another option, that also occurs in the majority of other research projects in the area, was to use a publicly available dataset.

In order to use a public dataset, a number of requirements were placed on the dataset and the data that the set would contain. The requirements were also formulated to ensure and strengthen the scientific validity of the project. The focus of these requirements was on the availability of the data, the nature of the data contained within the dataset, the methodology used for data collection, and the individuals who participated in the data collection process. The first requirement was that the dataset and its data should be publicly accessible. Fulfilling this requirement held significant importance for the project, both in ensuring accessibility for its implementation and in ensuring its scientific validity in comparison to other research conducted within the field. In terms of the data and data type of the dataset, the requirements specified that the data should contain accelerometer data, include both fall and ADL data, and be collected using the embedded sensors found in smartphones. These requirements were set since accelerometer data serves as a key component for identifying falls in relation to other ADLs and to have a basis for both falls and ADLs for learning and training of models. Furthermore, the requirement to collect data from smartphones was intended to facilitate integration with any IoT platforms and collection applications that han-

dle time series data. Smartphones can also be viewed as a resource that already incorporates accelerometer sensors, thereby enhancing the practical relevance of the project and enabling future applications. In relation to the collected data, the requirements also formulate that the collection should be conducted with a diverse group of participants. This aspect was seen important as an increased variability of participants enables a more generalizable representation of the models derived from training on the data. An additional aspect that was added into the requirements was that the data should be collected from real-world scenarios. It is conceivable that falls and activities performed in controlled setting could be considered misleading in relation to real-world situations, which the requirement aims to minimize. The requirements imposed are presented in Table 4.1.

| Req: | Description: |
|-------|--|
| Req 1 | Publicly available data |
| Req 2 | Contain accelerometer data |
| Req 3 | Include fall and ADL data |
| Req 4 | Data recorded from a smartphone |
| Req 5 | Diversity in age represented within participants |
| Req 6 | Data recorded from real world situations |

Table 4.1: Requirements for suitable dataset

For this thesis project, a number of datasets were evaluated with the requirements in Table 4.1 in mind. Three datasets, each presented in Section 3.3.1, were selected for the evaluation due to their availability, relevance for the project and relation to existing research. Presented below is a compilation of the datasets that have been evaluated in relation to their fulfillment of the requirements.

| Dataset: | Req 1: | Req 2: | Req 3: | Req 4: | Req 5: | Req 6: |
|-----------|--------|--------|--------|--------|--------|--------|
| SisFall | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| MobiAct | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| WEDA Fall | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |

Table 4.2: Datasets in relation to fulfilled requirements

As a result of the evaluation of public datasets, it was decided that MobiAct was the best suited dataset for the implementation in this project. One justification for the choice is that MobiAct fulfills the majority of the requirements in Table 4.1. A trade-off with the MobiAct dataset is that it does not contain fall data from elderly people. However, this is not seen as a problem for fall detection in comparison to the other datasets since SisFall and WEDA Fall did not allow for a general representation of elderly participants falling. In contrast to the other datasets, MobiAct is produced using a Samsung Galaxy smartphone. This,

in combination with the fact that the dataset contains scenarios to reflect real situations makes MobiAct the best suited dataset for the project.

4.2 Framework Selection

The choice of framework for the federated implementation plays a key role in how the federated solution is implemented. The framework dictates how models are trained and how privacy is preserved across distributed devices. Therefore, a number of frameworks were evaluated for this project, including frameworks such as Flower [36], NVIDIA FLARE (NVFLARE) [37], OpenFL [38], PySyft [39], IBM Federated Learning [40], and TensorFlow Federated [41].

The evaluation criteria for this assessment were developed in collaboration with RISE, leveraging their expertise in federated learning and privacy-preserving machine learning. The criteria were established based on general considerations in federated learning, rather than being tailored specifically to this project. The aim was to ensure a balance between technical capabilities and practical usability. The primary goal of the evaluation was to identify a framework that not only met the technical requirements of federated learning but also ensured accessibility to those with limited experience in the field. To assess these frameworks, several key aspects were explored to guide the selection process, including:

- **Documentation:** The quality of documentation varies across frameworks. A user-friendly documentation is critical for understanding the framework's capabilities, particularly for beginners.
- **ML Framework Support:** Support for major machine learning frameworks such as PyTorch and Tensorflow.
- **Beginner Friendliness:** Accessibility of the framework, including ease of setup, intuitive workflows, and availability of beginner-friendly tools or pre-configured settings.
- **Privacy Methods:** Support for integrity based solutions such as differential privacy.
- **Model Support:** The support of different model types can be limited.
- **Scalability:** The capability to scale across multiple devices and nodes.

The radar diagrams presented in Figures 4.1a-4.1f illustrate the evaluation of the federated learning frameworks, with each investigated aspect scored on a scale from 1 to 5. These visualization highlight the strengths and weaknesses of each framework.

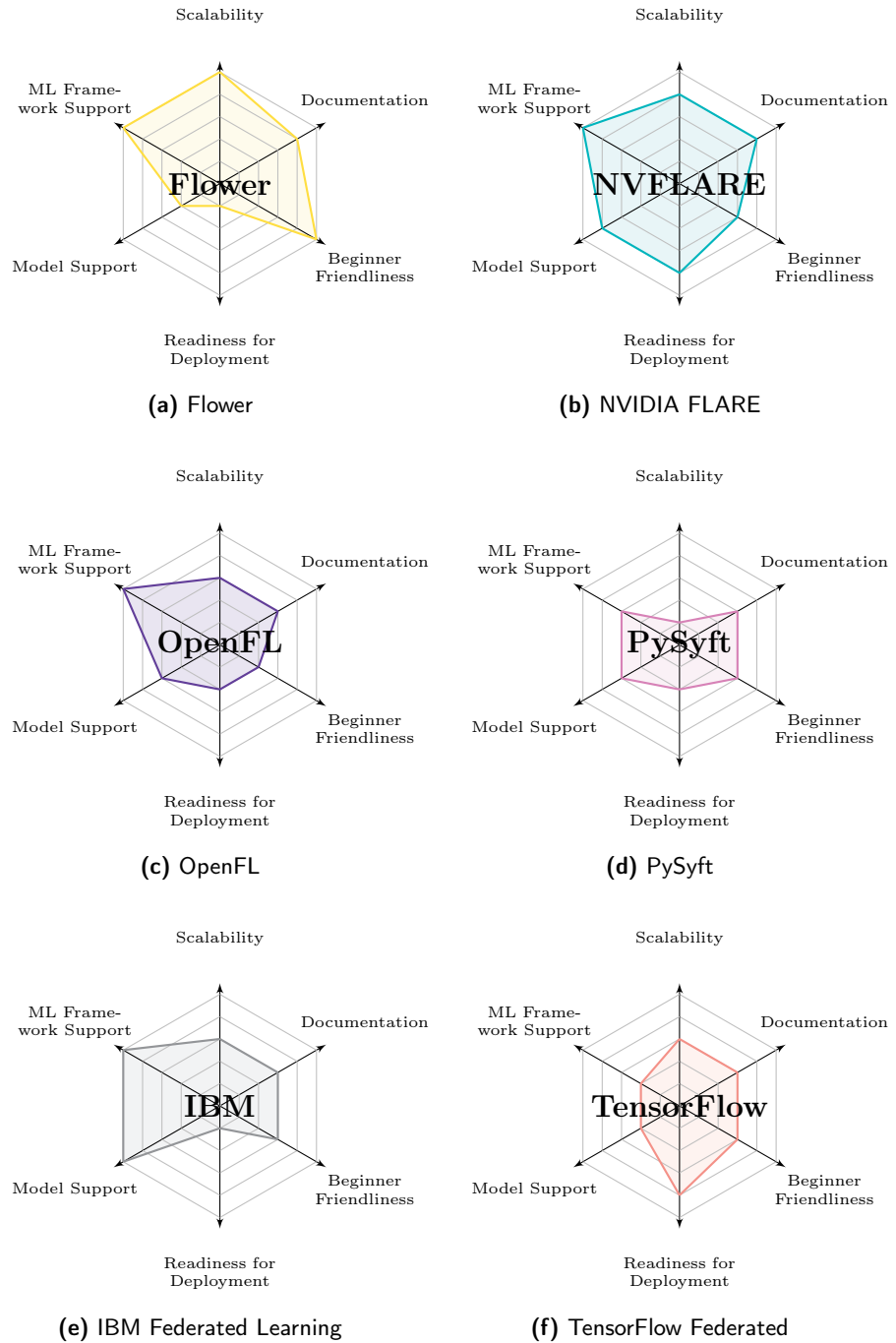


Figure 4.1: Evaluation of federated learning frameworks based on key selection criteria.

Each framework offers distinct advantages and trade-offs, as illustrated in Figures 4.1a-4.1f above. Among them, two frameworks emerge as particularly strong candidates for this project: Flower and NVIDIA FLARE. Both excel in framework support, scalability, and documentation. However, NVIDIA FLARE has an additional advantage in terms of model support and deployment readiness. While these factors are valuable in production environments, they are of lesser relevance to this project, as they do not directly contribute to its goals and purpose. Flower, on the other hand, surpasses NVIDIA FLARE when it comes to beginner friendliness, a crucial factor given the project's focus on research and iterative experimentation.

Based on the evaluation, Flower was chosen as the most suitable framework for this project due to its strong documentation, ease of use, and flexibility, particularly in a research setting. Unlike frameworks that prioritize model support and production readiness, Flower is designed with accessibility in mind, making it an excellent choice for iterative experimentation and prototyping, which aligns well with the project's scope. Additionally, a key factor in the decision was the limited project time frame, where ease of setup and a minimal learning curve were essential considerations. Flower provides an intuitive framework with strong support for major machine learning libraries such as PyTorch, allowing for efficient experimentation. Moreover, having access to guidance from RISE, whose team has prior experience with Flower, further reinforces its suitability, ensuring that potential challenges can be resolved efficiently.

Given these factors, Flower strikes the best balance between accessibility, flexibility, and research applicability within the given time constraints of this project, making it the most practical choice for this specific project.

4.3 Data Preprocessing

To prepare the data for the fall detection model, a comprehensive preprocessing pipeline was developed, encompassing several key steps: labeling, windowing, feature extraction, and dataset balancing. Each step was designed to ensure the data was well-suited for the model's requirements and to enhance the overall robustness of the analysis.

The preprocessing began with data labeling. The raw data, obtained from the MobiAct dataset, was initially labeled using activity codes that represented various types of activities. These codes were numerically encoded using a standard encoder from SciKit Learn to facilitate their use in subsequent modeling tasks [42]. However, as the use case of this study is fall detection rather than general activity classification, binary labels were assigned to each record. Specifically, activities associated with falls were labeled as "1" while all other activities were categorized as "0" creating a binary classification framework suitable for detecting fall events.

To enhance the quality of the sensor data and minimize the impact of high-frequency noise, a low-pass filter was applied as a part of the preprocessing pipeline. Given that human movement and fall dynamics predominantly occur within lower frequency ranges [21], high-frequency components are often associated with sensor noise rather than meaningful activity patterns. By filtering this noise, the filter

ensures that the extracted features more accurately reflect the underlying motion characteristics.

A fourth-order Butterworth low-pass filter was implemented, with a cutoff frequency set at 20Hz. This choice was based on prior studies. This filter was applied to each axis of the data of the three sensors, accelerometer, gyroscope and orientation sensors.

Following the filtering process, the data underwent segmentation using a sliding window approach. This method was chosen based on findings from prior research, [43][44][45], which demonstrated the effectiveness of time windows ranging from 1 to 5 seconds for accelerometer data. Sliding windows simplify the practical implementation of fall detection models, allowing real-time detection by analyzing data from the most recent time segments. The size and step of the windows were carefully selected to balance the need for capturing adequate contextual information with the computational efficiency of the model. These choices were informed by [43], which utilized time windows of 1.5 seconds with a 50% overlap for activity recognition tasks with favorable outcomes. Each window was then assigned a binary label, fall or non-fall, based on the proportion of fall events within it. A predefined threshold determined whether a window was classified as a fall (if the threshold was exceeded) or a non-fall.

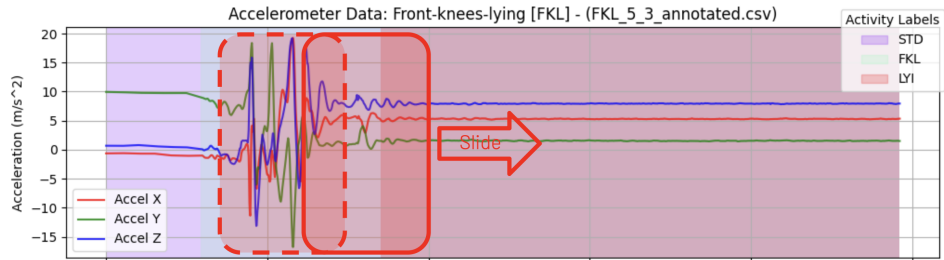


Figure 4.2: An illustration of window creation from the data.

Feature extraction formed the next phase of the preprocessing pipeline. Guided by insights from previous studies on fall detection and activity recognition [46][47], a total of 54 features were derived from each time window. These features were chosen to encapsulate critical characteristics of the sensor data and enhance the model's ability to distinguish between fall and non-fall activities.

As described in [47], three features were extracted from the accelerometer data: the acceleration magnitude vector (AVM), the maximum absolute value of the acceleration (MAV) and the sum of absolute values (SAV).

$$\text{AVM}(i) = \sqrt{a_x^2(i) + a_y^2(i) + a_z^2(i)} \quad (4.1)$$

$$\text{MAV}(i) = \max[|a_x^2(i)|, |a_y^2(i)|, |a_z^2(i)|] \quad (4.2)$$

$$\text{SAV}(i) = \sum_{i \in DW} |a_{x,y,z}|(i) \quad (4.3)$$

These features were then used to further compute additional features, presented in Table 4.3, that are relevant for falls. These metrics capture key phases in typical fall activities, such as the weightless, impact, and static stages, providing a comprehensive understanding of the underlying dynamics [47].

| Description | Representation: |
|---|-----------------------------|
| $MAV_{\min} = \min_{i \in DW} MAV(i)$ | Weightless stage |
| $MAV_{\max} = \max_{i \in DW} MAV(i)$ | Impact stage |
| $AVM_{\min} = \min_{i \in DW} AVM(i)$ | Weightless stage |
| $AVM_{\max} = \max_{i \in DW} AVM(i)$ | Impact stage |
| $AVM_{pp} = AVM_{\max} - AVM_{\min}$ | Weightless and impact stage |
| $AVM_{\text{mean}} = \frac{1}{N_{DW}} \sum_{i \in DW} AVM(i)$ | Static stage |

Table 4.3: Computed features capturing key phases of falls.

The slope feature was derived from the triaxial accelerometer data to capture the rate of change across axes. This was computed for both the raw accelerometer readings and the absolute values of the x , y , z components.

$$\text{Slope} = \sqrt{(\max_x - \min_x)^2 + (\max_y - \min_y)^2 + (\max_z - \min_z)^2} \quad (4.4)$$

The tilt angle feature quantifies the orientation of the device relative to the gravitational vector, with a focus on the y -axis.

$$\text{TiltAngle}_i = \arcsin \frac{y_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}} \quad (4.5)$$

This feature measures the angle between the device’s y -axis and the gravitational vector, effectively capturing the device’s inclination during motion or rest. The tilt angle is particularly relevant for fall detection as it provides insights into posture changes and abnormal movements. Sudden or extreme deviations in the tilt angle can be indicative of activities such as falling or stumbling, making it a valuable addition to the feature set for detecting and analyzing such events [46].

In addition to the tilt angle, various statistical features were computed from the accelerometer, gyroscope and the orientation data along the x , y , z axes. These include the maximum, minimum, mean, standard deviation, median, skewness, and kurtosis of both the raw data and the absolute values for each axis. These statistical descriptors capture the distribution and variability of sensor readings, providing a richer understanding of the motion patterns. Specifically, for each of the three axes, seven statistical measures were calculated, yielding a total of 42 additional features.

The processed dataset, derived from the feature engineering methods described above, revealed a significant class imbalance between processed fall and ADL windows. While the dataset contains a diverse range of ADL activities such as walking, standing, and sitting, the number of fall samples is disproportionately small

in comparison. This imbalance poses a major challenge in training an accurate fall detection model.

One key issue with imbalanced datasets is the risk of developing a model biased toward the majority class - in this case, ADL activities. A model might learn to predict "no fall" for all samples, resulting in seemingly high accuracy due to the overwhelming presence of ADL data. However, such a model would fail at its primary purpose: detecting falls. This limitation underscores the importance of addressing the imbalance before model training.

To mitigate this issue, we applied a combination of oversampling and down-sampling techniques to balance the dataset. First, the fall samples were oversampled to match the number of ADL samples. Simultaneously, the ADL activities were uniformly downsampled and oversampled to achieve a balanced and even distributed representation of ADL data. This ensured that no single ADL activity dominated the dataset while maintaining the diversity of activities. Similarly, the four types of falls in the dataset were uniformly balanced using the same approach.

Label Distributions Before and After Resampling

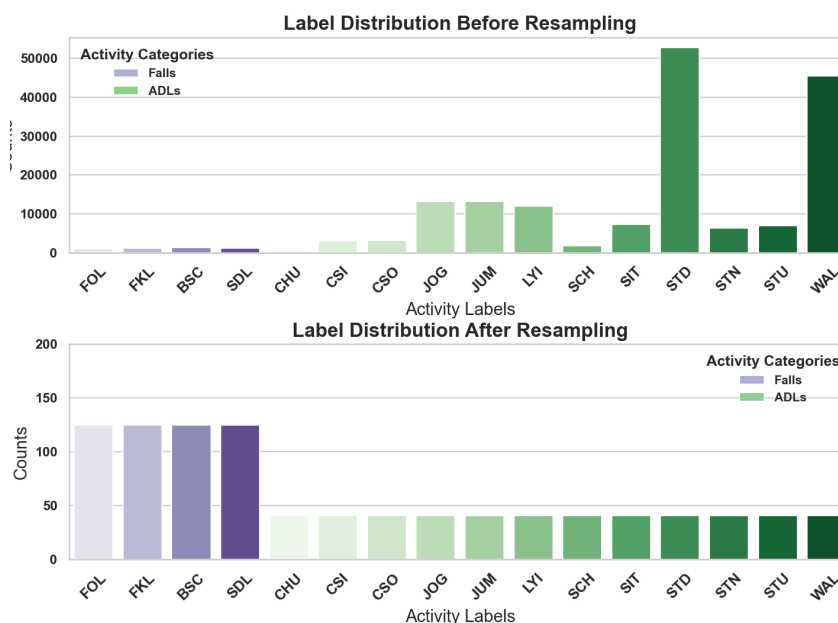


Figure 4.3: (a) Label distribution of activity categories before resampling, showing a significant imbalance. (b) Label distribution after resampling.

This resampling strategy resulted in a more balanced dataset, improving the model's ability to differentiate between fall and non-fall events. In Figure 4.3, the effects of the introduced resampling can be observed resulting in a substantially smaller set of ADL samples in proportion to fall data.

4.4 Model Architecture

The basis for the experimental study in this thesis project is that a centralized and a federated machine learning implementation are compared. In order for these two implementations to be compared, it is of great importance that they are both based on the same machine learning model and that they are trained on the same dataset.

The model chosen for the two implementation is a convolutional neural network (CNN) model, which is illustrated in Figure 4.4. A CNN is a class of deep learning models particularly effective for feature extraction and classification tasks. Originally developed for image processing, CNNs have also been successfully applied to sequential data, including time series [48]. CNNs use convolutional layers to detect local patterns in data, pooling layers to reduce dimensionality while retaining important features, and fully connected layers for classification [49][26].

The convolutional layer is the core component, applying small kernel to input data to detect local features. These kernels slide across the data, performing a convolution operation that creates feature maps. The network learns to detect increasingly complex patterns as data progresses through additional convolutional layers. Pooling layers downsample the feature maps, reducing dimensionality while retaining important information. Max pooling is the most common method, selecting the highest value in a region to preserve dominant features. Finally, fully connected layers interpret the extracted features and produce a classification output. A softmax activation function is typically used for classification tasks, outputting probabilities for different categories [49].

The justification for choosing a CNN in this project is that it is well suited for processing sequential data, such as accelerometer and gyroscope signals, which are used in this study. CNNs excel at capturing local characteristics in time series data and identifying relevant patterns, making them a suitable choice for fall detection [48].

The model used in this project consists of two convolutional layers, two pooling layers, two fully connected layers as well as an output layer.

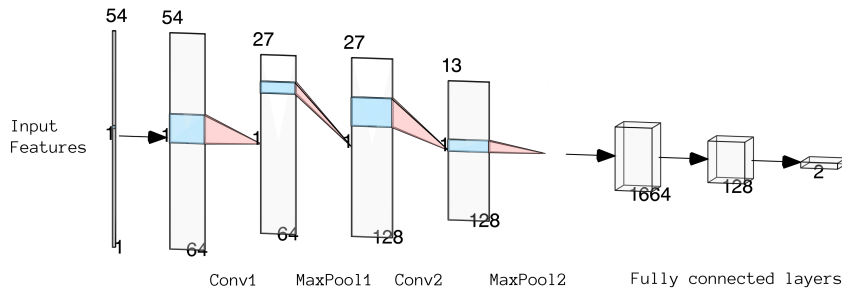


Figure 4.4: Illustration of the CNN network.

As illustrated in Figure 4.4, the input to the model is a feature vector of length 75, equal to the amount of features computed from a sliding window of sensor data. The first convolutional layer (Conv1) applies 64 kernel convolutions, producing an output of shape (64, 54). A max pooling layer (MaxPool1) reduces the size by half, resulting in (64, 27). The procedure then repeated a second time with the second convolutional and pooling layer. After the convolutional and pooling operations are done, the extracted features are flattened into a vector of size 1664 and passed through two fully connected layers (with 128 neurons in the hidden layer). The final output layer consists of two neurons, representing the two classes in the binary classification task: fall and no fall.

4.5 Centralized Implementation

For the centralized implementation a supervised learning process is used where the data was first split into two partitions, one for training and one for testing of the trained model. The partitioning of the data corresponds to 80% of the data being used for training while 20% of the data was used for testing. This step was done partly to enable the evaluation of the models accuracy on data that was not included in the training process, and to ensure that the model was not over-fitted during training process. All data was also normalized together at the beginning to ensure consistency in feature scaling across the entire dataset. The model was trained using the parameters presented in Table 4.4 below.

| Parameter: | Value: |
|------------------------|---------|
| Batch size | 64 |
| Learning rate | 0.0001 |
| Number of epochs | 100 |
| Window size (seconds) | 1.5-5.0 |
| Train data target size | 2000 |
| Test data target size | 500 |

Table 4.4: Training model parameters for the centralized implementation

The Adam optimizer was selected for the training process as it is a widely used optimization algorithm known for its efficiency and adaptive learning rate adjustments [50]. Similarly, the cross-entropy loss function was chosen to measure the performance of the classification, as it is a standard criterion for classification tasks [51]. Additionally, the dataset was resampled to address the imbalance between fall and non-fall samples, ensuring that the model could learn to differentiate between both classes effectively.

After training for 100 epochs, the model's performance was evaluated using the test dataset to assess its ability to generalize to unseen data. The resampled dataset, as shown in Figure 4.3, improved the model's ability to distinguish between fall and non-fall events, as the dataset became more balanced.

4.6 Federated Implementation

In the federated implementation, the same CNN model architecture and supervised learning method as in the centralized case was used to ensure consistency in the comparison. Unlike the centralized implementation, which was developed and evaluated using multiple window sizes, the federated implementation was only developed using a window size of 3.0 seconds. This decision was made because 3.0 seconds was found to be the optimal window size for the centralized model, and the results from the centralized model were based on this window size. By using the same window size in the federated implementation, we aimed to minimize the differing factors between the two implementations and aid in the direct comparison of the models. After splitting the data into partitions, a federated simulation using the Flower framework was initiated. Flower was chosen because of its excellent documentation, machine learning framework support, and ease of use, particularly for research purposes. The model was trained using the parameters presented in Table 4.5 below.

| Parameter: | Value: |
|-------------------------|---------------|
| Batch size | 64 |
| Learning rate | 0.0001 |
| Number of local epochs | 1-50 |
| Number of server rounds | 3-150 |
| Window size (seconds) | 3.0 |
| Train data target size | 400 |
| Test data target size | 100 |

Table 4.5: Training model parameters for the federated implementation

The federated implementation included two different setups related to how data was distributed in relation to the number of clients. Partly, a study was done where the participants in the dataset were grouped into six number of groups, i. e. a distribution with roughly 12 subjects per client, and an evaluation where each client corresponded to an individual participant in the dataset. This was done partly because it relates to a more reality-based situation where it is conceivable that, for example, a sub-organization manages all its data which is then shared further. An additional factor that plays a role in this evaluation is linked to the size of the dataset where it can have an impact on how much data each client has available for training.

These two setups were explored to analyze the impact of data distribution on the federated learning process. The dataset size and the amount of data each client has available for training are important factors that can influence the performance of the model. This distinction between group-based and individual-based data distribution is crucial in understanding how federated learning can be applied to real-world settings, where different clients may have varying amounts of data.

This chapter presents the findings from the experimental study, categorizing the results into two main areas: performance and privacy. It begins by evaluating performance-related results for both the centralized and federated implementations, followed by an objective comparison of their effectiveness. Next, the chapter presents the privacy-related results, assessing how well each approach preserves data security.

5.1 Performance Evaluation

The results presented below are the results related to the performance evaluation of the experimental study. The presentation is divided into two subsections containing the results for the centralized and federated implementations respectively. Included is also a subsection for comparison of the performance results of the two implementations.

5.1.1 Performance of the Centralized Implementation

Illustrated below in Figure 5.1 are the accuracy and loss for the centralized training process across five different window size configurations taken over 100 epochs. The different window sizes used in the training processes ranges from 1.5s to 5.0s. All configurations are presented in Table 5.1. One notable observation is that the accuracy and loss curves are remarkably similar across all configurations. This can suggest that the window size has a negligible impact on the current training process.

| Training Run: | Window Size: |
|---------------|--------------|
| Run 1 | 1.5s |
| Run 2 | 2.0s |
| Run 3 | 3.0s |
| Run 4 | 4.0s |
| Run 5 | 5.0s |

Table 5.1: Window size configurations

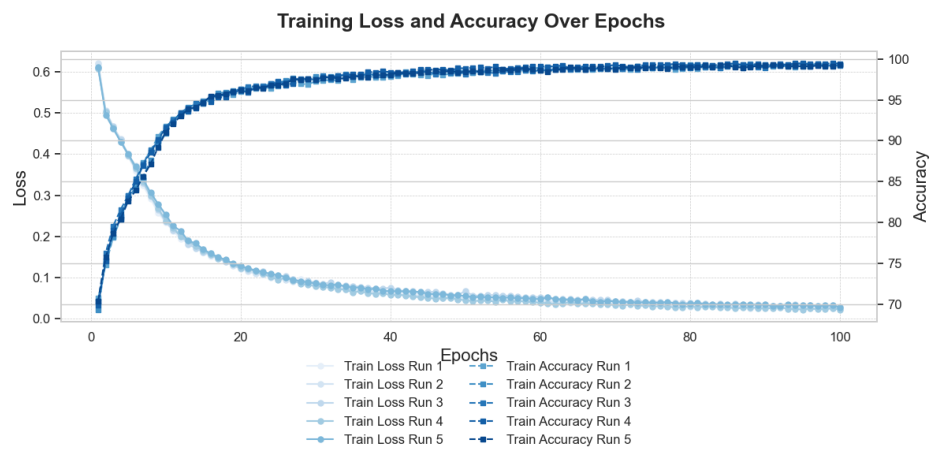


Figure 5.1: Accuracy and loss for the centralized training process over 100 epochs using different window sizes.

In contrast to Figure 5.1, the confusion matrices corresponding to each window size are presented in Figures 5.2a-5.2e. Unlike the accuracy and loss curve in Figure 5.1, the confusion matrices provide a clearer picture of the difference in performance for the different configurations from an accuracy perspective. The accuracy values in Figure 5.1 are based on the metric described in Table 3.2, which outlines how accuracy is calculated in this study. Similarly, the loss values shown in the figure correspond to the cross-entropy loss function.

Analyzing the matrices, it becomes evident that the window size used in the third run achieves the most optimal configuration for the current training process. The configuration used in run 3 yields the best values for both false positives and false negatives, while maintaining the best values for true positives and true negatives. Consequently, it corresponds to the highest accuracy among the various configurations.

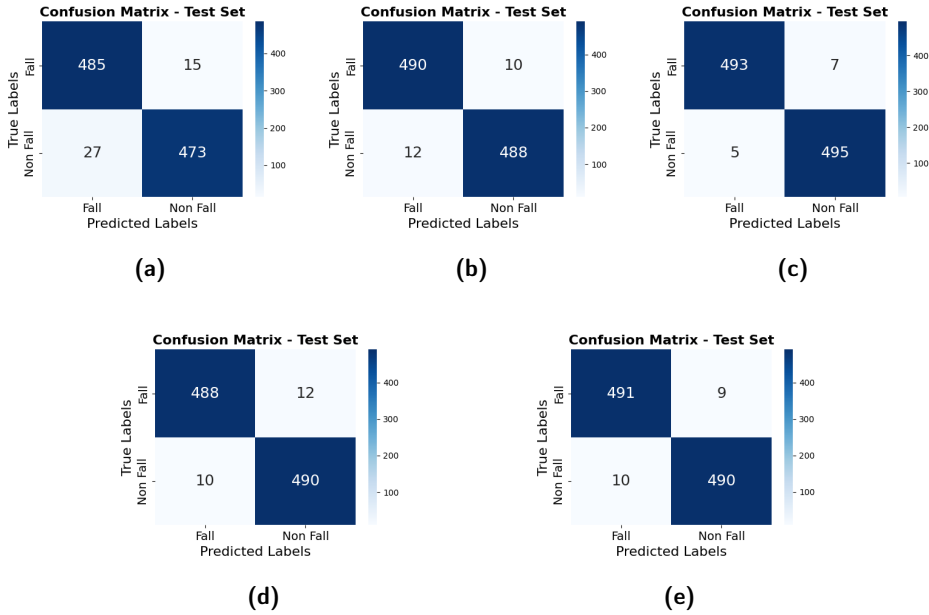


Figure 5.2: Confusion matrices for the centralized model on test data using different window sizes: (a) 1.5s, (b) 2.0s, (c) 3.0s, (d) 4.0s, and (e) 5.0s.

The performance metrics for the centralized implementation are presented in Table 5.2. The evaluated metrics include accuracy, precision, recall, and the F1-score. Table 5.2 also further indicates that a window size of 3 seconds is the most suitable for the centralized model since this configuration presents the best values of the performance metrics. This configuration is also the basis for the main result of the centralized implementation. For the purpose of comparison, the performance metrics for the other window sizes are also presented in the table.

The centralized model achieves an accuracy of 98.8% (0.988), a precision of 98.6% (0.986), a recall value of 99.0% (0.990), and an F1-score of 98.8% (0.988). Considering these results, it can be concluded that the centralized model performs exceptionally well. The implementation can accurately determine when a fall has occurred and distinguish between falls and daily activities with high certainty.

| Window Size: | Accuracy: | Precision: | Recall: | F1-Score: |
|---------------------|------------------|-------------------|----------------|------------------|
| 1.5s | 0.958 | 0.969 | 0.946 | 0.957 |
| 2.0s | 0.972 | 0.959 | 0.986 | 0.972 |
| 3.0s | 0.988 | 0.986 | 0.990 | 0.988 |
| 4.0s | 0.978 | 0.976 | 0.98 | 0.978 |
| 5.0s | 0.981 | 0.981 | 0.98 | 0.980 |

Table 5.2: Performance of the centralized model for different window sizes with a 50% overlap.

5.1.2 Performance of the Federated Implementation

Figures 5.3a-5.3e below illustrates the accuracy and loss curves for the federated implementation where the subjects were grouped into groupings of 12 subjects per group. The training process was conducted across five distinct configurations, varying in the number of server rounds and the number of local epochs. In federated learning, server rounds refer to the number of communication cycles between the central server and client devices, where model updates are aggregated. Local epochs represent how many training passes each client performs on their local data before sending updates back to the server. These five configurations are presented in Table 5.3 and were chosen to explore the influence of the choices of number of server rounds and local epochs on the training process.

| Configuration: | Server Rounds: | Local Epochs: |
|-----------------------|-----------------------|----------------------|
| Conf. 1 | 3 | 50 |
| Conf. 2 | 6 | 25 |
| Conf. 3 | 15 | 10 |
| Conf. 4 | 30 | 5 |
| Conf. 5 | 150 | 1 |

Table 5.3: Training configurations for federated learning

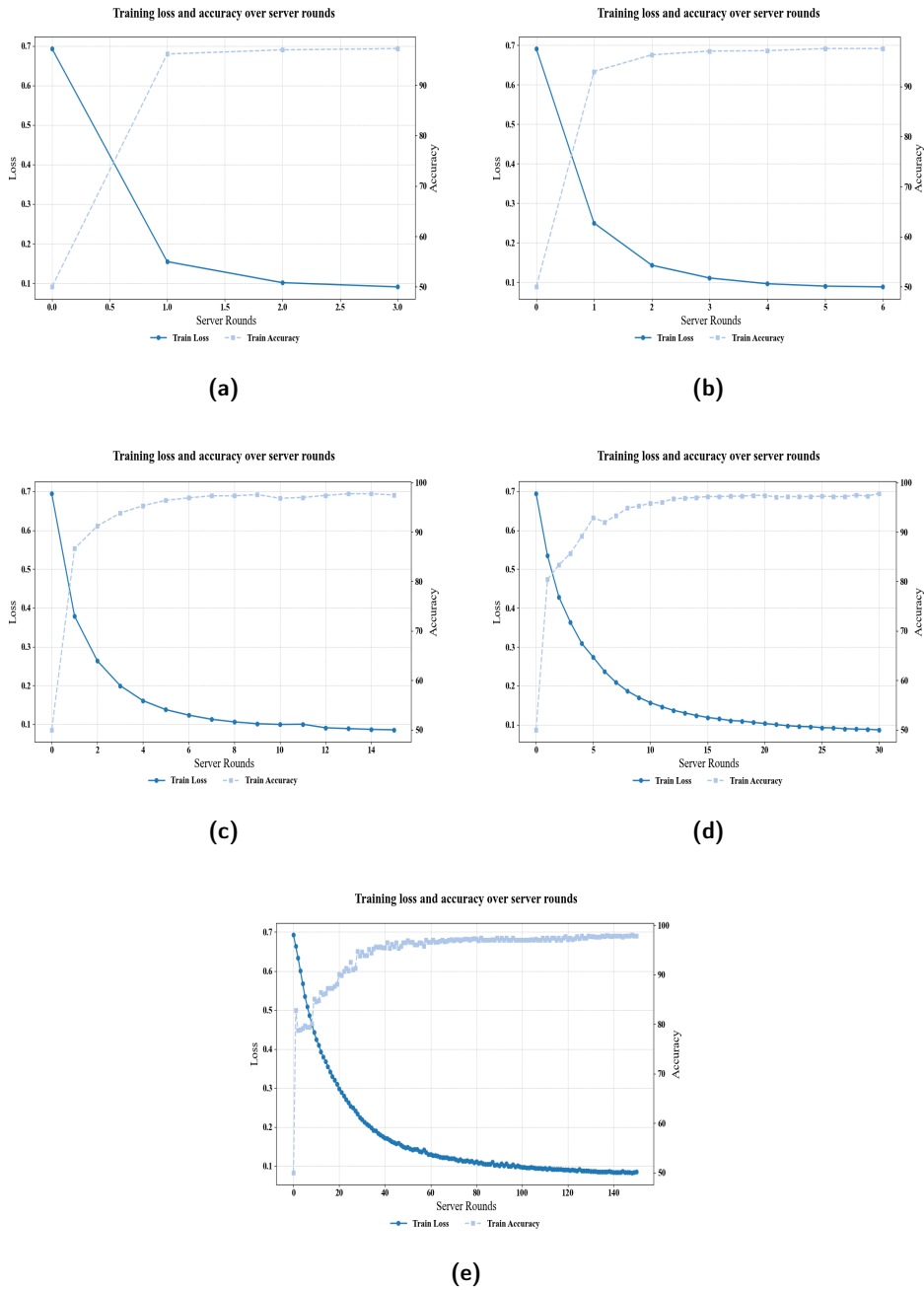


Figure 5.3: Accuracy and loss for the federated training process across different configurations: (a) Conf. 1, (b) Conf. 2, (c) Conf. 3, (d) Conf. 4, and (e) Conf. 5.

Comparing the influence of the training configuration can be considered a complex comparison by only reviewing the accuracy and loss curves presented in Figure 5.3a-5.3e. To aid in this comparison, the confusion matrices for configuration 1-5 are therefore presented in Figure 5.4a-5.4e below. Analyzing the matrices reveals that configuration 4 and 5 corresponds to the most optimal configurations for the federated implementation. This is evident since configurations 4 and 5 achieves the best values for false positives and false negatives as well as true positives and true negatives.

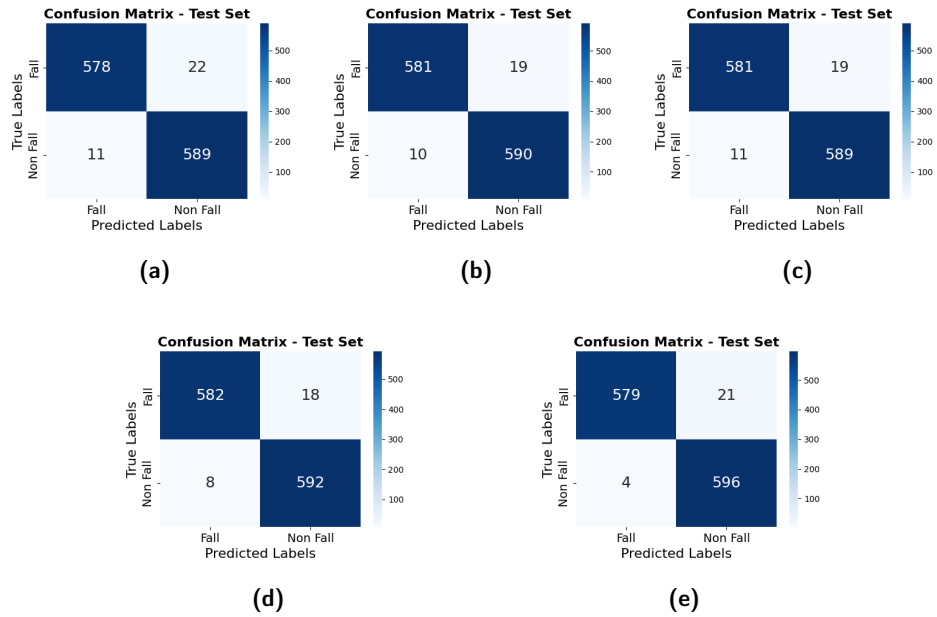


Figure 5.4: Confusion matrices for the federated model on test data using different configurations: (a) Conf. 1, (b) Conf. 2, (c) Conf. 3, (d) Conf. 4, and (e) Conf. 5.

The performance metrics for the federated implementation when subject grouping was used are presented in Table 5.4. Similar to the centralized implementation, these metrics include accuracy, precision, recall, and F1-score. The table presents these metrics in relation to the various configurations of server rounds and local epochs. Additionally, this result presentation further indicates that configurations 4 and 5 are the most suitable for the federated implementation. This is indicated in that these two configurations provides the highest values of F1-score compared to the other configurations.

Configuration 4 and 5 both achieved an F1-score of 97.9% (0.979). Although the F1-scores are the same, the performance-related result for the federated implementation where subjects were grouped is based on configuration 4. This is because configuration 4 has a more balanced distribution of server rounds and local epochs, which is preferable for federated learning. As a result, the federated implementation with subject grouping achieves an accuracy of 97.8% (0.978), a

precision of 97.0% (0.970), a recall value of 98.2% (0.982), and an F1-score of 97.9% (0.979). Considering these results, it can be concluded that the federated implementation with subject grouping is well-designed to identify falls and distinguish between falls and daily activities with high certainty.

| Server Rounds: | Local Epochs: | Accuracy: | Precision: | Recall: | F1-Score: |
|----------------|---------------|--------------|--------------|--------------|--------------|
| 3 | 50 | 0.973 | 0.964 | 0.982 | 0.973 |
| 6 | 25 | 0.976 | 0.969 | 0.983 | 0.976 |
| 15 | 10 | 0.975 | 0.969 | 0.982 | 0.975 |
| 30 | 5 | 0.978 | 0.970 | 0.982 | 0.979 |
| 150 | 1 | 0.979 | 0.966 | 0.993 | 0.979 |

Table 5.4: Performance of the federated model for different training configurations. Bold values highlight results from Configuration 4 (30 server rounds, 5 local epochs).

In contrast to the setup with subject grouping, an evaluation was also conducted without subject grouping, i.e. each client corresponds to an individual subject in the dataset. For this evaluation, the same configuration, Conf. 4, of number of server rounds and local epochs was used as for the primary result for the evaluation with subject grouping. Figure 5.5 below illustrates the accuracy and loss curve for this evaluation and as a compliment, Figure 5.6, presents the confusion matrix for the evaluation.

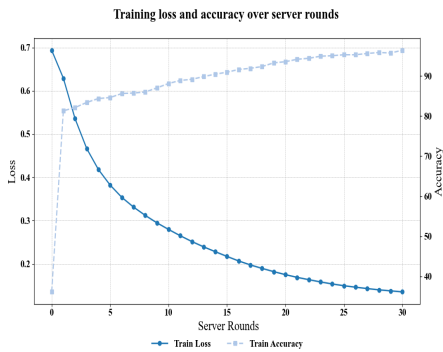


Figure 5.5: Accuracy and loss for the federated training process using Conf. 4 and no subject grouping

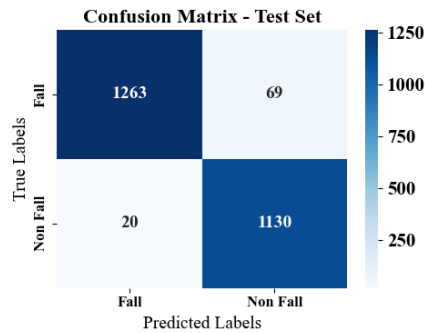


Figure 5.6: Confusion matrix for FL using Conf. 4 and no subject grouping

Table 5.5 below presents the performance-related results for the model where each client corresponds to an individual subject in the dataset. The federated

implementation without subject grouping achieves an accuracy of 96.4% (0.964), a precision of 94.2% (0.942), a recall value of 98.3% (0.983), and an F1-score of 96.2% (0.962). Considering these results, it can be concluded that the federated implementation is well-designed to identify falls and distinguish between falls and daily activities, even without subject grouping. However, its performance slightly deteriorates when grouping not is used.

| Server Rounds: | Local Epochs: | Accuracy: | Precision: | Recall: | F1- Score: |
|-------------------|------------------|-----------|------------|---------|---------------|
| 30 | 5 | 0.964 | 0.942 | 0.983 | 0.962 |

Table 5.5: Performance metrics of the federated model using 30 server rounds, 5 local epochs and no subject grouping.

5.1.3 Performance Comparison

When comparing centralized and federated implementations, it is evident that the centralized approach excels over the federated in performance. The centralized model can identify falls and distinguish them from daily activities with greater certainty than the federated model. To illustrate the differences between the two models, Table 5.6 below presents the performance evaluation results for both models when subject grouping is used, and Table 5.7 presents the difference without subject grouping. In both of these tables, the difference illustrates how much lower the performance of the federated model is compared to the centralized model.

The results presented for the setup with subject grouping are the two main results from both the centralized and federated models. For the centralized implementation, this corresponds to the model trained with the window size of 3.0 seconds. In the case of the federated implementation, the results correspond to the model trained with 30 server rounds and five local epochs. The table reveals that the centralized model outperforms the federated one in all four performance metrics. However, it is worth noting that the disparity between the two implementations is relatively small, considering the training methods and the access to data during the learning processes.

When subject grouping is not used, the performance of the federated model declines further. Compared to both the centralized model and the grouped federated setup, the single setup achieves lower accuracy, precision, recall, and F1-score. This indicates that subject grouping helps mitigate some of the performance loss associated with federated learning. However, even without grouping, the federated approach remains a viable alternative, particularly in scenarios where privacy constraints limit access to centralized data.

| Implementation: | Accuracy: | Precision: | Recall: | F1-Score: |
|------------------------|------------------|-------------------|----------------|------------------|
| Centralized | 0.988 | 0.986 | 0.990 | 0.988 |
| Fed. Grouped | 0.978 | 0.970 | 0.982 | 0.979 |
| Difference | 1,01% | 1,62% | 0,81% | 0,91% |

Table 5.6: Comparison of performance results of the centralized and federated model with subject grouping.

| Implementation: | Accuracy: | Precision: | Recall: | F1-Score: |
|------------------------|------------------|-------------------|----------------|------------------|
| Centralized | 0.988 | 0.986 | 0.990 | 0.988 |
| Fed. Single | 0.964 | 0.942 | 0.983 | 0.962 |
| Difference | 2,43% | 4,46% | 0,71% | 2,63% |

Table 5.7: Comparison of performance results of the centralized and federated model without subject grouping.

5.2 Privacy Evaluation

Presented below are the results related to the privacy evaluation of the experimental study. The results are divided into two subsections containing the results for the centralized and federated implementation respectively. Also included is a subsection for comparison of the privacy related results of the two implementations.

5.2.1 Privacy Evaluation of the Centralized Implementation

The privacy evaluation for the centralized implementation yields two distinct metrics as presented in Table 5.8. The first observation is that the shared data consists of uncompressed motion data. In essence, all the collected data is shared in the form of uncompressed accelerometer, gyroscope, and orientation data. Furthermore, the shared data was measured to have an average transfer size of 28.96 MB.

| Aspect: | Metric: |
|--------------------------|--------------------------|
| Shared data type | Uncompressed motion data |
| Size of transferred data | 28.96 MB (average) |

Table 5.8: Privacy metrics for the centralized model.

5.2.2 Privacy Evaluation of the Federated Implementation

Similar to the centralized implementation, the federated implementation yields two distinct metrics for the privacy evaluation as presented in Table 5.9. The shared data between client and server consists of model parameters. Indicating that the collected data is confined to the distributed device and only model parameters are shared. The size of the shared model parameters amounts to 1.28 MB.

| Aspect: | Metric: |
|--------------------------|------------------|
| Shared data type | Model parameters |
| Size of transferred data | 1.28 MB |

Table 5.9: Privacy metrics for the federated model.

5.2.3 Privacy Comparison

As illustrated in Table 5.10 below, the centralized and federated implementations differ significantly in their privacy approaches. Comparing the two implementations, a first observation is that the centralized implementation shares uncompressed motion data, while the federated implementation shares model parameters. This implies that the centralized implementation offers significantly inferior privacy protection compared to the federated implementation. Since the federated implementation only shares the model parameters, it provides a fundamental privacy protection by ensuring that the uncompressed motion data remains confined to the distributed device. Moreover, the sharing of model parameters instead of motion data implies that more sophisticated methods are needed to access sensitive data in the federated case.

Another observation is that the amount of data shared varies significantly between the federated and centralized implementations. While this may be expected given the type of data shared in the two different approaches, it also suggests that the federated implementation provides better privacy protections. Regardless of how data is shared and what data is shared, an increased amount of data transferred poses a security risk when it comes to privacy.

| Implementation: | Data type: | Data size: |
|------------------------|--------------------------|-------------------|
| Centralized | Uncompressed motion data | 28.96 MB |
| Federated | Model parameters | 1.28 MB |
| Difference | – | 27,68 MB |

Table 5.10: Comparison of data sharing between centralized and federated implementations, showing the type and size of data transmitted during model training. The significant reduction in data size (27.68 MB) for the federated approach represents enhanced privacy protection.

Discussion and Future Work

This chapter discusses the findings of the experimental study, exploring their implications within the broader research context and in relation to previous research within the field. The discussion initially focuses on the privacy impact of federated learning compared to centralized learning, considering how data security and user privacy are affected. It then evaluates performance and accuracy, assessing whether federated learning introduces trade-offs in fall detection effectiveness. Additionally, the chapter explores the integration of federated learning with an IoT platform, discussing the practical benefits and challenges associated with deploying such a system in real-world applications. The chapter then concludes by presenting possible areas for future work.

6.1 Privacy Impacts of Federated vs. Centralized Learning for Fall Detection

Federated learning has emerged as a promising approach to machine learning, particularly in privacy-sensitive applications such as healthcare and personal activity tracking. Unlike centralized machine learning, which requires raw data to be transmitted to a central server, federated learning enables model training to occur locally on edge devices, only sharing model parameters instead of raw data. This fundamental difference in data handling strongly relates to the first research question of this thesis: *How does federated machine learning for fall detection impact privacy preservation compared to centralized machine learning?*

The results presented in Section 5.2 of this report indicate that federated learning offers a clear advantage in preserving the privacy of data when compared to centralized learning. While the centralized implementation shares uncompressed motion data (accelerometer, gyroscope, and orientation data) directly, the federated implementation only shares model parameters. This in itself significantly reduces the risk of exposure due to data breaches and unauthorized access. These results are also in line with previous research in the field, which also demonstrates an advantage for federated learning when it comes to strengthening privacy and mitigating the risks of handling sensitive data. However, previous research has mainly focused its studies on combining federated learning with additional methods for privacy preservation. The results of this project, however, show that federated learning ensures fundamental privacy protection compared to centralized

learning in a fall detection system, which is a significant observation for cases and applications where additional protection mechanisms may not be feasible.

Another important observation from the experimental study is that the federated and centralized implementations showed substantial variation in the size of the shared data. In the centralized approach, as previously mentioned, the shared data consisted of raw motion data, which typically entails high-dimensional and uncompressed data. This suggests that the method may be more vulnerable to privacy attacks and potential reconstruction by a third party. Moreover, the larger amount of shared data increases the likelihood and risk of accidental data leakage. In contrast, the federated implementation shared significantly less data. A plausible explanation for this difference lies in the fact that the two methods share two different types of data. However, it can be concluded that while the federated implementation enhances privacy preservation by sharing model parameters, the data size remains a significant factor. With a reduction of shared data, the risk of data leakage and potential data breaches is reduced. This is because substantially less data is exposed in the federated scenario compared to the centralized one.

Although federated learning in this study appears to be significantly more trustworthy from a privacy standpoint compared to the centralized approach, there is still room for improvement. In this project, the focus was put on comparing federated learning to centralized learning without incorporating any additional privacy protection measures. To enhance the study's depth, a further exploration of potential vulnerabilities in both methods could be made. For instance, the impact of common privacy attacks could be explored and the effects of introducing additional privacy protections could be examined. By conducting such a study, a more comprehensive conclusion regarding the privacy advantages of federated learning could be drawn in comparison to the centralized implementation.

An additional aspect worth considering is the impact of federated and centralized learning from an economic and energy consumption perspective. As the results of the project demonstrated, the amount of shared data varied significantly between the two machine learning approaches. As previously mentioned, federated learning resulted in reduced bandwidth usage compared to centralized learning. This may seem appealing for IoT applications where battery consumption on edge devices is crucial. However, federated learning may require more communication rounds to achieve convergence, potentially leading to increased costs, both in terms of energy consumption and data transmission. This aspect was not factored into the project due to its time constraints but should be taken into account for various IoT use cases involving federated learning.

In conclusion, the findings of this study provide evidence that federated learning enhances privacy in fall detection applications compared to centralized learning. Federated learning minimizes exposure to sensitive motion data by sharing only aggregated model parameters, unlike centralized learning, which transmits raw motion data. This reduces the risk of data breaches, privacy attacks, and accidental leaks. Additionally, the reduced data volume in federated learning reduces the chances of data reconstruction attacks. While previous research often combines federated learning with other privacy techniques, this study highlights that federated learning alone offers superior privacy protection compared to centralized learning.

However, federated learning faces challenges, such as potential privacy attacks and increased communication overhead, which were not explored in this study. Future research should address these vulnerabilities, as well as the impact on energy efficiency and computational costs for edge devices. Despite these challenges, the results suggest that federated learning is a viable privacy-preserving solution for fall detection and could serve as a foundation for future improvements in privacy-sensitive machine learning applications.

6.2 Performance and Accuracy Implications of Federated vs. Centralized Learning for Fall Detection

In addition to the potential privacy-preserving advantages of federated learning over centralized learning, this project also aimed to evaluate the two approaches from a performance and accuracy perspective. This aspect holds great significance when considering the adoption of federated learning over centralized learning to capitalize on its privacy-preserving potential. Maintaining high accuracy in identifying falls is of significant importance and cannot be compromised. This aspect directly aligns with the project's second research question: *Is there any significant difference in performance and accuracy of detecting falls when comparing federated and centralized learning?*

As evident from the performance-related results presented in Section 5.1, the centralized implementation outperforms the federated implementation in efficiently and accurately identifying falls and distinguishing them from daily activities. However, the difference in accuracy and performance is marginal, with the federated model achieving only 1.01% lower accuracy and a 0.91% lower F1-score in the grouped setting than the centralized model. For the non-grouped setting the federated model achieved slightly larger difference between the two approaches with a accuracy difference of 2.43% and difference in F1-score of 2.63%. One possible explanation for the lower performance in the single-subject setup could be the dataset size. With fewer data points per client, the model may struggle to generalize well, leading to reduced accuracy and performance. Consequently, the experimental study suggests that federated learning can be a practical alternative to the more traditional centralized learning approach for a fall detection use case. The results also suggests that the grouped setting performs better than when each client correlated to a single subject. Maintaining high performance and accuracy is crucial in the use case of fall detection, as incorrect fall labeling can have severe consequences. A false positive, where a non-fall is mistaken for a fall, may not be the worst outcome, but a false negative, where a fall is mistaken for a non-fall, can have devastating effects. Although the two approaches have slight variations in accuracy and performance, the study showed that the risks of mislabeling are remarkably low between the two implementations.

A possible explanation for the marginal difference in performance and accuracy could be related to the dataset used in the experimental study. As discussed in the report, the use of public datasets can have its limitations. One such limitation could be the size constraint of the dataset. The chosen dataset was suitable for implementation but had a size limitation. This might have affected the results of the

two implementations. Training processes lacked access to a larger and more varied dataset, which might have affected the learning process negatively. Generally, machine learning models benefit from being trained on larger datasets with more complex and varied data, leading to increased accuracy. A larger dataset could also significantly impact federated learning results. A larger and more complex dataset might provide each federated client with greater data variation, influencing the aggregation of the global model.

When comparing the study's performance and accuracy-based results with previous research, several key differences must be considered. Previous studies in activity and fall detection using machine learning and federated learning frequently integrate additional privacy-preserving methods, such as differential privacy, to enhance data protection. This study, however, focused on directly comparing federated and centralized learning for fall detection without such techniques, creating a distinct difference from prior work.

Notably, the observed performance gap between federated and centralized learning of this study aligns with previous findings, which consistently report that centralized learning generally outperforms federated learning. However, prior studies generally indicate a larger discrepancy. For instance, Sozinov et al. [17] found that federated learning resulted in lower accuracy for human activity recognition compared to centralized learning. In the study, the federated model presented an accuracy of 89% compared to the centralized model with an accuracy of 93%, reinforcing the notion that federated models tend to underperform. Similarly, Fauzi et al. [24] demonstrated a substantial accuracy drop when using federated learning for smartwatch-based stress detection. This study presented an average accuracy of 93.55% and an average F1-score of 87.83% for the centralized implementation. In contrast, the results for the federated implementation were substantially lower, with an average accuracy and F1-score of 85.75% and 63.39% respectively, further supporting the argument that the gap between the two approaches is often significant.

Comparing this study's accuracy and performance results with previous research that incorporates privacy-preserving methods in federated learning can in many ways be considered a complex task. The performance and accuracy of federated learning models in other studies are often affected by the addition of such methods, which may degrade model performance. For instance, previous research, such as Mohammadi et al. [18], has shown that differential privacy techniques can further reduce federated learning accuracy. While this study's primary objective was to evaluate differential privacy and showed promising results for the addition of differential privacy, it also showed that the accuracy of the model decreased in comparison to the case where no differential privacy was used. This suggests that the larger performance gap observed in prior studies may partly be due to the inclusion of additional privacy mechanisms.

A significant aspect of this study is that the federated implementation was simulated rather than deployed in a real-world distributed setting. While an actual deployment would have been ideal, it was beyond the project's scope. However, this presents an opportunity for future research to evaluate federated learning performance under practical conditions. Factors such as network constraints, device heterogeneity, data distribution imbalances, and potential data loss were not

accounted for in this study but would be crucial in a real implementation. Investigating these aspects could provide further insights into federated learning's viability for fall detection in real-world applications.

In summary, despite its limitations, this study demonstrates that federated learning can achieve nearly the same accuracy as centralized learning for fall detection, with only a marginal performance loss. While previous research often reports a larger performance gap, the results suggest that federated learning alone can be a practical alternative to centralized learning under the right conditions. However, further research is needed to explore federated learning's performance on larger datasets, its scalability in real-world deployments, and the trade-offs introduced by additional privacy-preserving techniques.

6.3 Suggested Federated Learning Integration for IoT Fall Detection: Feasibility and Challenges

In addition to the primary purpose of this thesis, the project also aimed to explore how a federated machine learning approach for fall detection could be integrated with an IoT platform. This discussion can therefore be seen as a first step towards a complete fall detection system and answers the project's third research question: ***How can a federated machine learning approach be integrated with an IoT platform, and what benefits does this hold?*** To do this, a proposal is first raised on how an integration with an IoT platform could be realized, outlining the benefits of such an integration, followed by a discussion of the challenges it may entail. The integration is specifically explored within the IoT platform Yggio [52], as this thesis is conducted in collaboration with Sensative, the company developing the platform, ensuring that the proposal aligns with Yggio's existing architecture and capabilities.

In a federated learning-based fall detection system, each connected device, such as a wearable sensor, smartphone, or IoT-enabled fall detection device, would process data locally and make inferences. Unlike traditional machine learning approaches that transmit raw sensor data to a centralized server, each device would securely store its sensitive data and only share crucial status updates with the IoT platform. For instance, with an IoT platform like Yggio, all sensitive sensor data would remain locally on the connected device, while Yggio could serve as a central hub to monitor the status of the devices while maintaining data integrity. Figure 6.1 illustrates how this approach provides a clear overview and control over the connected devices within the fall detection ecosystem. By displaying only non-sensitive metadata, healthcare personnel can easily identify the location and time of a fall occurrence and clearly see if any device has lost connection to the system.

To ensure a well integrated fall detection system, the integration could leverage several key features already present in Yggio. One crucial aspect is real-time alerts and notifications. When a fall is detected by an edge device, the event could be instantly logged and transmitted to Yggio, where it appears in the activity log and notifies the user. In Yggio, this could be illustrated in a activity log including timestamps, device IDs, and priority levels as illustrated in Figure 6.1. By expanding this system, Yggio could introduce automatic alerts that notify caregivers

or emergency responders the moment a fall incident occurs. Acknowledgment features, such as the “Acknowledge” button seen in the interface, could ensure that critical alerts are actively managed and do not go unnoticed.

In addition to logging fall events, location-based tracking could further enhance response efficiency. Since Yggio already integrates mapping functionality, devices detecting falls could transmit location metadata, allowing incidents to be visually represented on the map. As seen in the "Map Overview" in Figure 6.1, falls could be highlighted in red, distinguishing them from active devices, which remain green when connected. This approach would provide immediate situational awareness, helping responders quickly assess where an incident has occurred and enabling faster assistance.

Beyond detecting falls, continuous device status monitoring is another key feature that the integration could facilitate. Device status monitoring would be an essential feature in ensuring that all devices remain connected and fully functional. Each device could periodically send status updates to confirm its operational state, allowing for proactive maintenance and reducing the risk of undetected failures. The "Devices" section in the dashboard, presented in Figure 6.1, could display real-time statuses, making it easy to identify any issues. If a device were to lose connection or experience a malfunction, it could result in severe consequences, as falls may go unnoticed, delaying critical assistance. By integrating this functionality, Yggio could provide a more reliable and fail-safe monitoring system, ensuring that every device remains active and responsive without compromising user privacy.

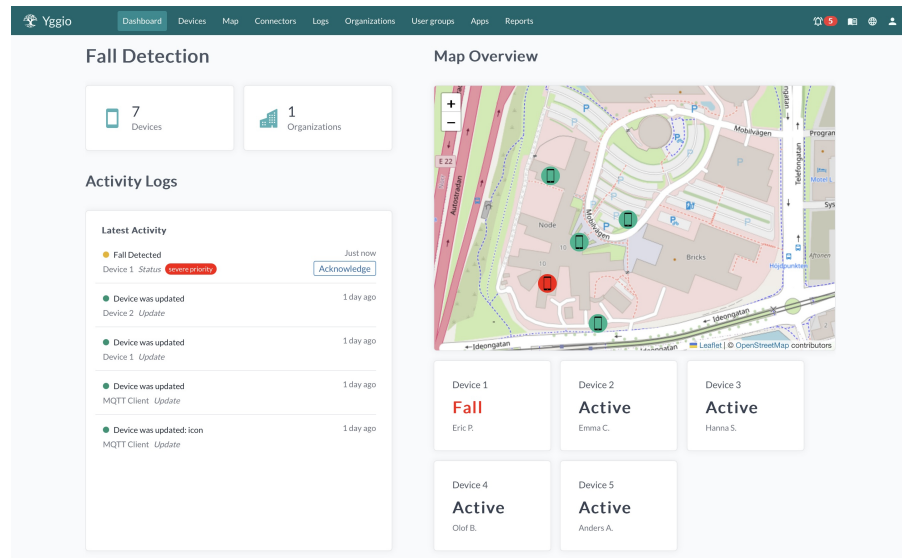


Figure 6.1: Fall detection dashboard in Yggio

By leveraging Yggio's existing capabilities, the proposed integration outlines how a federated machine learning-based fall detection system could be effectively incorporated into an IoT platform. Through real-time notifications, location-based tracking, and continuous device status monitoring, the system would enhance both the reliability and responsiveness of fall detection, ensuring that incidents are promptly identified and addressed. Furthermore, by keeping sensitive sensor data locally on each device while sharing only critical status updates with Yggio, this approach balances privacy, security, and functionality within the ecosystem.

However, while the proposed integration offers clear benefits, its implementation comes with several challenges that must be carefully addressed. One of the most critical aspects is ensuring the reliable streaming of real-time data. In a system like the one proposed, it is essential that data is transmitted securely and accurately, while also making sure that only the necessary information is shared. Non-sensitive metadata, such as device status and location, should be sent to the IoT platform, whereas sensitive sensor data must remain securely stored on the device itself. Furthermore, low-latency transmission is crucial as any delay in reporting a fall could have severe consequences.

Closely tied to real-time data streaming is the challenge of labeling data within the system. In machine learning, data labeling plays a fundamental role in the training process, particularly when using the supervised learning method. The implementation presented in this project is based on supervised learning, where all data is pre-labeled. However, in a real-time system, this approach becomes problematic, as data must be processed and classified on the device as events occur. To overcome this, alternative methods could be explored, such as unsupervised learning or anomaly detection models that identify unusual patterns without requiring labeled data. Another possible solution is an active learning approach, where users are prompted to confirm or reject detected falls. A way of doing this could be to prompt the user of the IoT platform to confirm or deny whether a reported fall was true or a false alarm. This feedback loop would allow the model to improve over time, refining its accuracy as real-world data is continuously validated.

In addition to these challenges, network reliability is another key factor that must be considered. If a device loses connection to the system, it would no longer be able to report falls, which could lead to critical incidents going unnoticed. To mitigate this risk, a mechanism could be implemented that notifies the system when a device disconnects or experiences a malfunction. By treating connectivity loss with the same urgency as a fall detection event, the system could ensure that technical issues are promptly addressed, reducing the likelihood of missed alerts.

In conclusion, integrating a federated machine learning-based fall detection system with an IoT platform presents clear benefits and advantages in terms of privacy, reliability, and responsiveness. By leveraging Yggio's capabilities, the proposed approach ensures that sensitive sensor data remains local while enabling real-time notifications, location tracking, and device status monitoring. However, successful implementation requires addressing key challenges such as reliable real-time data streaming, effective data labeling, and network stability. Despite these challenges, the proposed integration provides a solid foundation for a privacy-preserving and scalable fall detection system, contributing to the broader goal of enhancing safety in IoT-enabled environments.

6.4 Future Work

While this thesis provides valuable insights into the privacy and performance trade-offs of federated learning for fall detection, several key areas remain unexplored. Future research can build upon these findings to address limitations, improve system performance, and enhance privacy protection. Below are specific suggestions for future work and their significance.

6.4.1 Privacy-Enhancing Techniques and Performance Optimization

An interesting area for future work is to further investigate how different privacy-enhancing techniques can strengthen the privacy protection in federated learning for fall detection. While these methods reduce the risk of privacy violations, they can also introduce challenges in the form of increased computational complexity and communication costs.

A possible future study could therefore focus on analyzing these trade-offs and developing optimization strategies that maintain a high level of privacy protection without significantly affecting model performance or system resource consumption. Of particular interest would be to investigate how federated learning can be made more resilient to attacks such as reconstruction attacks and membership inference, while preserving efficiency. Such an analysis would contribute to the development of more robust and resource-efficient machine learning systems with strong integrity guarantees.

6.4.2 Scalability and Handling of Real-Time Data Streaming

A direction of great importance could be to investigate how federated learning performs when using larger and more varied datasets. This study used a public dataset processed in an offline manner, but in practical applications, datasets can be significantly larger, more heterogeneous, and collected in real-time. Gathering and utilizing real-time data would add substantial value to future research as it would better reflect actual deployment conditions. Larger datasets can improve the generalization ability of the model, but at the same time result in increased demands on computational resources and communication, which can affect the efficiency and scalability of the system.

Furthermore, handling of data streaming is a central challenge for real-time applications, such as fall detection from sensor systems. Applying federated learning to real-time data requires a deeper understanding of factors such as latency, synchronization between clients, and the ability of the model to adapt to changing data conditions. Future studies can therefore focus on developing methods to effectively integrate streaming data into federated fall detection systems, which would enable more responsive and dynamic applications. Moving beyond the offline approach used in this thesis to incorporate real-time data collection and processing would represent a significant advancement in creating practical and deployable federated learning systems for fall detection.

6.4.3 Economic and Energy Aspects of Federated Fall Detection

The economic and energy implications of federated learning can also be considered an important aspect for future work. One of the advantages of federated learning is the reduction of the need to transfer large amounts of data over the network, which can potentially reduce bandwidth costs. At the same time, the increased amount of local computations and communication rounds can lead to higher energy consumption and increased system costs.

For IoT systems, where devices are often battery-powered, this trade-off can be crucial for practical implementation. Future research should therefore focus on analyzing the balance between reduced network traffic and increased local computational load. A cost analysis that includes both network costs and energy consumption would provide valuable insight into how federated learning can be optimized to be more sustainable and resource-efficient, especially in resource-constrained environments.

This thesis has investigated how federated machine learning impacts fall detection in IoT-based applications, with a particular focus on the comparison between federated learning and traditional centralized machine learning from an integrity and performance perspective. The study has been guided by the primary research question: *What is the impact of federated machine learning for fall detection from a privacy preserving and accuracy perspective compared to centralized machine learning?*

The experimental results of the thesis show that federated machine learning offers significant advantages in terms of privacy protection. Unlike centralized learning, where raw data must be transferred to a central server, federated learning enables local model training on users' devices, where only model parameters are shared. This reduces the risk of data leaks and strengthens user privacy, making federated learning a promising alternative for privacy-sensitive applications such as healthcare. At the same time, federated learning may result in increased computational and communication costs, which are challenges that need to be addressed.

In terms of performance, the results of this study show that federated learning can achieve an accuracy comparable to centralized learning in fall detection. However, careful trade-offs of federated learning-specific parameters, such as the number of server rounds and local training cycles, are required to balance training speed and model convergence. The study also points out the importance of data distribution between clients, as non-IID data (non-independent and non-uniformly distributed data) can degrade the generalization ability of the model.

Furthermore, the thesis highlights both the opportunities and challenges of integrating federated learning into an IoT platform. Federated learning fits well in decentralized IoT systems, but real-time data management, energy efficiency, and system latency remain challenges that need to be addressed to enable practical use.

This thesis contributes to the growing body of research in privacy-preserving machine learning by providing empirical evidence on the trade-offs required between privacy and performance when using federated learning for fall detection. The results strengthen the potential of federated learning as a viable alternative to centralized learning in scenarios where data privacy is of the highest priority. Furthermore, the study provides insights into how federated learning can be integrated into IoT platforms and highlights the need for further optimization of

systems to handle real-time data and energy consumption.

While this thesis provides a solid foundation, there are several possible future research areas. Further studies could explore advanced integrity-preserving techniques to further strengthen the integrity protection in federated learning. In addition, research into improved scalability and resource efficiency in federated learning, especially for IoT devices with limited resources, would be valuable. Finally, practical implementation and testing of federated learning-based fall detection in real-world environments, rather than simulations, would provide deeper insights into its practical applicability and effectiveness.

In conclusion, this thesis shows that federated machine learning is a promising approach for fall detection, as it provides increased privacy without significantly reducing accuracy and performance. However, issues around system optimization and reality adaptation must be resolved for federated learning to become a widely used solution in healthcare and IoT applications.

References

- [1] World Health Organization. *Falls*. Accessed: 2024-10-31. 2021. URL: <https://www.who.int/news-room/fact-sheets/detail/falls>.
- [2] Rakesh Shrestha et al. “Anomaly detection based on LSTM and autoencoders using federated learning in smart electric grid”. In: *Journal of Parallel and Distributed Computing* 193 (2024), p. 104951. ISSN: 0743-7315. DOI: <https://doi.org/10.1016/j.jpdc.2024.104951>. URL: <https://www.sciencedirect.com/science/article/pii/S0743731524001151>.
- [3] RISE - Research Institutes of Sweden. *SID: Secure data sharing for Industrial Digitalization*. Accessed: 2025-01-16. 2025. URL: <https://www.ri.se/en/what-we-do/projects/sid-secure-data-sharing-for-industrial-digitalization>.
- [4] A R Vaishnavi, Sahana G, and N Guruprasad. “Wearable Devices in the IoT: A Security and Privacy Perspective”. In: *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*. 2023, pp. 1–4. DOI: 10.1109/ICICAT57735.2023.10263654.
- [5] Zulfiqar Ali Solangi et al. “The future of data privacy and security concerns in Internet of Things”. In: *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*. 2018, pp. 1–4. DOI: 10.1109/ICIRD.2018.8376320.
- [6] Lennart Kraft, Bernd Skiera, and Tim Koschella. “Economic Impact of Opt-in versus Opt-out Requirements for Personal Data Usage: The Case of Apple’s App Tracking Transparency (ATT)”. In: *IEEE Access* (Oct. 2023).
- [7] Shaista Ashraf Farooqi, Aedah Abd Rahman, and Amna Saad. “Differential Privacy Based Federated Learning Techniques in IoMT: A Review”. In: *2024 18th International Conference on Ubiquitous Infor-*

- mation Management and Communication (IMCOM)*. 2024, pp. 1–7. DOI: 10.1109/IMCOM60618.2024.10418361.
- [8] MIT Sloan School of Management. “Machine Learning, Explained”. In: *MIT Sloan* (Apr. 2021). Accessed: 2024-10-31. URL: <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>.
- [9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [10] Alexander Jung. *Machine Learning: The Basics*. Springer Singapore, 2022.
- [11] IBM. *Supervised vs. Unsupervised Learning: What’s the Difference?* Accessed: 2024-11-04. 2021. URL: <https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning>.
- [12] Chamith Mawela, Chaouki Ben Issaid, and Mehdi Bennis. “A Web-Based Solution for Federated Learning with LLM-Based Automation”. In: (Aug. 2024). DOI: 10.48550/arXiv.2408.13010.
- [13] Sawsan Abdulrahman et al. “A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond”. In: 8.7 (2021), pp. 5476–5497. DOI: 10.1109/JIOT.2020.3030072.
- [14] B. K. Mathew, J. C. Ng, and J. L. Zerbe. “Using proxies to enable on-device machine learning”. U.S. Patent App. 15 275 355. Jan. 2018.
- [15] Jonathan J.M. Seddon and Wendy L. Currie. “Cloud computing and trans-border health data: Unpacking U.S. and EU healthcare regulation and compliance”. In: *Health Policy and Technology* 2.4 (2013), pp. 229–241. ISSN: 2211-8837. DOI: <https://doi.org/10.1016/j.hlpt.2013.09.003>. URL: <https://www.sciencedirect.com/science/article/pii/S2211883713000622>.
- [16] Eike Petersen et al. “Responsible and Regulatory Conform Machine Learning for Medicine: A Survey of Challenges and Solutions”. In: *IEEE Access* 10 (2022), pp. 58375–58418. DOI: 10.1109/ACCESS.2022.3178382.
- [17] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. “Human Activity Recognition Using Federated Learning”. In: (2018), pp. 1103–1111. DOI: 10.1109/BDCLOUD.2018.00164.
- [18] Mohammadreza Mohammadi et al. “Privacy-preserving Federated Learning System for Fatigue Detection”. In: (2023), pp. 624–629. DOI: 10.1109/CSR57506.2023.10224953.

- [19] H.B. McMahan and D. Ramage. *Federated learning: Collaborative machine learning without centralized training data*. Accessed: 2025-02-05. 2017. URL: <https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/>.
- [20] Muskan Bhasin et al. “A Novel System for Fall Detection in the Elderly”. In: *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*. 2023, pp. 862–866.
- [21] Jinxi Zhang et al. “An Effective Deep Learning Framework for Fall Detection: Model Development and Study Design”. In: *J Med Internet Res* 26 (Aug. 2024), e56750. ISSN: 1438-8871. DOI: 10.2196/56750. URL: <https://doi.org/10.2196/56750>.
- [22] Khalid Nafil et al. “Fall Detection System for Elderly People using IoT and Machine Learning technology”. In: *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 2023, pp. 1–5. DOI: 10.1109/ICECCME57830.2023.10252183.
- [23] Md Wasif Islam Wasi et al. “Machine Learning Algorithm for Fall Classification Using Wearable Device”. In: *2022 IEEE Symposium on Future Telecommunication Technologies (SOFTT)*. 2022, pp. 62–66. DOI: 10.1109/SOFTT56880.2022.10010102.
- [24] Muhammad Fauzi, Bian Yang, and Bernd Blobel. “Comparative Analysis Between Individual, Centralized, and Federated Learning for Smartwatch Based Stress Detection”. In: *Journal of Personalized Medicine* 12 (Sept. 2022), p. 1584. DOI: 10.3390/jpm12101584.
- [25] Yen-Hung Liu et al. “Automatic Fall Risk Detection Based on Imbalanced Data”. In: *IEEE Access* 9 (2021), pp. 163594–163611. DOI: 10.1109/ACCESS.2021.3133297.
- [26] Rahul Jain and Vijay Bhaskar Semwal. “A Novel Feature Extraction Method for Preimpact Fall Detection System Using Deep Learning and Wearable Sensors”. In: *IEEE Sensors Journal* 22.23 (2022), pp. 22943–22951. DOI: 10.1109/JSEN.2022.3213814.
- [27] Nguyen Truong et al. “Privacy preservation in federated learning: An insightful survey from the GDPR perspective”. In: *Computers Security* 110 (2021), p. 102402. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2021.102402>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404821002261>.

- [28] Kai Hu et al. “An overview of implementing security and privacy in federated learning”. In: *Artificial Intelligence Review* 57 (July 2024). DOI: 10.1007/s10462-024-10846-8.
- [29] Angela Sucerquia, Jose Lopez, and J. Vargas-Bonilla. “SisFall: A Fall and Movement Dataset”. In: *Sensors* 17 (Jan. 2017), p. 198. DOI: 10.3390/s17010198.
- [30] Biomedical Informatics eHealth Laboratory. *The MobiFall and MobiAct datasets*. URL: <https://bmi.hmu.gr/the-mobifall-and-mobiact-datasets-2/>.
- [31] George Vavoulas et al. “The MobiFall dataset: An initial evaluation of fall detection algorithms using smartphones”. In: *MobiFall* (2013), pp. 1–4. DOI: 10.1109/BIBE.2013.6701629.
- [32] João Marques and Plinio Moreno. “Online Fall Detection Using Wrist Devices”. In: *Sensors* 23.3 (2023). ISSN: 1424-8220. DOI: 10.3390/s23031146. URL: <https://www.mdpi.com/1424-8220/23/3/1146>.
- [33] João Marques. *WEDA-FALL - Wrist Elderly Daily Activity and Fall Dataset*. URL: <https://github.com/joaojtmarques/WEDA-FALL>.
- [34] E. Stack. “Falls are unintentional: Studying simulations is a waste of faking time”. In: *Journal of Rehabilitation and Assistive Technology Engineering* 4 (2017). Published 2017 Oct 9, p. 2055668317732945. DOI: 10.1177/2055668317732945.
- [35] Fabio Bagalà et al. “Evaluation of accelerometer-based fall detection algorithms on real-world falls”. In: *PLoS One* 7.5 (2012). Epub 2012 May 16, e37062. DOI: 10.1371/journal.pone.0037062. pmid: 12036952. URL: <https://doi.org/10.1371/journal.pone.0037062>.
- [36] Flower Labs GmbH. *Flower - A Friendly Federated AI Framework*. Accessed: 2025-01-28. 2024. URL: <https://flower.ai>.
- [37] NVIDIA Corporation. *NVIDIA FLARE*. Accessed: 2025-01-28. 2025. URL: <https://developer.nvidia.com/flare>.
- [38] The Linux Foundation. *OpenFL: an open source framework for Federated Learning*. Accessed: 2025-01-28. 2023. URL: <https://openfl.io>.
- [39] OpenMined Foundation. *PySyft - Data Science on data you are not allowed to see (GitHub)*. Accessed: 2025-01-28. 2025. URL: <https://github.com/OpenMined/PySyft>.
- [40] IBM Corporation. *IBM Federated Learning*. Accessed: 2025-01-28. 2021. URL: <https://ibmfl.res.ibm.com>.

- [41] TensorFlow Team. *TensorFlow Federated: Machine Learning on Decentralized Data*. Accessed: 2025-01-28. 2024. URL: <https://www.tensorflow.org/federated>.
- [42] Raul Garreta and Guillermo Moncecchi. *Learning scikit-learn: machine learning in python*. Vol. 2013. Packt Publishing Birmingham, 2013.
- [43] Hongkai Chen, Sazia Mahfuz, and Farhana Zulkernine. “Smart Phone Based Human Activity Recognition”. In: (2019), pp. 2525–2532. DOI: 10.1109/BIBM47256.2019.8983009.
- [44] “Fall Detection Using Neural Network Based on Internet of Things Streaming Data”. In: *UHD Journal of Science and Technology* 4 (Nov. 2020), pp. 91–98. DOI: 10.21928/uhdjst.v4n2y2020.pp91-98.
- [45] T. Theodoridis et al. “Human Fall Detection from Acceleration Measurements Using a Recurrent Neural Network”. In: (Nov. 2017), pp. 145–149. DOI: 10.1007/978-981-10-7419-6_25.
- [46] Sazia Mahfuz et al. “Detecting Irregular Patterns in IoT Streaming Data for Fall Detection”. In: (2018), pp. 588–594. DOI: 10.1109/IEMCON.2018.8614822.
- [47] Zhigang Yu et al. “An Elderly Fall Detection Method Based on Federated Learning and Extreme Learning Machine (Fed-ELM)”. In: *IEEE Access* 10 (2022), pp. 130816–130824. DOI: 10.1109/ACCESS.2022.3229044.
- [48] Bendong Zhao et al. “Convolutional neural networks for time series classification”. In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169. DOI: 10.21629/JSEE.2017.01.18.
- [49] IBM. “What are Convolutional Neural Networks?” In: *IBM* (2025). Accessed: 2025-02-28. URL: <https://www.ibm.com/think/topics/convolutional-neural-networks>.
- [50] E. M. Dogo et al. “A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks”. In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. 2018, pp. 92–99. DOI: 10.1109/CTEMS.2018.8769211.
- [51] Like Hui and Mikhail Belkin. “Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks”. In: *CoRR* abs/2006.07322 (2020). arXiv: 2006.07322. URL: <https://arxiv.org/abs/2006.07322>.
- [52] Sensative AB. *YGGIO - Data infrastructure Management System*. Accessed: 2025-03-10. 2025. URL: <https://sensative.com/yggio/>.



LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2025-1047
<http://www.eit.lth.se>