Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης Και Υπολογιστων

On-Device Federated Learning for Human Activity Recognition

Διπλωματικη Εργασια

του

ΔΗΜΗΤΡΙΟΥ ΜΑΤΣΟΥΚΑ

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής, Ε.Μ.Π

Αθήνα, Ιούνιος 2025

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομεας Τεχνολογίας Πληροφορικής Και Υπολογιστών

On-Device Federated Learning for Human Activity Recognition

Διπλωματικη Εργασια

του

ΔΗΜΗΤΡΙΟΥ ΜΑΤΣΟΥΚΑ

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής, Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20 Ιουνίου 2025.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Παναγιώτης Τσανάκας	Δημήτριος Σούντρης	Σωτήριος Ξύδης
Καθηγητής, Ε.Μ.Π	Καθηγητής, Ε.Μ.Π	Επίκουρος Καθηγητής, Ε.Μ.Π

Αθήνα, Ιούνιος 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Technology, Information and Computers

On-Device Federated Learning for Human Activity Recognition

Diploma Thesis

of

DIMITRIOS MATSOUKAS

Supervisor: Panayiotis Tsanakas
Professor, NTUA

Athens, June 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Technology, Information and Computers

On-Device Federated Learning for Human Activity Recognition

Diploma Thesis

of

DIMITRIOS MATSOUKAS

Supervisor: Panayiotis Tsanakas
Professor, NTUA

Approved by the examination committee on 20th June 2025.

(Signature)	*(Signature)*	*(Signature)*
Panayiotis Tsanakas	Dimitrios Sountris	Sotirios Xydis
Professor, NTUA	Professor, NTUA	Assistant Professor, NTUA

Athens, June 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Technology, Information and Computers

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

. . . . . .
Dimitrios Matsoukas
20th June 2025

Ευχαριστίες

Αναγνωρίζοντας ότι αυτή η εργασία αποτελεί προϊόν συστηματικής ενασχόλησης και συντονισμένης δουλειάς, θα ήθελα να απευθύνω τις ευχαριστίες μου στους Ανθρώπους που συνέβαλαν στην επίτευξη αυτού του σκοπού. Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Παναγιώτη Τσανάκα, που μου έδωσε την ευκαιρία να καταπιαστώ με ένα τόσο ενδιαφέρον και πολυδιάστατο θέμα. Εν συνεχεία , οφείλω τις θερμές μου ευχαριστίες στους υποψήφιους διδάκτορες Γεώργιος Δραινάκης και Παναγιώτης Πανταζόπουλος για την αμέριστη και διαρκή υποστήριξή τους σε όλα τα στάδια της εκπόνησης της εργασίας, δίνοντάς μου σημαντική καθοδήγηση. Τέλος θα ήθελα να ευχαριστήσω τους δικούς μου ανθρώπους που με την κατανόηση, την στήριξη και την θετική τους σκέψη , συνέβαλαν στην επίτευξη ενός ακόμα στόχου μου. Όλοι οι προαναφερόμενοι, ο κάθε ένας από την δική του πλευρά μου έδωσαν την θέληση και το όραμα να συνεχίσω να προσπαθώ για τους στόχους και τις φιλοδοξίες μου.

Περίληψη

Οι συσκευές περιορισμένων υπολογιστικών πόρων, όπως τα έξυπνα κινητά τηλέφωνα (smartphones) και οι έξυπνες φορετές συσκευές (wearables), έχουν πλέον καθιερωθεί ως οι βασικές πλατφόρμες υπολογιστικής δραστηριότητας των χρηστών, παράγοντας σε καθημερινή βάση μεγάλους όγκους ευαίσθητων και εξατομικευμένων δεδομένων. Τα μοντέλα μηχανικής μάθησης μπορούν να αξιοποιήσουν αυτά τα δεδομένα σε εφαρμογές που αφορούν την όραση υπολογιστών, την επεξεργασία φυσικής γλώσσας και την παρακολούθηση της υγείας.

Ως επί το πλείστον, η λειτουργία της μηχανικής μάθησης βασίζεται στην κεντρικοποιημένη συλλογή δεδομένων. Ωστόσο αυτή η προσέγγιση ενέχει σοβαρούς κινδύνους για την ιδιωτικότητα των χρηστών και επιπλέον περιορίζεται όλο και περισσότερο από κανονισμούς όπως ο Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR) και ο Νόμος περί Φορητότητας και Λογοδοσίας Ασφάλισης Υγείας (HIPAA).

Μία πολλά υποσχόμενη εναλλακτική είναι η Ομοσπονδιακή Μάθηση (Federated Learning), η οποία αντιμετωπίζει τα ζητήματα αυτά μέσω μιας κατανεμημένης προσέγγισης. Η εκπαίδευση των μοντέλων πραγματοποιείται τοπικά, απευθείας στις συσκευές των χρηστών, εξαλείφοντας την ανάγκη αποστολής ευαίσθητων δεδομένων σε κεντρικούς διακομιστές. Η πλειονότητα της έρευνας στον τομέα της ΟΜ βασίζεται σε προσομοιώσεις με χρήση τυποποιημένων συνόλων δεδομένων ως σημεία αναφοράς για την αξιολόγηση της απόδοσης. Ωστόσο, οι προσεγγίσεις αυτές συχνά παραβλέπουν τις προκλήσεις που προκύπτουν σε πραγματικές συνθήκες, οι οποίες επηρεάζονται σημαντικά από τυχόν περιορισμούς υλικού, την κατανάλωση ενέργειας και την αστάθεια των δικτύων. Η παρούσα διπλωματική εργασία επιχειρεί να καλύψει αυτό το κενό, υλοποιώντας ένα σύστημα Ομοσπονδιακής Μάθησης για το πρόβλημα της Αναγνώρισης Ανθρώπινης Δραστηριότητας (Human Activity Recognition), μια εφαρμογή με έντονη σημασία για την προστασία της ιδιωτικότητας καθώς βασίζεται σε δεδομένα που συλλέγονται από προσωπικές συσκευές.

Το προτεινόμενο σύστημα χρησιμοποιεί έναν διακομιστή βασισμένο στο λογισμικό Flower, ο οποίος συντονίζει την εκπαίδευση πέντε Android κινητών τηλεφώνων. Η εκπαίδευση και αξιολόγηση των μοντέλων πραγματοποιούνται τοπικά, με τη χρήση του TensorFlow Lite (TFLite), ενός εκ των ελάχιστων εργαλείων που υποστηρίζουν όχι μόνο την εξαγωγή προβλέψεων (inference), αλλά και την τοπική εκπαίδευση (training) μοντέλων σε συσκευές με περιορισμένους πόρους.

Μέσω πειραματικής αξιολόγησης, η εργασία ποσοτικοποιεί την επίδραση βασικών προκλήσεων της ΟΜ στην ΑΑΔ κατά μήκος τριών κρίσιμων αξόνων: ετερογένεια δεδομένων, ενεργειακή απόδοση και αξιοπιστία του δικτύου. Τα αποτελέσματα δείχνουν ότι η έντονη ανισοκατανομή μεταξύ κατηγοριών (class imbalance) μπορεί να μειώσει την ακρίβεια του μοντέλου κατά περισσότερο από 55%. Αντίθετα, όταν ο όγκος των δεδομένων εκπαίδευσης

ανά συσκευή μειώνεται έως και στο 10%, η απόδοση του μοντέλου υποχωρεί μόλις κατά 2%, υποδεικνύοντας περιορισμένη ευαισθησία στην ανισότητα όγκου. Τα ενεργειακά πειράματα καταδεικνύουν ότι η ενίσχυση της τοπικής εκπαίδευσης (epochs) σε κάθε συσκευή, με ταυτόχρονη μείωση των γύρων επικοινωνίας (federated learning communication rounds), μπορεί να μειώσει την κατανάλωση ενέργειας κατά ποσοστό που υπερβαίνει το 84%, χωρίς αξιοσημείωτη απώλεια στην ακρίβεια. Τέλος, τα πειράματα δικτύου δείχνουν ότι οι αποσυνδέσεις χρηστών και η διαλείπουσα συμμετοχή μπορούν να οδηγήσουν σε απώλεια απόδοσης έως και 20%, με ταυτόχρονη αύξηση της αστάθειας κατά τη διάρκεια της εκπαίδευσης.

Η μελλοντική έρευνα μπορεί να προχωρήσει προς δύο κατευθύνσεις: **(i)** την επέκταση του υπάρχοντος πλαισίου Ομοσπονδιακής Μάθησης με πιο σθεναρούς αλγορίθμους συγχώνευσης (aggregation) όπως οι FedProx και SCAFFOLD **(ii)** την επέκταση του πειραματικού πεδίου, μέσα από διάφορες πιθανές εκφάνσεις, όπου μία εξ αυτών είναι η διερεύνηση του τρόπου με τον οποίο διαφορετικές αρχιτεκτονικές μοντέλων επηρεάζουν τη σχέση μεταξύ κατανάλωσης ενέργειας και απόδοσης.

Λέξεις Κλειδιά

Ομοσπονδιακή Μάθηση, Μηχανική Μάθηση, Αναγνώριση Ανθρώπινης Δραστηριότητας, Συσκευές περιορισμένων πόρων, Τοπική εκπαίδευση μοντέλων, Ανισοκατανομή κατηγοριών, Κατανάλωση ενέργειας, Αξιοπιστία δικτύου, TensorFlow Lite, Flower framework

Abstract

Edge devices such as smartphones and wearables have become the primary computing platforms, generating large volumes of sensitive, user-specific data. Machine Learning (ML) models can utilize this data for tasks in areas like computer vision, natural language processing, and health monitoring. Traditionally, ML relies on centralized data collection, but this approach introduces serious privacy risks and is increasingly constrained by regulations such as GDPR and HIPAA. Federated Learning (FL) offers a promising alternative by addressing privacy concerns through a decentralized training approach, where model training occurs directly on users devices. This eliminates the need to transmit sensitive data to a central server. However, most FL research relies in simulation-based studies using standardized datasets, often neglecting the real-world challenges posed by hardware limitations, energy constraints, and network instability. This thesis addresses that gap by implementing a real-world FL system for Human Activity Recognition (HAR), which is a privacy-sensitive task that leverages sensor data from mobile devices. HAR is selected for its practical relevance and dependence on data commonly collected by personal devices. The system uses a Flower-based server coordinating training across five Android smartphones, with on-device training and evaluation conducted via TensorFlow Lite (TFLite) which is one of the few frameworks supporting local updates on mobile hardware.

Through experimental evaluation, the thesis quantifies how key FL challenges impact HAR across three critical axes: data heterogeneity, energy efficiency, and network reliability. Results show that extreme label imbalance can degrade model accuracy by over 55%. In contrast, when the amount of training data per client is reduced to just 10%, model's performance drops by only 2%, indicating the relatively low sensitivity to data volume imbalance. Energy experiments show that increasing local training on each device while reducing the number of communication rounds can reduce energy consumption by over 84% without compromising accuracy. Finally, network experiments reveal that client dropouts and intermittent participation lead to up to 20% performance loss and increased training instability, emphasizing the importance of robust aggregation strategies in real-world deployments.

Future work can proceed in two directions: **(i)** extending the current FL framework with more robust aggregation algorithms such as FedProx and SCAFFOLD to improve resilience to non-IID data and client dropout, and **(ii)** expanding the experimental space by exploring aspects such as energy–accuracy trade-offs across different model architectures.

Keywords

Edge device, Machine Learning, Federated Learning, Human Activity Recognition, TensorFlow Lite, Flower framework, Data privacy, GDPR, HIPAA, On-device training, Data heterogeneity, Energy efficiency, Network reliability, Mobile devices, Decentralized learning, Android, Label imbalance

Εκτεταμένη Ελληνική Περίληψη

Τα τελευταία χρόνια, οι συσκευές στα Άκρα του Δικτύου (edge devices) με περιορισμένους υπολογιστικούς πόρους, όπως τα έξυπνα κινητά τηλέφωνα (smartphones), οι φορητές συσκευές (wearables) καθώς και οι συσκευές του Διαδικτύου των Πραγμάτων (Internet of Things - IoT) έχουν καταστεί κυρίαρχες υπολογιστικές πλατφόρμες, χρησιμοποιούμενες καθημερινά από δισεκατομμύρια ανθρώπους. Οι συσκευές αυτές είναι εξοπλισμένες με πληθώρα αισθητήρων, π.χ., κάμερες, μικρόφωνα, συστήματα εντοπισμού (Global Positioning System - GPS), επιταχυνσιόμετρα, γυροσκόπια και βιομετρικούς αναγνώστες και παράγουν μεγάλο όγκο προσωπικών δεδομένων. Αξιοποιώντας αυτές τις πλούσιες πηγές δεδομένων, η Μηχανική Μάθηση (Machine Learning - ML) έχει επιφέρει ριζικές αλλαγές προσφέροντας αποδοτικές λύσεις σε τομείς όπως η υγεία και τα χρηματοοικονομικά.

Ένα μοντέλο μηχανικής μάθησης όπως ένα βαθύ νευρωνικό δίκτυο (Deep Neural Network - DNN) εκπαιδεύεται τυπικά μέσω ενός κεντρικού διακομιστή (server), ο οποίος συλλέγει το σύνολο των δεδομένων των χρηστών. Ωστόσο, η προσέγγιση αυτή εγείρει σοβαρές ανησυχίες σχετικά με την ιδιωτικότητα και σε πολλές περιπτώσεις δεν αποτελεί πλέον βιώσιμη λύση λόγω της εκτεταμένης εφαρμογής κανονισμών, όπως αυτής του Γενικού Κανονισμού Προστασίας Δεδομένων της Ευρωπαϊκής Επιτροπής (General Data Protection Regulation - GDPR). Αυτές οι ανησυχίες καθίστανται ακόμη πιο κρίσιμες όταν τα δεδομένα έχουν άκρως προσωπικό χαρακτήρα, όπως στον τομέα της υγείας ή στα χρηματοοικονομικά, όπου εφαρμόζονται αυστηρότεροι κανονισμοί όπως ο Νόμος περί Φορητότητας και Λογοδοσίας στην Ασφάλιση Υγείας (Health Insurance Portability and Accountability Act - HIPAA).

Η Ομοσπονδιακή Μάθηση - ΟΜ (Federated Learning - FL) έχει αναδειχθεί ως μια λύση στα υπάρχοντα προβλήματα ιδιωτικότητας, επιτρέποντας σε πολλούς υπολογιστικούς κόμβους να εκπαιδεύσουν από κοινού ένα μοντέλο χωρίς να κοινοποιούν τα τοπικά (προσωπικής φύσης) δεδομένα. Σε υψηλό επίπεδο, η Ομοσπονδιακή Μάθηση λειτουργεί (επαναλαμβανόμενα) μέσα από τρία βασικά βήματα: 1) οι συσκευές που συμμετέχουν (clients) εκπαιδεύουν τοπικά ένα κοινό (γενικό) μηχανικό μοντέλο χρησιμοποιώντας τα ιδιωτικά τους δεδομένα, 2) στέλνουν τις ενημερώσεις των επιμέρους μοντέλων σε έναν κεντρικό διακομιστή για συγχώνευση (aggregation) σε ένα νέο γενικό μοντέλο, και 3) ο διακομιστής μεταδίδει το ανανεωμένο γενικό μοντέλο πίσω στους συμμετέχοντες για τον επόμενο γύρο εκπαίδευσης.

Παρά τα πλεονεκτήματα που προσφέρει η Ομοσπονδιακή Μάθηση, αντιμετωπίζει βασικές προκλήσεις, όπως η ετερογένεια των συσκευών αλλά και των συλλεγόμενων δεδομένων καθώς η κατανομή τους μπορεί να διαφέρει σημαντικά μεταξύ των συσκευών. Αν και μεγάλο μέρος της υφιστάμενης έρευνας επικεντρώνεται στην αντιμετώπιση αυτών των ζητημάτων, συχνά περιορίζεται σε μελέτες προσομοίωσης που χρησιμοποιούν τυποποιημένα σύνολα δεδομένων αναφοράς, όπως το CIFAR-10. Τέτοιες προσεγγίσεις αγνοούν ουσιώδεις περιορισμούς του

πραγματικού κόσμου, όπως οι περιορισμένοι υπολογιστικοί πόροι, η αστάθεια του δικτύου και η διάρκεια ζωής της μπαταρίας. Αυτοί οι παράγοντες είναι κρίσιμοι για την πρακτική εφαρμογή της Ομοσπονδιακής Μάθησης.

Η παρούσα εργασία υποστηρίζει ότι το χάσμα μεταξύ προσομοίωσης και υλοποίησης πρέπει να καλυφθεί, προκειμένου να προχωρήσει η εξέλιξη της Ομοσπονδιακής Μάθησης. Η εφαρμογή της Ομοσπονδιακής Μάθησης σε πραγματικές συσκευές (π.χ., smartphones), με τη χρήση μιας ρεαλιστικής εφαρμογής, μπορεί να αναδείξει τη βιωσιμότητα του παραδείγματος αυτού σε πραγματικές συνθήκες, ενώ παράλληλα να καθοδηγήσει τόσο την ανάπτυξη υποδομών (federated framework) όσο και τη βελτιστοποίηση των αλγορίθμων συγχώνευσης (aggregation algorithm).

Ένας ιδανικός τομέας για τη δοκιμή της ΟΜ σε πραγματικές συνθήκες είναι το πρόβλημα της Αναγνώρισης Ανθρώπινης Δραστηριότητας - ΑΑΔ (Human Activity Recognition - HAR). Η ΑΑΔ αφορά τη διαδικασία αυτόματης ανίχνευσης και ταξινόμησης ανθρώπινων συμπεριφορών ή δραστηριοτήτων, αξιοποιώντας δεδομένα αισθητήρων που συλλέγονται από συσκευές όπως smartphones και wearables. Οι δραστηριότητες αυτές μπορεί να κυμαίνονται από απλές κινήσεις (π.χ., περπάτημα, καθιστή στάση, τρέξιμο) έως πιο σύνθετες συμπεριφορές (π.χ., μαγείρεμα, οδήγηση, άσκηση), περιλαμβάνοντας και εργασίες σχετικές με την παρακολούθηση της υγείας (π.χ., ανάλυση βαδίσματος για ανίχνευση πτώσεων ή παρακολούθηση καρδιακών παλμών). Οι εφαρμογές της στον πραγματικό κόσμο είναι ποικίλες και έχουν αποδεδειγμένα σημαντική επίδραση σε τομείς όπως η υγειονομική περίθαλψη και η φυσική κατάσταση. Ο ευαίσθητος χαρακτήρας των δεδομένων της ΑΑΔ και η μεγάλη πρακτική της αξία την καθιστούν κατάλληλη για τη χρήση της ΟΜ. Επιπλέον, η ΑΑΔ μπορεί να υλοποιηθεί σε υπάρχοντα smartphones και smartwatches, τα οποία διαθέτουν ήδη τους απαραίτητους αισθητήρες, γεγονός που καθιστά τη συγκεκριμένη εφαρμογή ιδιαίτερα προσβάσιμη.

Για να διερευνηθεί η ενσωμάτωση της ΟΜ σε πραγματικό περιβάλλον, αναπτύχθηκε ένα πλήρες σύστημα ΟΜ. Ο διακομιστής υλοποιήθηκε με χρήση του Flower, ενός ευρέως διαδεδομένου και ανοιχτού λογισμικού ΟΜ, σχεδιασμένου για την υποστήριξη εκπαίδευσης σε πολλαπλές συσκευές στα άκρα του δικτύου (edge devices). Το Flower ακολουθεί την κλασική αρχιτεκτονική server-client και προσφέρει υψηλό βαθμό παραμετροποίησης καθιστώντας το κατάλληλο για ερευνητική χρήση. Στην πλευρά των συσκευών, χρησιμοποιήθηκαν smartphones με λειτουργικό Android. Η εκπαίδευση και αξιολόγηση των μοντέλων πραγματοποιούνται τοπικά (on-device), με τη χρήση του TensorFlow Lite (TFLite), ενός εκ των ελάχιστων εργαλείων που υποστηρίζουν όχι μόνο την εξαγωγή προβλέψεων (inference), αλλά και την τοπική εκπαίδευση (training) μοντέλων σε συσκευές με περιορισμένους πόρους. Το σύστημα περιλαμβάνει επίσης ένα μέρος προεπεξεργασίας δεδομένων, υπεύθυνο για την προετοιμασία και διανομή των συνόλων δεδομένων ΑΑΔ στους συμμετέχοντες. Για την αξιολόγηση της πρακτικής βιωσιμότητας της ΟΜ και την καλύτερη κατανόηση των συμβιβασμών που συνεπάγεται, κάθε συσκευή έχει προσαρμοστεί ώστε να καταγράφει διάφορες μετρικές καθ' όλη τη διάρκεια της εκπαίδευσης. Οι μετρικές αυτές περιλαμβάνουν: κατανάλωση ενέργειας, απόκριση επικοινωνίας (communication latency), χρόνο εκπαίδευσης και ακρίβεια του μοντέλου. Συνολικά, οι μετρήσεις αυτές επιτρέπουν μια ολοκληρωμένη αξιολόγηση της ΟΜ σε τρεις βασικούς άξονες: απόδοση, ενεργειακή αποδοτικότητα και αξιοπιστία δικτύου. Εν συνεχεία παρουσιάζεται αναλυτικότερα το περιεχόμενο των ενοτήτων.

Το Κεφάλαιο 2 θέτει τη θεωρητικό υπόβαθρο της παρούσας διπλωματικής. Ξεκινά με μια συνοπτική επισκόπηση των βασικών αρχών της μηχανικής μάθησης, περιλαμβάνοντας τις αρχές της αυτο-εποπτευόμενης μάθησης (supervised learning), τη δομή των νευρωνικών δικτύων, καθώς και τη σημασία των υπερπαραμέτρων (hyper-parameters), όπως ο ρυθμός μάθησης (learning rate), το μέγεθος παρτίδας (batch size) και ο αριθμός εποχών (epochs).

Ιδιαίτερη έμφαση δίνεται στις προκλήσεις που προκύπτουν από την ύπαρξη ετερογενών δεδομένων (non-independent and identically distributed data - non-IID data), δηλαδή σε περιπτώσεις όπου οι κατανομές δεδομένων διαφέρουν σημαντικά μεταξύ των συμμετεχόντων συσκευών. Αυτό το πρόβλημα είναι ιδιαίτερα εμφανές στην περίπτωση της ΑΑΔ, όπου η συμπεριφορά του εκάστοτε χρήστη προκαλεί σημαντική διαφοροποίηση στα σήματα των αισθητήρων.

Επιπλέον, το κεφάλαιο εξετάζει υπάρχοντα λογισμικά (Federated Learning frameworks) για την ΟΜ, όπως τα TensorFlow Federated, FedML, PySyft και Flower, αξιολογώντας τα με βάση την υποστήριξη συσκευών, την τεκμηρίωση και τη δυνατότητα επέκτασης. Η επιλογή του Flower οφείλεται στη συνεχή του ανάπτυξη, στη διαλειτουργικότητα μεταξύ διαφορετικών πλατφορμών και στην υποστήριξη συμμετεχόντων συσκευών με λειτουργικό Android.

Το κεφάλαιο ολοκληρώνεται με ανασκόπηση προηγούμενων σχετικών εργασιών και εντοπίζει το κύριο ερευνητικό κενό: την απουσία εφαρμογών Ομοσπονδιακής Μάθησης σε πραγματικές κινητές συσκευές, ιδίως σε σενάρια που αφορούν ευαίσθητες και ιδιωτικού χαρακτήρα εφαρμογές όπως η ΑΑΔ.

Το Κεφάλαιο 3 εμβαθύνει στο πεδίο του προβλήματος της ΑΑΔ ξεκινώντας με τον επίσημο ορισμό του και μια επισκόπηση της σημασίας του σε εφαρμογές του πραγματικού κόσμου.

Στη συνέχεια εξετάζει τόσο τις ενδογενείς προκλήσεις της ΑΑΔ όσο και εκείνες που σχετίζονται ειδικά με την ΟΜ. Οι ενδογενείς προκλήσεις περιλαμβάνουν την ανισοκατανομή κατηγοριών (π.χ., η δραστηριότητα ΄καθιστή στάση΄ υπεραντιπροσωπεύεται, ενώ η άνάβαση σκάλας΄ είναι σπάνια), την ομοιότητα μεταξύ κατηγοριών (π.χ., ΄περπάτημα΄ έναντι έλαφρού τρεξίματος΄) και την ενδοκατηγορική μεταβλητότητα (π.χ., διαφορές στον τρόπο βάδισης μεταξύ χρηστών). Τα ζητήματα αυτά περιπλέκουν την εκπαίδευση και την αξιολόγηση των μοντέλων. Από την πλευρά της ΟΜ, η ΑΑΔ παρουσιάζει επιπλέον δυσκολίες, όπως:

- η εξατομίκευση (δηλαδή η προσαρμογή του μοντέλου σε ατομικά χαρακτηριστικά)
- οι αλλαγές στη συμπεριφορά των χρηστών με την πάροδο του χρόνου
- η ετερογένεια συσκευών (διαφορετικές υπολογιστικές δυνατότητες μεταξύ smartphones και wearables)

Σημαντικό μέρος του κεφαλαίου αφιερώνεται στη διαχείριση των δεδομένων. Παρουσιάζεται μια δομημένη επισκόπηση της ροής επεξεργασίας δεδομένων στην ΑΑΔ: από την απόκτηση δεδομένων μέσω επιταχυνσιόμετρων και γυροσκοπίων, έως τα στάδια προεπεξεργασίας όπως η φιλτραρίσματος θορύβου και κανονικοποίησης, καθώς και οι τεχνικές τμηματοποίησης (segmentation) και εξαγωγής χαρακτηριστικών (feature extraction). Παρουσιάζεται μελέτη περίπτωσης του συνόλου δεδομένων UCI HAR, με έμφαση σε τεχνικές όπως φιλτράρισμα μέσω

διαμέσου (median filtering), βαθυπερατό φίλτρο Butterworth και τμηματοποίηση με χρήση παραθύρων (windowed segmentation).

Τέλος, το κεφάλαιο εισάγει τα έξι σύνολα δεδομένων που χρησιμοποιήθηκαν στη διπλωματική εργασία: UCI HAR, PAMAP2, MHealth, HARSense, PhysioNet, MotionSense. Κάθε σύνολο επελέγη ώστε να καλύπτει όλο το φάσμα σε τύπους αισθητήρων, συχνότητες δειγματοληψίας, πολυπλοκότητα προεπεξεργασίας και εύρος δραστηριοτήτων. Η εν λόγω ποικιλομορφία επιτρέπει την εύρωστη αξιολόγηση του συστήματος ΟΜ υπό διαφορετικές συνθήκες δεδομένων και συμπεριφορών χρηστών.

Το Κεφάλαιο 4 παρουσιάζει το πλήρες σύστημα ΟΜ που αναπτύχθηκε για την αξιολόγηση του προβλήματος ΑΑΔ σε πραγματικές συσκευές στα άκρα του δικτύου (edge devices). Η υποδομή σχεδιάστηκε με βάση δύο θεμελιώδεις αρχές: την επεκτασιμότητα και την ταχεία πειραματική αξιολόγηση. Αν και όλα τα πειράματα πραγματοποιήθηκαν σε Android smartphones χρησιμοποιώντας το TFLite, το σύστημα δύναται να υποστηρίξει ευρύτερη επεκτασιμότητα σε άλλες πλατφόρμες και Machine Learning frameworks.

Η πειραματική διάταξη αποτελείται από έναν κεντρικό διακομιστή και πέντε Android smartphones, όλα συνδεδεμένα σε τοπικό δίκτυο Wi-Fi. Οι συμμετέχοντες εκτελούν μια Android εφαρμογή, η οποία διαχειρίζεται την τοπική εκπαίδευση και αξιολόγηση του μοντέλου, ενώ ο διακομιστής είναι υπεύθυνος για τη συγχώνευση των μοντέλων, τη διανομή των δεδομένων και τον έλεγχο των πειραμάτων. Για την υποστήριξη ταχέων πειραματικών κύκλων, το σύστημα περιλαμβάνει αυτοματοποιημένα εργαλεία για την προεπεξεργασία των δεδομένων, τη διαμόρφωση αρχιτεκτονικών μοντέλων, και την καταγραφή βασικών μετρικών όπως ο χρόνος εκπαίδευσης, η απόδοση του μοντέλου, η κατανάλωση ενέργειας και οι συνθήκες του δικτύου.

Μία από τις βασικές τεχνικές προκλήσεις που αντιμετωπίστηκαν στο σχεδιασμό του συστήματος ήταν η αδυναμία του TFLite να ενημερώνει τις παραμέτρους των μοντέλων προγραμματιστικά. Αυτό απαίτησε από τον διακομιστή να ανασυνθέτει και να επαναμεταδίδει ολόκληρο το μοντέλο σε κάθε γύρο, γεγονός που δημιουργεί σημαντικό υπολογιστικό και επικοινωνιακό κόστος.

Η υποδομή περιλαμβάνει επίσης ένα σύστημα επεξεργασίας δεδομένων (data pipeline), ικανό να προσομοιώνει διάφορες καταστάσεις ανομειογενών δεδομένων (non-IID) μέσω ελεγχόμενης τμηματοποίησης και δημιουργίας ανισοκατανομής κατηγοριών. Οι συμμετέχοντες, ο διακομιστής και το σύστημα συλλογής μετρικών (Statistics Collector) συνεργάζονται ώστε να καταστήσουν δυνατή τη συστηματική αξιολόγηση της ΟΜ. Η αξιολόγηση χωρίζεται στις εξής κατηγορίες: την ετερογένεια δεδομένων, την ενεργειακή αποδοτικότητα και την αξιοπιστία δικτύου.

Το Κεφάλαιο 5 αποτελεί τον πυρήνα της πειραματικής ανάλυσης. Παρουσιάζει μια σειρά από δομημένα πειράματα, με σκοπό την αξιολόγηση της επίδρασης των περιορισμών της ΟΜ στην απόδοση των μοντέλων κατά μήκος τριών βασικών αξόνων: ετερογένεια δεδομένων, ενεργειακή αποδοτικότητα και αξιοπιστία δικτύου.

- **Ετερογένεια Δεδομένων:** Τα πειράματα προσομοιώνουν διάφορα σενάρια non-IID κατανομών, περιλαμβάνοντας παραδείγματα ανισοκατανομής ετικετών (class imbalance),

ανισομερούς όγκου δεδομένων (quantity imbalance) καθώς και τον συνδυασμό τους. Ενδεικτικά, μοντέλα που εκπαιδεύτηκαν υπό ακραίες συνθήκες ανισοκατανομής ετικετών παρουσίασαν μείωση στην απόδοση που υπερέβη το 55%, ενώ η μείωση του όγκου δεδομένων ανά συμμετέχοντα στο 10% είχε ως αποτέλεσμα απώλεια ακρίβειας μόλις 2%. Τα αποτελέσματα αυτά υποδεικνύουν ότι η OM είναι περισσότερο ευαίσθητη στην κατανομή των ετικετών, παρά στον ανισομερή όγκο δεδομένων.

- **Ενεργειακή Αποδοτικότητα:** Ένα σύνολο πειραμάτων διερεύνησε τη σχέση μεταξύ κατανάλωσης ενέργειας και απόδοσης, μεταβάλλοντας τον αριθμό των τοπικών εποχών εκπαίδευσης (local epochs). Με την αύξηση της τοπικής υπολογιστικής επιβάρυνσης και τη μείωση της συχνότητας επικοινωνίας, η κατανάλωση ενέργειας μειώθηκε έως και 84%, χωρίς απώλειες στην ακρίβεια. Επιπρόσθετα πειράματα εξέτασαν την επίδραση της πολυπλοκότητας του μοντέλου, αποδεικνύοντας ότι τα απλά μοντέλα αποδίδουν ομοίως με τα σύνθετα, καταναλώνοντας σημαντικά λιγότερη ενέργεια.

- **Αξιοπιστία δικτύου:** Η επίδραση της ασταθούς συνδεσιμότητας προσομοιώθηκε μέσω πιθανολογικής συμμετοχής (probabilistic participation model) συμμετεχόντων και μόνιμων αποσυνδέσεων (permanent dropouts). Η διαλείπουσα συμμετοχή οδήγησε σε μείωση της ακρίβειας έως και 20% και προκάλεσε αστάθεια στη σύγκλιση των μοντέλων. Τα αποτελέσματα αυτά υπογραμμίζουν τη σημασία ανθεκτικών στρατηγικών συγχώνευσης (fault-tolerant aggregation).

Συνολικά, τα πειράματα αυτά προσφέρουν μια ολοκληρωμένη εικόνα των αναπόφευκτων συμβιβασμών που προκύπτουν από την υλοποίηση της OM σε πραγματικές συνθήκες. Επιπλέον, επικυρώνουν τη χρηστικότητα της υποδομής που αναπτύχθηκε στο Κεφάλαιο 4 και προσφέρουν πρακτικά συμπεράσματα για τη βελτίωση της OM σε περιβάλλοντα κινητών συσκευών.

Το τελευταίο κεφάλαιο συνθέτει τα ευρήματα της παρούσας διπλωματικής εργασίας. Αποτυπώνονται επίσης οι περιορισμοί της εργασίας. Συγκεκριμένα, επισημαίνεται ότι η υλοποίηση αξιολογήθηκε σε περιορισμένη κλίμακα με μόλις πέντε συμμετέχοντες. Επιπλέον δεν ενσωματώθηκε κάποιος μηχανισμός ασφαλούς συγχώνευσης (secure aggregation), ενώ διαπιστώνεται και η ανάγκη για βελτιωμένη υποστήριξη της τοπικής εκπαίδευσης από το TFLite. Τέλος, προτείνονται δύο βασικές κατευθύνσεις μελλοντικής έρευνας που αφορούν: 1) τη βελτίωση του υπάρχοντος πλαισίου OM που εκφράζεται κυρίως στην ενσωμάτωση προηγμένων αλγορίθμων aggregation και 2) την επέκταση της πειραματικής διερεύνησης για να εξεταστούν πιο σύνθετες συμπεριφορές της OM, όπως για παράδειγμα η σχέση μεταξύ της αρχιτεκτονικής του μηχανικού μοντέλου με την απόδοση (ακρίβεια μοντέλου) και το αντίστοιχο ενεργειακό κόστος.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, resource-constrained edge devices such as smartphones, wearables and IoT devices have become the dominant computing platforms, used daily by billions of people [1]. These devices are equipped with a wide variety of sensors (e.g., cameras, microphones, GPS, accelerometers, gyroscopes, and biometric readers) and generate large volume of user-centric and often sensitive data. By leveraging this rich data sources, Machine Learning (ML) has revolutionized industries such as health, transportation, and finance. Traditionally, a ML model like deep neural network (DNN) is trained on a central server by data collected from every edge device [2]. However, this approach poses serious privacy concerns and in many case it is no longer a sustainable solution due to increasing legislative pressures such as European Commission's General Data Protection Regulation (GDPR) [3]. These concerns are even more critical on cases where data is highly personal such as health or finance, where there are stricter regulations like Health Insurance Portability and Accountability Act (HIPAA) [4] which protect the privacy and security of sensitive health-related information.

Federated Learning (FL) [5] has emerged as a solution to privacy challenges by enabling many computing nodes to collaboratively train a model while keeping data private. At a high level, FL operates by iterating three core steps: i. clients locally update a shared model on their private data, ii. they send these updates to a central server for aggregation, and iii. the server broadcasts the updated global model back to clients for the next round of training. Despite its promise, there are some key challenges such as the heterogeneous nature of edge devices [6] and the collected data [7] where data distribution can significantly vary across devices. While much of the current FL research focuses on addressing these issues, it is often limited to simulation-based studies using standardized benchmark datasets such as CIFAR-10 [8]. These setups overlook key real-world constraints such as limited computational resources, network instability, and battery life which are essential deploying FL to address practical problems.

This thesis argues that closing the gap between simulation and deployment is essential for the evolution of FL. Demonstrating FL on real edge devices (e.g smartphones) using a practical use case can illuminate the real-world feasibility of this paradigm and guide both infrastructure development and algorithmic advances. An ideal application domain to test the feasibility of real-world FL is the Human Activity Recognition (HAR) problem [9]. HAR is the process of automatically detecting and classifying human behaviors or

activities using sensor data collected from devices such as smartphones and wearables. These activities can range from simple actions (e.g., walking, sitting, running) to more complex behaviors (e.g., cooking, driving, exercising), and may also include health-related monitoring tasks (e.g., gait analysis for fall detection or heart rate tracking for cardiac conditions). The real-world applications are diverse and have demonstrated significant impact in domains such as healthcare, fitness, and smart environments. HAR's privacy-sensitive data and significant practical value make it an ideal use case for FL. Moreover, HAR systems can already be deployed on existing smartphones and smartwatches which are equipped with the necessary sensors, making it an highly accessible testbed.

To explore the integration of FL in a real-world setting, a complete end-to-end FL system was developed. The server was implemented using Flower [10] which is a widely used open-source FL framework designed to support training across multiple edge devices. Flower follows the standard server–client architecture and offers high customization capabilities, making it suitable for experimentation and research. On the client side, Android smartphones were used to perform local training and evaluation tasks using TensorFlow Lite (TFLite) [11] which is one of the few frameworks that supports on-device training. The system also includes a data preprocessing pipeline responsible for preparing and partitioning HAR datasets across clients. To assess the practical feasibility of FL and better understand its trade-offs, the server is configured to log various metrics throughout the training process. These metrics include energy consumption, communication latency, training time, and model accuracy. Together, these measurements allow for a comprehensive evaluation of FL across three key dimensions: performance, energy efficiency, and network reliability.

The contributions of this thesis are closely aligned with its motivating objectives. First, it highlights the scarcity and real-world limitations of current ML infrastructure for edge devices. A key challenge encountered was the inability of TFLite to programmatically update model weights. This limitation required the full recompilation and redistribution of the model at each training round, which increased system complexity and convergence speed. Second, this thesis provides an experimental assessment of how key FL challenges affect the performance of the HAR tasks. The performance-focused experiments show that data heterogeneity can significantly degrade model accuracy and stability. Specifically, results show that extreme label imbalance can degrade model accuracy by over 55%. In contrast, when the amount of training data per client is reduced to just 10%, model's performance drops by only 2%, indicating the relatively low sensitivity to data volume imbalance. Energy experiments show that increasing local training on each device while reducing the number of communication rounds can reduce energy consumption by over 84% without compromising accuracy. Finally, network experiments reveal that client dropouts and intermittent participation lead to up to 20% performance loss and increased training instability, emphasizing the importance of robust aggregation strategies in real-world deployments. In summary, this work provides a practical evaluation for applying FL for HAR task on edge devices. By implementing and testing a complete FL system, it offers empirical insights into the trade-offs, bottlenecks, and design considerations that currently shape the feasibility and scalability of FL in real-world applications.

The structure of this thesis is organized as follows: Chapter 2 introduces the foun-

dational concepts of ML and FL, including the core architecture of FL and its major challenges, such as data heterogeneity and system-level constraints. It also provides a concise overview of relevant software frameworks, explaining the rationale behind selecting Flower for this work. Finally, it reviews related studies focused on real-world FL implementations on edge devices. Chapter 3 provides a detailed overview of HAR, exploring its primary use cases, common challenges, and their intersection with FL's constraints. The chapter also presents the six datasets selected for experimentation, which were chosen to represent diverse sensing modalities, preprocessing methods, and activity types. Chapter 4 describes the design and implementation of the end-to-end FL system developed for this thesis. It details both the server-side and client-side components, the integration of TFLite for on-device training, and the mechanisms used to measure energy consumption, network behavior, and model performance. Chapter 5 presents the experimental evaluation, structured around three key dimensions: i) model performance under non-IID and client-scaling conditions, ii) energy consumption as a function of configuration parameters and iii) the impact of unstable network conditions on training efficiency and convergence.

Chapter 2

Foundations of Federated Learning for Edge-Based HAR Applications

This chapter establishes the theoretical background of FL along with preliminary concepts such as neural networks and ML while focusing on the key considerations when it is applied to HAR related applications. Given the broad scope of the research field, the foundations of HAR will be analyzed separately in Chapter 3.

This chapter is organized as follows: Section 2.1 presents an overview of supervised learning and the structure of deep neural networks. Section 2.2 defines FL and distinguishes between its cross-device and cross-silo settings. Section 2.3 formalizes the FL optimization problem and describes the Federated Averaging (FedAvg) algorithm, which is selected in this work as it serves as the standard and most widely adopted method for model aggregation in the FL literature. Section 2.4 outlines key challenges in applying FL to edge environments, including system heterogeneity, limited device resources, and data non-IIDnesss, where data across devices varies in distribution, deviating from the assumption of uniform and independent data. Section 2.5 evaluates existing FL frameworks for edge deployment and provides rationale behind the selection of Flower. Finally, Section 2.6 reviews related work and identifies specific research gaps that this thesis aims to address.

2.1 Fundamentals of Machine Learning and Neural Networks

Machine Learning (ML) represents a paradigm shift in computational modeling, enabling systems to learn complex functions from data rather than relying on explicitly programmed rules. In supervised learning, the most common paradigm, models are trained to minimize a loss function that measures the discrepancy between predicted outputs and ground-truth labels. As described in the standard ML literature [2], parameters are typically optimized via gradient-based methods (e.g. stochastic gradient descent), iteratively adjusting model weights to reduce loss and improve generalization on unseen data.

Neural Networks (NNs) form a powerful class of machine learning (ML) models. Like classical approaches such as Support Vector Machines (SVMs) and Bayesian methods, they are used to learn highly nonlinear relations from data. They have achieved state-of-the-art performance in domains such as computer vision [12] and natural language processing (NLP) [13]. Neural networks are composed of interconnected computational units called

neurons, organized into layers as illustrated in Figure 2.1. These layers include an input layer, which receives the raw data, one or more hidden layers, which extract intermediate representations through transformations and an output layer, which produces the final prediction. Each neuron performs a weighted sum of its inputs (via matrix multiplication and bias addition) followed by a nonlinear activation function such as ReLU or sigmoid [2].

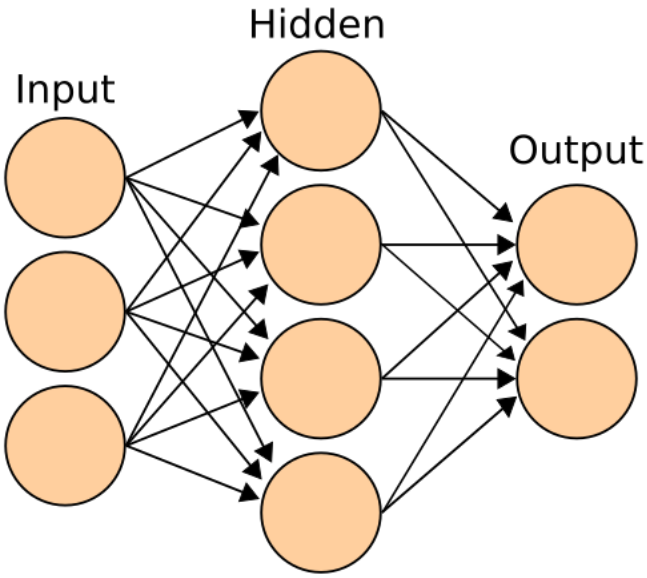


Figure 2.1. *A generic Neural Network architecture*

The ML process can be divided into two main phases: the training phase, where the model learns patterns from data by updating its internal parameters, and the inference phase, where the trained model generates predictions on new, unseen data. During training, several key hyperparameters must be configured to guide the learning process:

Epochs: One epoch represents a full pass through the training dataset. Multiple epochs allow the model to progressively refine its parameters. However, using too many epochs can lead to overfitting, where the model memorizes the training data rather than generalizing to new inputs.

Batch Size: This defines the number of training samples processed before the model's parameters are updated. Smaller batch sizes introduce more variability (or noise) into each update, which can sometimes help improve generalization.

Learning Rate: This determines the step size used during gradient descent to update model weights. It directly affects both the speed and stability of the training process.

In summary, ML and neural networks provide a powerful approach for building data-driven solutions to complex tasks. The performance and behavior of these models are influenced by the chosen architecture (e.g., number of layers, neuron types), the loss function (which quantifies prediction error), and hyperparameters like those defined above.

2.2 Overview of Federated Learning: Principles and Use Cases

FL is a distributed ML paradigm in which model training occurs locally on multiple clients rather than centrally on a single server. Instead of uploading raw data, each client trains a local copy of the global model on its own private dataset and transmits only its updated model parameters (e.g., weights or gradients) to the central server [5]. The server then performs an aggregation step, most commonly by a weighted average method known as Federated Average (FedAvg), to produce an improved global model, which is then redistributed to all clients for the next round of local training. This process is illustrated in Figure 2.2

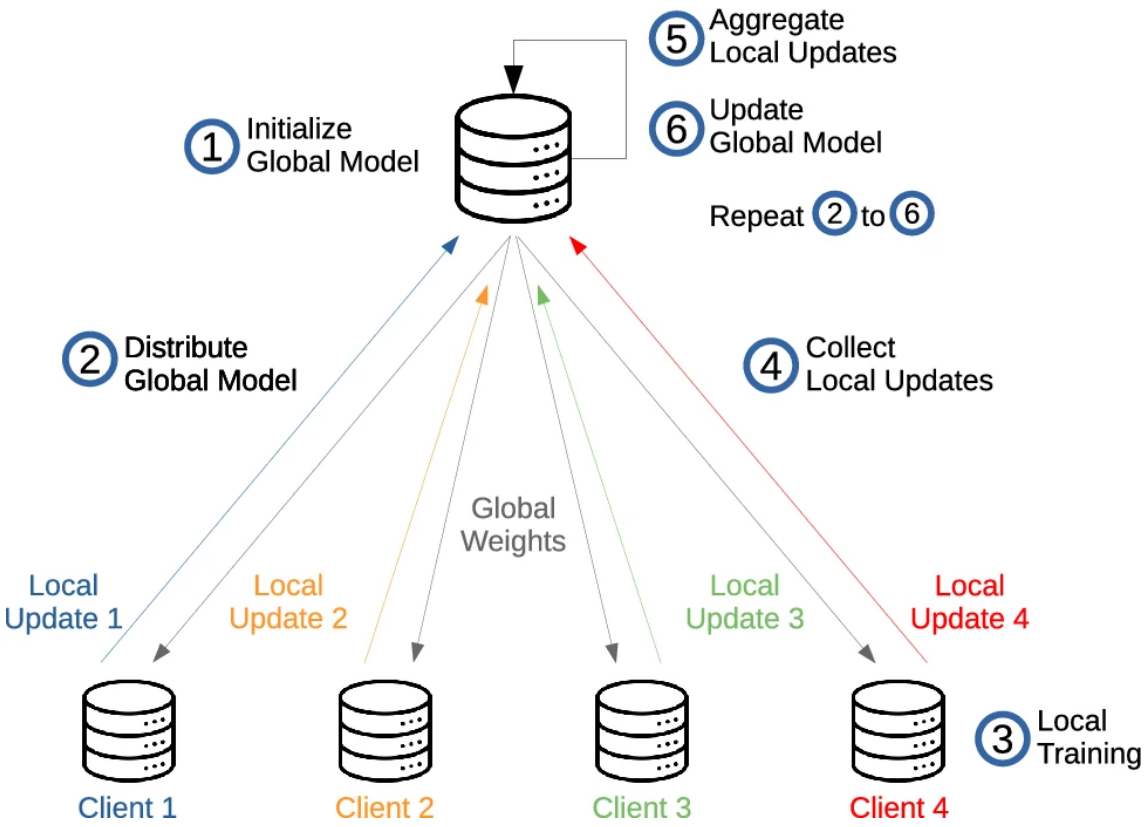


Figure 2.2. *Federated Learning Framework*

FL addresses the challenge of limited data availability, which along with computational resource constraints represents one of the two fundamental challenges in ML development. State-of-the-art models, whether deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformers, typically require large datasets and significant computing power. For instance, AlphaGo which marked a landmark in AI development since it defeated a world champion in the game of Go, relied on over 300,000 human-played games to achieve its breakthrough performance [14]. However, in many domains such as finance, healthcare, mobile applications, data is isolated and stored in separate systems or departments due to privacy regulations (e.g., GDPR [3]), competitive barriers, or logistical hurdles which prevents effective integration and sharing across the organization.

FL is categorized into cross-device and cross-silo settings [15]. Cross-device FL involves millions of low-resource clients, such as smartphones or IoT devices, generating local data like sensor readings or user interactions. Real-world applications include Google's Gboard, which enhances next-word prediction using on-device data while preserving privacy [16], and Apple's "Hey Siri" wake-word detection, which personalizes voice recognition without uploading sensitive audio [17].

In contrast, cross-silo FL involves a small number of high-resource organizations, such as hospitals or banks, collaborating to train a shared model while keeping sensitive data private due to strict regulations like HIPAA [4] or competitive barriers [18]. Data sharing may also be restricted within the same organization across regions due to legal constraints [15]. This setting features abundant computational power and large datasets but faces challenges like data heterogeneity, trust issues among competing entities, and high communication costs from complex model updates [18]. Promising applications include medical image analysis, where hospitals use FL to improve brain tumor segmentation in MRI scans [19] and financial crime detection where banks employ frameworks like Fed-RD to identify anomalous transactions privately [20].The differences between cross-device and cross-silo settings are illustrated in Figure 2.3.

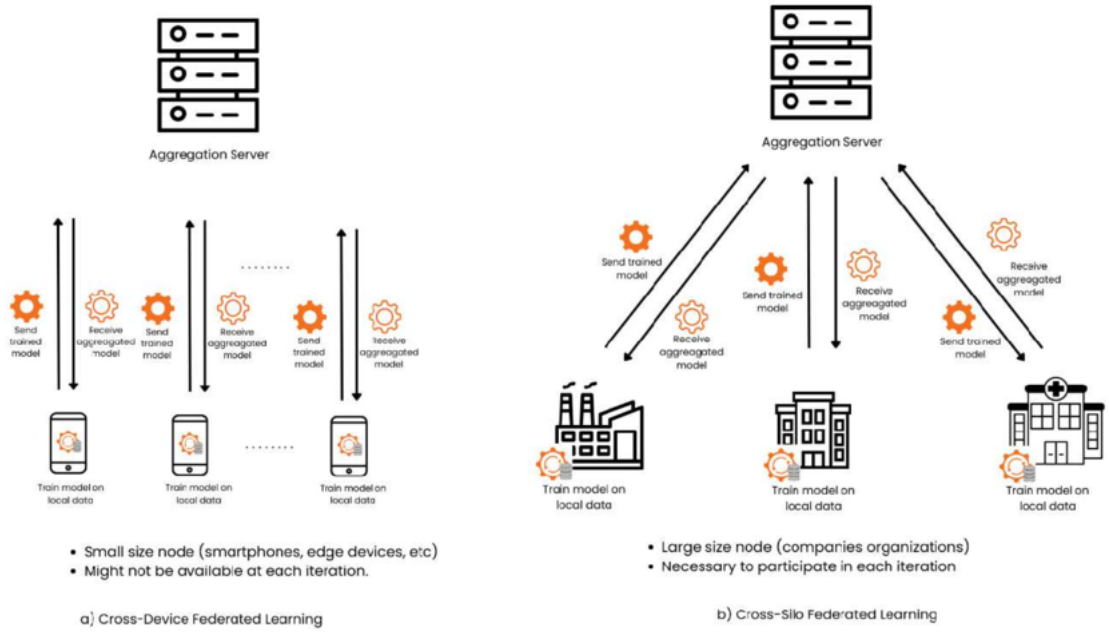


Figure 2.3. *Cross-Device vs Cross-Silo Federated Learning.*

.

2.3 Formulation of the Federated Learning Pipeline

This section presents a formal definition of the FL paradigm. A typical FL pipeline which is applicable to both cross-device and cross-silo scenarios works as follows:

1. **Client Selection & Model Initialization:** The server selects a subset of available clients based on predefined conditions. For example, in mobile applications like

Gboard, a client device is chosen only if it is idle, charging, and connected to Wi-Fi to minimize user disruption. The server then initializes the global model with parameters θ_0.

2. **Model Broadcast:** The initialized model is distributed to all selected clients.

3. **Local Training:** Each client trains the received model on its local dataset for a predefined number of epochs (either fixed or variable). This step generates a locally updated model for each client.

4. **Upload Local Updates:** Clients send their updated model parameters (e.g., weights or gradients) back to the server.

5. **Aggregation:** The server collects the updates from all participating clients and aggregates them to form a new global model.

6. **Broadcast Updated Model:** The server broadcasts the updated model weights back to clients (either the same set or a newly selected subset) for the next round of training.

7. **Repeat:** Steps 2–6 are repeated until a stopping criterion is met (e.g., a target global-model accuracy or a predefined number of federated rounds).

Aggregation algorithms are the cornerstone of FL, as they combine local client updates to optimize a global model. Although FedAvg can be considered as benchmark due to its simplicity and effectiveness, it struggles with data heterogeneity. This a common FL challenge that causes client drift, a phenomenon where local models, optimized on client's unique data, diverge from the global model which complicates the aggregation and degrades global model's performance.

Consequently, aggregation remains an active research area, with numerous algorithms proposed to address different FL hurdles [21]. For example, FedProx [22] extends FedAvg by adding a proximal term to the local loss function, penalizing large deviations from the global model to enhance convergence in non-IID settings. Similarly, SCAFFOLD [23] mitigates client drift by using control variates to align local updates with the global gradient direction, achieving faster convergence but with higher communication costs.

2.4 System and Data Challenges in Edge-Based Federated Learning

This section outlines the major challenges of deploying FL in real-world edge environments. Edge devices are computing units such as smartphones, smartwatches, and IoT sensors that are located near the data source and perform local processing instead of relying on centralized servers. These devices commonly operate under constraints such as limited connectivity, energy, and computational resources. The key challenges examined in this context include system heterogeneity, resource scarcity, and data heterogeneity.

2.4.1 System Heterogeneity

FL system must operate across a wide variety of devices, each with different hardware capabilities, operating systems, and network conditions. Edge devices vary in CPU performance, memory availability, energy constraints, and connectivity (e.g., 4G, 5G, Wi-Fi) [6].

2.4.2 Resource Scarcity

Edge devices are typically constrained in processing power, RAM, and battery life. These limitations restrict the size of models that can be trained locally and the frequency of communication with the server. Addressing these constraints is essential for scalable and energy-efficient FL systems [24].

2.4.3 Data Heterogeneity

Data non-IIDness is perhaps the most pervasive challenge across all FL deployments. Given its critical importance, we will now analyze it in greater depth. In statistics, the IID assumption means that the data is independent and identical distributed, and it is foundational to the ML concept. Independence means that samples do not influence each other, while identical distribution means that all samples originate from the same underlying probability distribution. An IID example is the well-known coin-toss experiment: each flip is independent, and a fair coin yields a constant head-vs-tail probability over time (e.g., $p(\text{head}) = 40\%$, $p(\text{tail}) = 60\%$). In contrast, HAR data are strongly temporally correlated, and each user's gait patterns vary according to factors such as age and gender [25], violating both independence and identical distribution. This distinction matters because standard ML practices, such as train/test splits rely on the assumption that unseen data follow the same distribution as the test set used for validation. In federated settings, non-IIDness is inevitable, since each client's data reflects unique habits and device characteristics [7].

2.5 Software Frameworks for FL on Edge devices

Selecting an appropriate framework is a critical decision for both research and practical deployment of FL. Although numerous frameworks have been proposed, the diversity of implementations and the absence of comprehensive comparative surveys make choosing one for real-world use non-trivial. For the purposes of this thesis, the framework must support cross-device configurations on heterogeneous edge hardware and enables efficient communication. Although the experiments will be conducted exclusively on Android smartphones, the chosen framework should offer portability to other edge or IoT platforms (e.g., Raspberry Pi) in order to meet FL broader requirements for the cross-device setting.

A quick review of popular Federated Learning (FL) frameworks showed that most frameworks, including TensorFlow Federated (TFF) [26] and FATE [27], are intended primarily for simulation environments. This design focus limits their applicability for on-device execution in edge scenarios. A recent and highly useful comparative study eval-

uated fifteen frameworks based on key criteria such as support for heterogeneous hardware, aggregation algorithms, and privacy-preserving features (e.g., weight encryption, differential privacy, secure aggregation) [28]. Among these, PySyft [29], FedML [30], and Flower [10] emerged as the most suitable for real-world edge deployments as they are compatible with various devices and operating systems

To guide the final selection, each of these three frameworks was examined across three primary dimensions:

- **Development Activity**: Is the framework under active maintenance and frequent updates?
- **Ease of Deployment**: How straightforward is it to install, configure, and deploy the framework on diverse devices?
- **Documentation Quality**: Does the framework provide clear, comprehensive guides and examples for various use cases?

PySyft was excluded due to its lack of Android support. This narrowed the focus to FedML and Flower. Flower was ultimately selected as the framework of choice because it provides comprehensive documentation and includes examples demonstrating FL deployment on Android devices. Although advanced privacy-preserving features such as secure aggregation and differential privacy are not fully supported, this limitation does not conflict with the research objectives, which primarily focus on addressing system and data heterogeneity.

Flower's key strengths also include its communication, language, and ML-framework agnostic design. It supports multiple communication protocols (gRPC, REST), programming languages (Python, Java, C++), and machine learning backends (TensorFlow, PyTorch, Keras). Moreover, Flower is compatible with a wide range of operating systems, including Linux, macOS, Windows, Android, and iOS, making it well-suited for our application.

2.6 Related Work and Research Gaps

Since its emergence as a research field, FL has attracted significant attention from the ML community. Most existing studies have focused on theoretical formulations and simulation-based experiments, addressing key challenges such as data heterogeneity (non-IIDness), communication efficiency, privacy, scalability, aggregation algorithms, and energy consumption. For instance, the authors in [31] introduced FedNAS to address the challenge of non-IID data distributions by using neural architecture search to identify the optimal model configurations depending on the used dataset. Communication efficiency has been improved through techniques like model update sparsification and quantization, which significantly reduce bandwidth usage [32]. Scalability issue which are often caused by intermittent or unreliable communication between edge devices and central servers have been tackled using asynchronous aggregation methods which allow clients to send updates independently [33]. Additionally, the energy impact of FL has been studied. For example,

[34] showed that in certain configurations, FL can consume up to twice the energy of centralized training, which highlighs the importance of using strategies that balance model accuracy with energy efficiency. Although these topics are critical to FL research, the vast majority of studies remain limited to simulation environments. Comparatively few works explore the practical implementation of FL in real-world settings.

In contrast to the simulation-based literature, a smaller but growing number of studies have explored real-world FL deployments on heterogeneous hardware. For example, the authors in [35] implemented the FedAvg algorithm on a testbed of five Raspberry Pi 4 devices connected via Wi-Fi using a TCP-based socket interface. Their experiments, are based on the CIFAR-10 dataset and examine factors such as client participation, local training epochs, data heterogeneity, and client mobility. These findings contribute useful observations for FL deployment in the IoT context.

In [36], the authors explore on-device FL using the Flower framework over Android smartphones and Nvidia Jetson TX2 embedded devices. This study also uses the CIFAR-10 dataset, but also extend the evaluation beyond model performance by examining energy consumption and network overhead in FL settings.

Authors at [37] proposed the FedIoT platform and FedDetect algorithm for anomaly detection on nine Raspberry Pi 4Bs, using MQTT for communication. This setup utilized the N-BaIoT dataset [38], which captures network traffic generated by malware attacks such as Distributed Denial of Service (DDoS) on IoT devices.

While these studies have successfully demonstrated FL execution on physical edge hardware, their experimental setups are limited in scope. The studies in [35] and [36] rely on CIFAR-10 which a standard image classification benchmark dataset that does not capture the complexities of real-world applications like HAR. As a result, it does not address critical domain-specific challenges such as class imbalance, sensor noise, or temporal data dependencies. In contrast, the work in [37] is more realistic in scope due to its use of IoT-relevant anomaly detection data. However, it focuses primarily on model performance, without analyzing other critical aspects such as energy consumption or network usage which are both essential for evaluating FL feasibility in constrained environments. This thesis aims to bridge this gap by implementing and evaluating an FL system using real-world, non-IID HAR data on heterogeneous Android smartphones, while also focusing on both energy efficiency and communication performance.

Chapter 3

Human Activity Recognition: Foundations, Challenges, and Data Considerations

This chapter establishes the theoretical foundations of the HAR problem and explains the rationale behind selecting this as the topic of research. But before presenting the core concepts, it is important to provide the formal definition. HAR is the process of automatically detecting and classifying human behaviors or activities using sensor data collected from devices such as smartphones and wearables. These activities can range from simple actions (e.g., walking, sitting, running) to more complex behaviors (e.g., cooking, driving, exercising), and may also include health-related monitoring tasks (e.g., gait analysis for fall detection or heart rate tracking for cardiac conditions). The real-world applications are diverse and have shown great promise, as demonstrated by their use in domains such as healthcare, fitness, and smart environments. This chapter is organized as follows: Section 3.1 presents the motivation for selecting HAR as a representative case study for evaluating FL in a practical and privacy-sensitive domain. Section 3.2 discusses traditional and FL-specific challenges in HAR. Section 3.3 describes the characteristics of HAR data on smartphones, including acquisition, preprocessing, and feature extraction. Section 3.4 provides an overview of the six public datasets used in this study.

3.1 Motivation for Selecting HAR as a FL Use Case

3.1.1 Why HAR is Suitable for Federated Learning on Edge Devices

The primary motivation for selecting a HAR dataset in this thesis is based on its suitability as a proof-of-concept application for FL on edge and IoT devices. It is as a highly promising technology, which is already integrated into numerous fitness and health applications [39] [40]. More crucially, it involves the processing of highly sensitive user data, making it well aligned with the privacy-preserving nature of the FL framework. Other viable FL applications on edge devices include Network Intrusion Detection Systems (NIDS). These have been extensively studied in recent research [41] [42] [43] and are supported by well-documented datasets such as Edge-IIoTset [42]. However, they typically require domain-specific expertise in cybersecurity and network engineering as well as access to specialized test environments, such as simulated industrial networks or malware detection frameworks. In contrast, a HAR application can be built and evaluated using

data collected from everyday devices, without requiring extensive knowledge in the network engineering field. Furthermore, HAR benefits from a wide range of publicly available and well-documented datasets, making it a practical and accessible choice for research in this field. Below are the key factors that support the selection of HAR for this thesis:

- **Feasibility of Real-World Deployment**
  HAR data can be collected passively in real time with minimal user burden, enabling scalable on-device processing. Moreover, the use of personal smartphones as data collection devices make it a feasible and no-cost approach as it requires no additional hardware.

- **Sensor Availability**
  Modern smartphones and wearables are equipped with accelerometers, gyroscopes, and physiological sensors (e.g., heart rate monitors), making them ideal platforms for HAR data collection [44].

- **Practical Relevance**
  HAR supports applications in healthcare (e.g., monitoring elderly patients), fitness tracking, rehabilitation, and safety systems (e.g., detecting driver drowsiness) [39] [40].

- **Privacy Requirements**
  Activity data can reveal sensitive information, such as daily routines, sleep patterns, or health conditions, necessitating privacy-preserving approaches like FL [45].

- **Alignment with FL**
  As discussed in Chapter 2, FL enables collaborative model training across distributed devices without sharing raw data, enhancing privacy and reducing communication overhead [46]. This makes FL particularly suited for HAR, where data is generated by heterogeneous edge devices [47].

3.1.2 Real-World Applications of HAR in Health Domain

This section provides a deeper exploration into real-world applications of HAR with a particular focus on the health domain. In health monitoring, HAR facilitates remote cardiac monitoring and arrhythmia detection by analyzing physiological signals such as heart rate and electrocardiogram data, enabling timely interventions for cardiovascular conditions [39]. In clinical diagnosis, HAR supports the detection of gait abnormalities associated with neurodegenerative diseases, such as Parkinson's or Alzheimer's, by analyzing movement patterns to aid in early diagnosis and treatment planning [46]. For mental health, HAR leverages wearable sensors to monitor stress through physiological markers like electrodermal activity (EDA) and skin temperature, as demonstrated in studies using multimodal data to classify self-reported stress and mental health states in college students with high accuracy [48]. In safety applications, HAR plays a critical role in detecting driver drowsiness using biosignals, such as heart rate and breathing rate, measured by wireless wearables like the BioHarness 3, to alert drivers and prevent accidents [49]. Collectively,

these diverse applications highlight HAR's potential to enhance individual well-being, support clinical decision-making, and promote safety in real-world environments.

3.2 HAR Challenges and Its Integration with FL

Deploying HAR systems in real-world settings presents significant challenges, both inherent to HAR and specific to FL. These challenges can be broadly categorized into two groups: intrinsic challenges inherent to HAR itself, and challenges specific to deploying HAR in FL setting.

3.2.1 Intrinsic HAR Challenges

HAR datasets are inherently non-IID (refer to Chapter 2.4 for a detailed analysis of the non-IID problem). Sensor signals vary significantly depending on the user, the placement of the device on the body, and the specific hardware used. As a result, the same activity can produce different signal patterns across individuals and contexts, making generalization difficult and increasing the risk of overfitting to user- or environment-specific features [47].

Researchers have further characterized this non-IID nature using the concepts of:

- **Intraclass Variability**
  The same activity (e.g., "walking") can appear very differently depending on the user's gait, device placement, or walking speed. This variability complicates classification and increases the likelihood of false negatives, requiring robust and adaptable models.

- **Interclass Similarity**
  Different activities (e.g., brushing teeth vs. eating) may produce similar sensor signals. This overlap introduces ambiguity and misclassification risks. Such challenges can be addressed, to some extent, by incorporating multi-sensor or multimodal data (e.g., using sensors on the wrist, ankle, and chest) [9].

Another major challenge is class imbalance. Most HAR datasets are dominated by common, low-effort activities (e.g., sitting, standing), while rarer or more complex behaviors (e.g., stair climbing, jumping) are underrepresented. This imbalance biases the model toward majority classes, reducing performance on infrequent but often more critical activities. Finally, HAR systems are affected by concept drift; that is the phenomenon whereby user behavior and activity patterns change over time due to factors such as aging, injuries, seasonal variations, or lifestyle changes. Without mechanisms for continual or online learning, static HAR models degrade in performance as the underlying data distribution shifts [9] [50].

This thesis addresses intraclass variability, interclass similarity, and class imbalance, as these are intrinsic characteristics of HAR datasets (the datasets used in this study are no exception). However, addressing concept drift requires temporary data to track changes in activity patterns over time, which falls outside the scope of this work.

3.2.2 FL Challenges in HAR Deployments

Applying Federated Learning to HAR introduces an additional set of challenges:

- **Need for Personalization**
  Generic FL models trained across a diverse population often fail to capture the nuances of individual movement patterns, particularly for groups such as older adults, individuals with disabilities, or those engaged in specialized activities (e.g., sports training or physical therapy). HAR applications demand personalized models, which poses significant technical challenges in the federated paradigm [50].

- **Amplified Concept Drift**
  Although FL offers privacy advantages, it also inherits the same vulnerability to concept drift as centralized models. However, adapting to drift in FL is more complex, as data remains decentralized. Continual learning must be implemented without accessing historical data and with minimal communication overhead. Without drift-aware mechanisms, FL-HAR models risk becoming outdated over time [50].

- **Privacy Leakage Risks**
  While FL avoids direct transmission of raw data, model updates can still leak sensitive behavioral information (e.g., gesture-specific gradients). In domains like healthcare, where data sensitivity is paramount, untrusted aggregation servers may pose a risk. This requires the use of secure aggregation protocols or other privacy-enhancing technologies to ensure robust protection [51].

- **Device Heterogeneity**
  Edge devices involved in FL (e.g., smartphones, smartwatches, fitness trackers) vary widely in computational power, memory and battery life. Although smartphones may handle training reasonably well, resource-constrained devices may struggle to complete even a single training round. These disparities lead to issues such as client dropout, slower convergence, and poor global model quality [47].

While challenges such as the need for personalization, amplified concept drift, and privacy leakage risks represent important areas of research in FL, they are not the focus of this thesis. Instead, this work specifically addresses the challenge of device heterogeneity. The primary objective is to investigate how variations in computational resources across edge devices impact FL performance, and how these constraints can be managed to enable efficient and reliable training.

3.3 Human Activity Data Characteristics and Preprocessing

A typical HAR pipeline consists of data acquisition, feature preprocessing, feature extraction and then the training step. This section, will provide details for each part of the process except the training step, which is examined in depth in the implementation and experimentation section of this thesis (Chapter 4.1).

3.3.1 Sensor Modalities and Data Acquisition Methods

In HAR, sensor data is acquired from either external wearable devices (e.g., heart rate monitors, electrodermal sensors connected via Bluetooth) or from embedded sensors in smartphones and smartwatches (e.g., integrated accelerometers and gyroscopes). The data can then be processed in three main ways. First, it can be processed locally on the device itself using its CPU resources. Second, it can be streamed to a centralized application server for processing, which is useful when a single device lacks sufficient data or computational power. Third, as in the case of this thesis, FL can be used. This approach allows each device to train a local model independently, while only sharing model updates. These updates are then aggregated across to form a single global model.

Typically used sensors are mechanical like accelerometers which measure the acceleration along x,y and z axis, and gyroscopes which measure the rotational movement (roll, pitch, yaw). There are also Physiological Sensors like Photoplethysmography (PPG) which detect blood flow for heart rate monitoringl [52] [53]. Finally there are case where Biochemical Sensors are utilized which they monitor glucose or hydration in advanced wearables [44].

Currently, all smartphones are equipped with accelerometers and gyroscopes and there is a plethora of accessories available that augment smartphones sensoring capabilities which can further enhance the application range of smartphone-based HAR.

3.3.2 Preprocessing of sensor data

Preprocessing is a critical step in the ML pipeline, especially when working with sensor data, which is often noisy, inconsistent, and incomplete. The goal is to transform raw sensor readings into a clean, structured format suitable for analysis and model training. Key preprocessing steps include:

- **Data alignment and interpolation**
  The step typically involves aligning sensor data and interpolating missing values, as sensor measurements are rarely uniformly sampled especially on general-purpose devices, where the sampling rate acts more as a suggestion to the operating system rather than a strict constraint [54]. Additionally, it is often necessary to synchronize measurements across multiple sensors (e.g., gyroscope and accelerometer) so that rotation matrices can be accurately applied to acceleration vectors. This is commonly achieved using linear interpolation to fill in missing values in all sensor streams [55].

- **Noise filtering**
  Once data is aligned, noise filtering is applied to smooth the signal. Common techniques include median filtering and low-pass filtering (LPF) to attenuate high-frequency noise. Research suggests that setting a low-pass filter cut-off frequency at 25 Hz retains over 99% of the relevant information when starting from a 100 Hz signal, making it sufficient for most HAR applications while also conserving energy [56].

- **Data normalization**
  Following denoising, data normalization is often performed to bring all values to a common scale. This helps account for inter-individual differences and sensor variability, allowing the ML model to focus more on patterns rather than raw magnitudes. Common normalization techniques include Min-Max scaling (0 to 1) and standardization (zero mean and unit variance).

- **Data segmentation**
  A particularly important step in HAR preprocessing is data segmentation, which involves dividing continuous sensor data into time windows that are likely to contain discrete activities. This process enables the mapping of time-series signals to labeled actions (e.g., walking, sitting) and is often referred to as activity detection. Although manual segmentation is possible, it is prone to frequent errors and also impractical due to the sheer volume of data typically involved in HAR tasks. Instead, automated techniques like sliding windows (with or without overlap) or energy-based methods (thresholding based on signal magnitude) are commonly used to segment the data [57].

- **Feature extraction**
  Once segmented, feature extraction is performed to convert raw time-series data into meaningful descriptors. There are two general approaches: one is handcrafted and the other automated [58]. Handcrafted feature extraction relies on domain expertise to derive informative metrics from the data, such as time-domain features (mean, standard deviation), frequency-domain features (FFT, Power Spectral Density), or application-specific features (zero-crossing rate, peak acceleration) [59]. For instance, in [49], heart rate and breathing rate signals were transformed using FFT, and their corresponding PSD values were used as input to a neural network model. In contrast, deep learning models such as DNNs or CNNs often learn relevant features directly from raw time-series data. These models are capable of identifying complex patterns automatically and typically generalize well across related tasks [57].

Although this thesis does not experimentally evaluate different preprocessing schemes, a diverse set of datasets has been selected to represent various preprocessing levels from raw sensor data to manually extracted features. These datasets are presented in detail in Chapter 3.4.

3.3.3 Case Study: Preprocessing Pipeline in the UCI Smartphone HAR Dataset

A representative example of an HAR preprocessing pipeline can be found in the UCI Smartphone HAR dataset [60]. In this dataset, data was collected from triaxial linear accelerometers and gyroscopes at a sampling rate of 50 Hz. To reduce noise, the signals were first processed using a median filter followed by a third-order Butterworth low-pass filter [61]. Based on their analysis, a cut-off frequency of 20 Hz was chosen, as it preserved over 99% of the signal energy which alignes with the findings mentioned from a

revious study [56]. For segmentation, the data was divided into fixed-width 2.56-second sliding windows with a 50% overlap to capture sufficient temporal context while maintaining data continuity. From the raw triaxial sensor readings, additional features were derived, including Euclidean magnitudes and time derivatives. Further feature engineering was performed by transforming time-domain signals into the frequency domain using the Fast Fourier Transform (FFT). Additional time-domain features such as mean, correlation, and autoregression coefficients were also computed. In total, these steps yielded a 561-dimensional feature vector for each windowed segment, forming the input to the machine learning model.

To evaluate the performance of FL in HAR systems, diverse datasets are essential for capturing a wide range of activities, sensor modalities, and real-world conditions. The following section introduces six publicly available datasets used in this thesis, detailing their sensor configurations, activity sets, and preprocessing methods. Each dataset offers unique characteristics such as different preprocessing levels, feature extraction techniques, and activity classification schemes which enable a robust evaluation of FL performance across both controlled and real-world HAR scenarios.

3.4 Overview of HAR Datasets Used in This Thesis

The datasets listed below range from raw sensor signals such as those found in the MotionSense dataset, to thoroughly processed feature vectors, like the 561-dimensional features used in the UCI HAR using Smartphones dataset. The data collection hardware also varies significantly, from general-purpose smartphones (e.g., Samsung Galaxy S2, iPhone 6) to specialized wearable devices like the ActiGraph GT3X+, which is specifically designed for health-related monitoring. The applicability of FL becomes clear even in these basic HAR scenarios, as many datasets and devices capture sensitive physiological data, such as ECG and heart rate. By training models locally on user devices and avoiding the need to transmit raw data, FL reduces the risk of exposing personal health information.

- **HARSense**
  The HARSense dataset comprises triaxial accelerometer and gyroscope data collected from smartphones (Poco X2 and Samsung Galaxy A32s) mounted on the waist and front pockets of 12 subjects aged over 23 years and weighing over 50 kg. It includes six activities of daily living (ADLs): walking, standing, sitting, running, upstairs, and downstairs, performed in a laboratory setting except for running, which occurred on a football playground. The dataset provides raw triaxial acceleration (linear and gravity), rotational rate, and rotational vector at an unspecified sampling rate. One notable limitation is the lack of documentation regarding preprocessing steps, which may affect reproducibility and comparability with other datasets. [62]

- **HAR Using Smartphones**
  The UCI Human Activity Recognition Using Smartphones dataset available via the UCI Machine Learning Repository includes triaxial accelerometer and gyroscope data

collected at 50 Hz from a waist-mounted Samsung Galaxy S II smartphone carried by 30 subjects. The preprocessing pipeline is discussed Chapter 3.3.3 [60]

- **PAMAP2**
  The PAMAP2 Physical Activity Monitoring dataset, accessible through the UCI Machine Learning Repository, comprises data from three Colibri wireless inertial measurement units (IMUs) sampled at 100 Hz, placed on the dominant wrist, chest, and ankle, and a BM-CS5SR heart rate monitor sampled at approximately 9 Hz. Each IMU includes two 3-axis accelerometers, a 3-axis gyroscope, and a 3-axis magnetometer providing 52 raw sensory attributes with missing values marked as NaN. Nine subjects (8 males, 1 female aged 27.22 $\pm$ 3 years old) performed 12 protocol activities (lying, sitting, standing, walking, running, cycling, Nordic walking, ironing, vacuum cleaning, rope jumping, ascending stairs, descending stairs). Data is segmented using 5.12-second sliding windows with 1-second shifts (approximately 0.8 overlap), with features extracted in time (e.g., mean, variance) and frequency domains (e.g., FFT-based energy) for acceleration, and mean and gradient for heart rate [63].

- **MotionSense**
  The MotionSense dataset consists of time-series accelerometer and gyroscope data from iPhone 6s smartphones, collected from 24 subjects performing six activities: walking, jogging, sitting, standing, upstairs, and downstairs. Data is sampled at 50 Hz, with sensors typically placed in pockets or hands, reflecting real-world variability. The dataset provides raw inertial data without precomputed features [64].

- **MHealth**
  The MHealth dataset, hosted by the UCI Machine Learning Repository, comprises multimodal sensor data from 10 subjects performing 12 activities, including walking, running, jumping, cycling, and sedentary tasks. Sensors include triaxial accelerometers, gyroscopes, and magnetometers placed on the chest, right wrist, and left ankle, with two-lead electrocardiogram (ECG) data, all sampled at 50 Hz. Except for classical activity recognition MHealth is also ideal for clinical HAR applications, such as cardiovascular health monitoring and mobility disorder detection [65].

- **PhysioNet Acceleration Data**
  This dataset includes triaxial accelerometer data collected at 100 Hz from a thigh-worn sensor (Actigraph GT3X +) worn by 20 subjects during walking, stair climbing, and driving. The dataset provides raw acceleration signals without precomputed features [66].

Chapter 4

System Architecture and Experimental Setup

This chapter presented the infrastructure and experimental setup developed to evaluate the HAR problem using FL. The two main design considerations that guided the development were scalability and the ability to support rapid experimentation.

Scalability was prioritized due to the challenges introduced by system heterogeneity in on-device FL deployments, as discussed in Section 2.4. By leveraging Flower's modular and ML-agnostic architecture, the system was designed to scale across a wide range of heterogeneous clients, such as smartphones and Raspberry Pi devices, while maintaining compatibility with various machine learning frameworks, including TensorFlow, PyTorch, or event custom-built solutions. Although the experiments in this thesis are conducted solely on Android devices running TensorFlow Lite (TFLite) [11], it is important for the infrastructure to remain compatible with a broad range of devices and platforms in order to ensure that the system is extensible to other use cases and not narrowly tailored to this specific application.

Rapid experimentation was the second major design focus since as demonstrated in Chapter 5, the space of hyperparameters and evaluation metrics is both large and multidimensional. To support this, the system provides tools to easily create and modify datasets according to experimental needs, build and port TFlite models with varying architectures, and systematically log critical metrics such as model accuracy, training time, energy consumption, and network utilization. Additionally, the infrastructure automates the generation of informative visualizations including accuracy/loss plots, convergence speed graphs, and energy consumption diagrams which is highly useful tool that enables faster insights and iterative tuning.

4.1 System Overview and Design Considerations

This section provides a high-level overview of the FL system developed for this thesis. The objective is to enable a group of Android smartphones to collaboratively train a shared machine learning model under the coordination of a central server without exchanging raw user data. The experimental setup consists of:

- A central FL server, implemented in Python, running on a personal laptop (MacBook Pro 2012, Intel i5, 16GB RAM).

- Five Android smartphones, each running a custom FL application and acting as a client.
- A local Wi-Fi network used to exchange model updates between the server and clients.

The FL process follows the standard server–client paradigm, as shown in Figure 2.2, where the server sends a global model to the clients, clients train locally, and then send model updates back for aggregation. However, a key distinction is that instead of requiring clients to store their own local datasets, both training and test data are transmitted by the server during the initial discovery phase, when it waits for clients to connect. This approach enables centralized control over dataset allocation and support faster experimentation. Additionally, to maintain consistent performance tracking throughout the experiment, each client evaluates the updated global model on its local test data at the end of every training round. Figure 4.1 illustrates the testbed used during experiments.

Figure 4.1. *Federated Learning Testbed*

The Android smartphones used as clients in the experiments vary in hardware specifications, providing a realistic representation of system heterogeneity. Table 4.1 summarizes the key characteristics of each device used in the testbed.

Table 4.1. *Specifications of Mobile Devices Used on the Experiments*

Device Name	Release Date	CPU	RAM	Android
Realme GT Neo 2 (RMX3370)	2021	Snapdragon 870	8 GB	12
Samsung Galaxy A5 (SM–A510F)	2016	Snapdragon 615	2 GB	7
Samsung Galaxy J7 (SM–J730F)	2017	Exynos 7870	3 GB	8
Samsung Galaxy S10+ (SM–G975F)	2019	Snapdragon 855	6 GB	12
Xiaomi Redmi 9C (M2006C3MNG)	2020	Helio G35	2 GB	10

4.1.1 Technical Considerations

The system's design was significantly impacted by an infrastructure limitation. Specifically, TFLite which is used for on-device training, does not currently support programmatically setting model weights. As a result, during each aggregation round, the server must recompile and transmit the entire TFLite model to the clients, rather than simply sending the updated weights for clients to apply to their existing models.

This limitation introduces both time and communication overhead. Converting a TensorFlow model to the TFLite format takes approximately 15–20 seconds which is often longer than the actual training time on the client device. Additionally, transmitting the entire TFLite model significantly increases data transfer. For example, a model with around 300 parameters results in a file size of roughly 42KB, whereas transmitting only the weights would require just 1.2KB. While this overhead may be negligible for large models where training time dominates and TFLite's metadata is relatively minor, it becomes a critical bottleneck in edge scenarios involving small models and short training durations. Therefore, both the conversion delay and communication cost are explicitly accounted for in the experimental analysis.

4.2 Android Client Application: Design and Implementation

The client component is implemented as an Android application written in Java, based on Flower's official example for FL on Android devices [67]. To join the FL process, the user must open the application, type server's IP address and port number, and then press the "Connect" and "Train" button. This action signals to the server that the device is ready to participate in training. Then once the predefined number of clients have connected, the server will initiate the FL process. Android application's user interface is illustrated in Figure 4.2.

As previously noted, TFLite was selected as the ML framework for this implementation, despite its current limitation of not allowing programmatically setting model's weights. The primary motivation behind this choice was TFLite's extensive documentation, and broad cross-language compatibility which includes support for Java, C++, Python, and Swift. This makes it well-suited for heterogeneous system environments. Additionally, TensorFlow offers reliable tools for converting standard TensorFlow models into the TFLite format.

Alternative frameworks such as PyTorch Mobile and Deeplearning4j were also considered. However, PyTorch Mobile currently lacks on-device training capabilities and it is solely used for inference using a pretrained model. In contrast, while Deeplearning4j does support training, has a relatively small user base, limited documentation, and only support only the Java language which making it less suitable for heterogeneous environments.

The client application is composed of three key Java classes:

- `MainActivity.java`: This class handles the main activity lifecycle, manages communication with the server via gRPC, and orchestrates the training and evaluation processes.

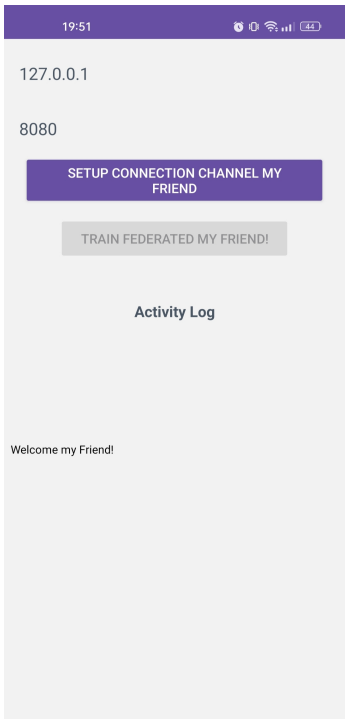


Figure 4.2. *Android Client Application*

- `DataLoader.java`: This class is responsible for loading and managing the dataset received from the server. It parses the feature and label arrays into a PyTorch-like format, enabling batched data access. Batching is essential for efficiency, especially when large datasets are used which cannot fit entirely in memory. It also acts as a training hyperparameter which influence both performance and convergence speed. Batches are retrieved incrementally using the `.next()` method. Example usage:

```
trainLoader = new DataLoader(X_train, y_train, batchSize, numFeatures);
BatchOfData trainBatchOfData = trainLoader.next();
float[][] X_train_batch = trainBatchOfData.getX();
float[] y_train_batch = trainBatchOfData.getY();

testLoader = new DataLoader(X_test, y_test, batchSize, numFeatures);
BatchOfData testBatchOfData = testLoader.next();
float[][] X_test_batch = testBatchOfData.getX();
float[] y_test_batch = testBatchOfData.getY();
```

TensorFlowModelWrapper.java: This class reconstructs the model on the client using the architecture metadata and weights received from the server. The metadata includes input size, number of hidden layers, hidden layer size, and number of outputs. The model is then compiled into a form usable by TFLite. To mimic a PyTorch-like format, this class provides fit() and evaluate() methods:

- `fit()`: Takes a training `DataLoader` and the number of epochs as input, and performs model training.

- `evaluate()`: Takes a test `DataLoader` and returns performance metrics, including accuracy, true labels, and predicted labels.
- Example usage:

```
TransferLearningModelWrapper modelWrapper =
    new TransferLearningModelWrapper(formatModelFile(modelFile),
                                     inputLayer, hiddenLayer, outputLayer);

// Training
modelWrapper.fit(trainLoader, n_epochs);

// Evaluation
Metrics metrics = modelWrapper.evaluate(testLoader);
float accuracy = metrics.getAccuracy();
float[] yTrue = metrics.getYtrue();
float[] yPred = metrics.getYpred();
```

4.3 Server-Side Architecture

The server was implemented in Python using the Flower framework, an popular open-source platform for FL. One of Flower's core strengths lies in its flexibility since it allows custom server-side logic to be defined without altering the framework's underlying source code. The implementation follows the standard FL cycle described in Section 4.1.1 and integrates additional components for training control, monitoring, and experiment automation.

The server controls a combination of ML hyperparameters (such as the number of local training epochs) and system-level parameters (such as the minimum number of clients required to begin a training round). ML hyperparameters affect the learning dynamics and model behavior, system parameters determine how the FL process is orchestrated in distributed settings. The key parameters configurable at the server include:

- **Number of Local Epochs:** The number of epochs each client uses to train on its local dataset before sending updates.
- **Number of Federated Rounds:** The total number of global training rounds to be executed.
- **Minimum Clients per Round:** The minimum number of clients required for training to proceed.
- **Target Clients per Round:** The number of clients the server attempts to gather before initiating a training round.

For example, if the minimum client threshold is set to 3 and the target is 5, then training will begin once 5 clients are available, but can continue as long as at least 3 remain active.

Training data is prepared by the internal Data Factory module and structured according to a standardized directory format (as described in Section 4.4):

```
dataset_name/
 X_train/
    partition_0.txt
    partition_1.txt
    ...
 X_test/
    partition_0.txt
    partition_1.txt
    ...
 y_train/
    partition_0.txt
    partition_1.txt
    ...
 y_test/
    partition_0.txt
    partition_1.txt
    ...
```

Each partition_idx.txt file corresponds to a specific client, with the partition index indicating the client ID. This structure allows the server to map data partitions directly to participating clients during the federated training process.

Another important hyperparameter is the model configuration, which is handled by the Model Factory which is an internal component of our infrastructure that automates the construction of DNN models. In this thesis, experimentation was focused exclusively on DNNs, as they provided sufficient performance while remaining efficient to train on resource-constrained devices. Early experimentation confirmed that simple DNNs were able to achieve satisfactory performance even with limited client data, which aligned with the thesis goals.

The Model Factory allows the user (or the server) to define arbitrary DNN configurations, including the number of hidden layers and the size of each hidden layer. Models are initialized using the Adam optimizer with a predefined learning rate and can be created with either randomly initialized weights or custom weights. The ability to load custom weights into a newly created model is what enables the core FL functionality, since after each round, the server can compile a new model loaded with the aggregated weights and send it to the clients for the next round of training.

The final key server-side component is the Statistics Collector. Once the federated learning process is complete, this module performs two essential tasks. First, it records all relevant metrics as outlined in Section 4.5 into a csv file. Second, it generates a set of visualizations and summary statistics that enable quick evaluation of each experiment's outcome without the need for manual data analysis. This level of automation is particularly important given the large number of hyperparameter combinations explored in our

experiments.

The following outputs are automatically generated:

- `accuracy_loss_plot.png`: Accuracy and loss of the global model on the test set across all federated rounds.
- `performance_scores.png`: Final model performance summary, including the confusion matrix, precision, recall, F1-score, and accuracy per class and overall.
- `training_time_plot.png`: Training time per round and cumulative training time.
- `charge_drop_plot.png` and `voltage_drop_plot.png`: Per-device and per-round battery charge and voltage drops.
- `useful_stats.txt`: Summary statistics, including the number of rounds, maximum number of participating clients, number of test samples, client participation per round, and relevant network metrics.

4.4 Dataset Processing and Partitioning Infrastructure

The final component of the infrastructure is the Data Factory, which is responsible for loading, processing, and formatting datasets in a way that the server can understand. The first step involves creating dataloaders for each dataset. These dataloaders convert the raw dataset into a unified .csv format, where all features are included and the activity labels are stored in a dedicated "activity" column. This unification is essential because datasets vary significantly in their original formats. For example, MHealth stores a separate .txt file for each subject, while MotionSense organizes data into a directory per subject, with separate .txt files for each activity type. A consistent format simplifies the preprocessing and integration.

After loading, the dataset is partitioned using a function called split_to_client, which takes a train-test split of the entire dataset and partitions it randomly among N clients. The random seed can be specified to ensure reproducibility. The user can also choose between two FL testing configurations: a single, shared test set used by all clients, or individual test sets assigned to each client. The pipeline supports both configurations by assigning the correct test set to each client during the data distribution process.

The Data Factory also provides control over the number of samples allocated to each client, enabling the simulation of real-world scenarios where data volumes are unevenly distributed across clients.

Finally, the function class_imbalance is used to simulate class imbalanceness,which is an important aspect of non-IID experiments. Since original datasets are not perfectly balanced, normalization is first applied via upsampling or subsampling. Then, user-defined imbalance ratios are used to simulate varying label distributions across clients.

Once partitioned, the client data is stored using the save_client_data function in the directory structure that is expected by the server. The following example illustrates how the MotionSense dataset is loaded, processed, and prepared for use in a FL experiment:

- First, the dataset is loaded, and all features are scaled using the Standard Scaler to ensure consistent value ranges across inputs.
- The data is then split into training and testing sets using a 70%-30% ratio. Since this process involves shuffling the data, a random seed is used to ensure reproducibility across runs.
- Next, the training set is partitioned among 5 clients to simulate a federated setup.
- To simulate class imbalance, each client receives a version of the dataset where one of the six activity classes is reduced to 20% of its original size. This is done in a round-robin fashion, meaning that each client has a different class underrepresented in order to ensure the distribution diversity of labels across clients. Before applying the imbalance, the dataset is first balanced using the downsampling technique to ensure equal class distribution.

```
# load MotionSense dataset
df6 = dataloading.load_data6()

# apply train test split
X_train, y_train, X_test, y_test, labels = dataloading.train_test_split(
    df6,
    test_size=0.3,
    scaler_type="standard",
    should_map_labels=True,
    random_seed=42
)

# partition data to 5 clients
client_data = dataloading.to_client(
    data=(X_train, y_train, X_test, y_test),
    max_clients=5
)

# define class ratio list
class_ratio_list = [
    [0.2, 1.0, 1.0, 1.0, 1.0, 1.0],
    [1.0, 0.2, 1.0, 1.0, 1.0, 1.0],
    [1.0, 1.0, 0.2, 1.0, 1.0, 1.0],
    [1.0, 1.0, 1.0, 0.2, 1.0, 1.0],
    [1.0, 1.0, 1.0, 1.0, 0.2, 1.0]
]

# apply the class ratio list
client_data_im = []
```

```
for idx, (X_train, y_train, X_test, y_test) in enumerate(client_data):
    class_ratio = class_ratio_list[idx]
    X_train_im, y_train_im, X_test_im, y_test_im = dataloading.class_imbalance(
        (X_train, y_train, X_test, y_test),
        class_ratio,
        balance="downsampling"
    )

# save client data
dataloading.save_client_data(client_data_im,"test", labels)
```

4.5 Experimental Setup and Measurement Metrics

To gain meaningful insights into the behavior of HAR under a FL setup, a comprehensive set of metrics is measured to reflect model performance, with special emphasis placed on the unique challenges of deploying FL on edge devices in the context of HAR, as discussed in Sections 2.4 and 3.2. These challenges include system heterogeneity, energy constraints, network availability, and the non-IID nature of HAR datasets.

The collected metrics are grouped into four main categories: performance, training time, energy consumption, and network usage. As previously noted, all of these metrics are automatically gathered and logged by the Statistics Collector component of the infrastructure.

4.5.1 Model Performance Metrics

The performance evaluation of each model will be based on confusion matrix, accuracy, precision, recall, and F1-score which is a standard set of classification metrics. The confusion matrix is a crucial evaluation tool for classification problems. It visualizes the model's predictions by comparing them with the actual ground truth labels which allows the identification of patterns in class confusion. For example, a model may more frequently misclassify the activity "running" as "walking upstairs" rather than as "standing". The confusion matrix reveals such tendencies by displaying the counts of:

- **True Positives (TP):** Correct predictions where the model correctly identifies a sample as belonging to a particular class.
- **True Negatives (TN):** Correct rejections where the model correctly identifies a sample as not belonging to a particular class.
- **False Positives:** Incorrect predictions where the model incorrectly classifies a sample as belonging to a class it does not.
- **False Negatives (FN):** Missed predictions where the model fails to identify a sample that actually belongs to a class.

The following evaluation metrics can be derived from the confusion matrix:

- **Accuracy:** The proportion of correct predictions (both true positives and true negatives) out of all predictions. It is the most commonly used metric for evaluating classification models. However, in the presence of class imbalance which is common in HAR datasets the accuracy metric can be misleading. For example, if a particular activity dominates the dataset, a model might achieve high accuracy simply by predicting that class most of the time, while performing poorly on minority classes.

- **Precision:** The proportion of correct positive predictions out of all positive predictions made by the model. It reflects the model's ability to avoid false positives.

- **Recall:** The proportion of correct positive predictions out of all actual positives. It reflects the model's ability to capture all relevant instances.

- **F1-Score:** The harmonic mean of precision and recall. It provides a balanced view of a model's performance, which is especially useful in cases where there is an uneven class distribution or where both false positives and false negatives carry significant costs. The F1-score can be treated as a single metric that effectively captures a model's overall classification performance, particularly in scenarios where accuracy alone may be misleading.

4.5.2 Training Time Measurement

FL convergence speed is measured by the per-round training time and the total training. As explained in Section 4.1.1, TFLite introduces an additional time overhead, since the model must be recompiled at the start of each round. To account for this, two distinct training time metrics are reported. The first, referred to as training time, captures only the time spent on the actual training process, excluding compilation. The second, training time with compilation, includes the time required for model recompilation. This distinction is essential for identifying differences between delays caused by infrastructure limitations and those inherent to the training process itself.

4.5.3 Energy Consumption Estimation

Energy consumption on Android devices is measured using the BatteryStats API. An initial alternative considered was the built-in battery usage profiler available in Android Studio. However, this tool has several limitations, since it lacks support for older Android versions and requires a wired connection with the development PC which introduces bias by charging the device during profiling. These limitations make it unsuitable for fast, repeatable, and scalable experimentation.

Hardware power monitors were also evaluated as an option. Although they offer high accuracy and fine-grained energy measurements, their use was deemed impractical. Each device must be physically wired to the monitor, which significantly limits scalability. Moreover, these monitors measure the total power consumption of the device, including background processes, Wi-Fi activity, and screen usage which introduces the same variability issues found in software-based methods.

Given these constraints, the BatteryStats API was selected for its practicality and ease of integration. It allows automated, scalable energy measurement directly within the application code and provides access to key energy-related metrics, including:

- Battery voltage (Volts)
- Battery Capactiy
- Charge (mAh)
- Instantaneous and average current (mA)
- Battery temperature

This approach supports a wider range of Android versions, does not require any external hardware, and fits well with our goal of fast and repeatable experimentation.

To improve measurement accuracy and reduce noise, several controls were applied during the experiments. First, all unnecessary background applications and services were disabled to minimize extraneous energy consumption. Next, Airplane mode was enabled to eliminate power usage from cellular connections, while Wi-Fi was kept active to support federated communication. Finally, the screen brightness was set to its minimum level to reduce display-related power draw. Although these adjustments do not eliminate all external influences, they help ensure that energy measurements are standardized across devices and experimental runs.

While not as precise as hardware-based methods, this approach is considered a reasonable tradeoff between accuracy and scalability, making it well-suited for the experimental requirements of this study.

Since the BatteryStats API provides real-time readings of battery charge (in mAh) and voltage (in volts), these values can be used to compute the energy consumption for each FL communication round using the following formula:

$$Energy(round; device) = ChargeDrop(round; device) \cdot AvgVoltage(round; device) \quad (mWh) \tag{4.1}$$

Where the charge drop for each round and device is calculated by recording the battery charge at the start and end of the round and taking the difference. The average voltage for the round is computed as the mean of the voltage values recorded at those two time points.

Then, the total energy consumption for a given device is obtained by summing its energy usage across all communication rounds, as shown in Equation 4.2:

$$E_{\text{total}}(d) = \sum_{r=1}^{R} \text{Energy}(r, d) \tag{4.2}$$

where:

- $E_{\text{total}}(d)$ is the total energy consumed by device d,

- Energy(r, d) is the energy used by device d during round r,
- R is the total number of communication rounds in the Federated Learning process.

4.5.4 Network Quality Monitoring

Several network-related metrics are measured throughout the training process to assess how network conditions affect FL performance. These metrics offer a comprehensive view of each client's network environment, helping explain performance variations and potential training disruptions across devices. These include:

- **RSSI (Received Signal Strength Indicator)**: This metric is captured from Android's `WifiManager` before each training round. RSSI reflects the strength of the Wi-Fi signal received by the device, typically measured in dBm. It is a critical indicator of network quality, as poor signal strength can lead to increased latency, packet loss, or disconnections. These factors directly affect the stability and reliability of FL on edge devices.
- **Latency**: Measured using ping requests sent from the client device to the FL server. This metric helps estimate the time delay in client-server communication, which can influence synchronization and overall training speed.
- **Download and Upload Speeds**: To assess data transfer rates, a lightweight auxiliary HTTP server was developed using Flask and deployed alongside the Flower server. By sending and receiving a dummy 10MB file from each client, it can estimate both download and upload speeds under real-world conditions. This provides a more accurate evaluation of how network throughput may influence communication efficiency during the FL communication rounds.

Chapter 5

Systematic Experimental Evaluation of Federated Learning on HAR Tasks

This chapter assess the practical feasibility and trade-offs of applying FL to the HAR problem using a real-world mobile device setup. The experiments are structured around three key dimensions, each reflecting major challenges inherent to the FL paradigm, especially when deployed on edge devices, as outlined in Section 2.4.

1. **Performance experiments** investigate how the resulted model behaves under various non-IID conditions and different client scaling configurations, given the critical impact of data heterogeneity in FL systems.

2. **Energy experiments** explore the energy demands of FL on edge devices, focusing on how model size, number of local epochs, and training rounds affect accuracy, energy footprint, and convergence speed.

3. **Network experiments** evaluate the consequences of limited network availability which is a common issue on edge devices [32], by analyzing how unstable client participation and network dropouts influence model performance and training efficiency.

5.1 Experimental Setup and Methodology

Each experiment is conducted on five Android smartphones unless stated otherwise due to specific experimental requirements. These devices differ in compute power, memory capacity, and Android OS version. Their specifications are summarized in the table 4.1.

Table 5.1 illustrates all the experimental hyperparameters which are available by the implemented infrustructure (as described in Chapter 4) grouped into three main categories: data, model, training. The data category includes parameters related to the dataset used and how it is handled like the strategy selected for test set evaluation (e.g., global vs. per-client), and then modifications applied to generate synthetic variants, as described in Section 5.2. The model category contains the parameters related to DNNs model architectures, since only this is supported at this stage. There are the number of hidden layers, hidden layer size, and learning rate. Training category include the number of participating clients , the number of federated communication rounds, and the number of local training epochs.

Table 5.1. *Summary of Configurable Parameters in Federated Learning Experiments*

Parameter	Category	Description	Comments
dataset_id	Dataset	Choice of HAR dataset	Datasets differ in number of classes, sampling rate, etc.
class_ratio	Dataset	Controls class imbalance per client	Simulates non-IID label distributions.
volume_knob	Dataset	Controls data volume per client	Tests quantity-skew scenarios.
test_strategy	Dataset	Shared vs per-client test set	default value: Shared test set
hidden_layer_sizes	Model	Neural-network architecture	default value: [x, 10, 10, y] (where x: input features, y: output classes)
learning_rate	Model	Step size for gradient updates	default value: 10^{-3}
n_clients	Training	Number of participating clients	default value: 5 (unless specified otherwise for experiment requirements)
n_epochs	Training	Local training epochs per round	default value: 3 (unless specified otherwise for experiment requirements)
n_rounds	Training	FL communication rounds	Typical range: 20–100 rounds, depending on the experiment

In order to narrow down the parameters needed to be varied in each experiment, a series of exploratory runs was conducted to establish reasonable default values. This preliminary phase was essential to avoid unnecessary reconfiguration of every parameter in each experiment, which would be impractical given the large number of hyperparameters involved. Based on empirical observations from experiments conducted across all six datasets, a lightweight neural network with two hidden layers of 10 units each, trained for 3 local epochs per round, was found to provide stable and reliable performance. The number of FL communication rounds was set to 20, as increasing it beyond this value in most of the cases examined yielded diminishing returns. However, this parameter may be adjusted as needed in specific experiments that require additional training. Additional parameters, such as batch size and learning rate, were fixed at 32 and 0.01, respectively. These were found to have a less significant impact on performance compared to model architecture, and including them in the hyperparameter search space would unnecessarily increase the experiments complexity.

Finally, all experiments in this thesis use a consistent evaluation procedure based on a single global test set, created by applying a 70%–30% train-test split before partitioning the data across clients. Although using per-client test sets can provide insight into client-specific performance and reveal heterogeneity among participants, this thesis focuses on evaluating the performance of the resulted global model. Since this global model is ultimately intended for deployment on new or unseen devices, a centralized test set offers a more representative measure of its generalization ability. Moreover, a shared evaluation test set provides uniform metric for comparing model performance across experimental conditions

To ensure statistical significance and reproducibility, each experiment was conducted three times using fixed random seeds (42, 1082, 2025).

5.2 Impact of Data Heterogeneity and Client Scaling on Model Performance

The first set of experiments will examine how HAR data characteristics influences model dynamics in the FL setup across two key dimensions. The first experiment focuses on how the inherent characteristics of HAR datasets impact model performance. As discussed in Chapter 3, HAR data can be collected using various sensors, including accelerometers, gyroscopes, and PPG using both general-purpose (e.g. smartphones, smartwatches) and specialized wearable devices (e.g., medical-grade wearables). These datasets differ in sensor type and placement (e.g. wrist, ankle, chest) as well as in other aspects such as preprocessing pipelines, feature representation (raw vs. handcrafted features), sampling rates, and activity diversity. A broad set of six publicly available HAR datasets was selected to ensure that these characteristics were well represented.

The remaining four experiments are designed to explore the effect of non-IID data, which as discussed in Chapter 2.4 represents a core challenge in FL. In addition to the inherent non-IID characteristics present in the datasets due to user and sensor specific variations, synthetically partitioned datasets are also introduced using the methods described in [7]. These synthetic partitions allow precise control over the size of skew levels (e.g., low, moderate, high) enabling targeted investigation into the effects of data imbalance. However, unlike real-world datasets, such synthetic partitions cannot fully replicate the complex and often subtle correlations found in naturally decentralized data. Using both real and synthetic non-IID distributions, a series of experiments is conducted to evaluate different types of statistical heterogeneity. The second experiment explores label skew, where certain activity classes are missing or underrepresented in specific clients. This is common in HAR settings, where frequent activities like walking or sitting tend to dominate, while rare actions such as falling or cycling appear less often. Next quantity skew is investigated where clients possess unequal volumes of training data. This scenario reflects natural disparities among users, where some may generate far more sensor data than others due to differences in usage patterns or device availability. Next the mix skew is investigated by combining both label and quantity imbalances in order to create a more complex and realistic non-IID scenario making FL model's ability to generalize and converge effectively even more challenging. Finally, client scaling is explored by gradually increasing the number of participating clients while maintaining their true non-IID data structure. Monitoring model's ability to scale provides the most accurate assessment of its expected performance under real-world conditions.

5.2.1 Model Evaluation Across HAR Datasets

To evaluate the model's performance across the six different datasets, the data is shuffled and randomly split among five clients. This approach removes the inherent non-IIDness of the original datasets. This setup is intended to isolate the impact of dataset-specific properties without introducing additional skew or imbalance. All the other experimental parameters are set to their default values, as defined in Table 5.1.

As shown in Figure 5.1, accuracy converges rapidly across all datasets. Most models reach over 95% of their final accuracy by round 6, regardless of dataset characteristics such as preprocessing pipeline, number of activity labels or number and type of sensor modalities. Among all datasets, PAMAP2 achieved the highest final accuracy (98%), while MHEALTH, PhysioNet Accelerometry, and UCI Smartphone HAR performed slightly worst, achieving accuracies in the range of 87%–90%. MotionSense and HAR-Sense had the lowest performance, with final accuracies between 76% and 79%.

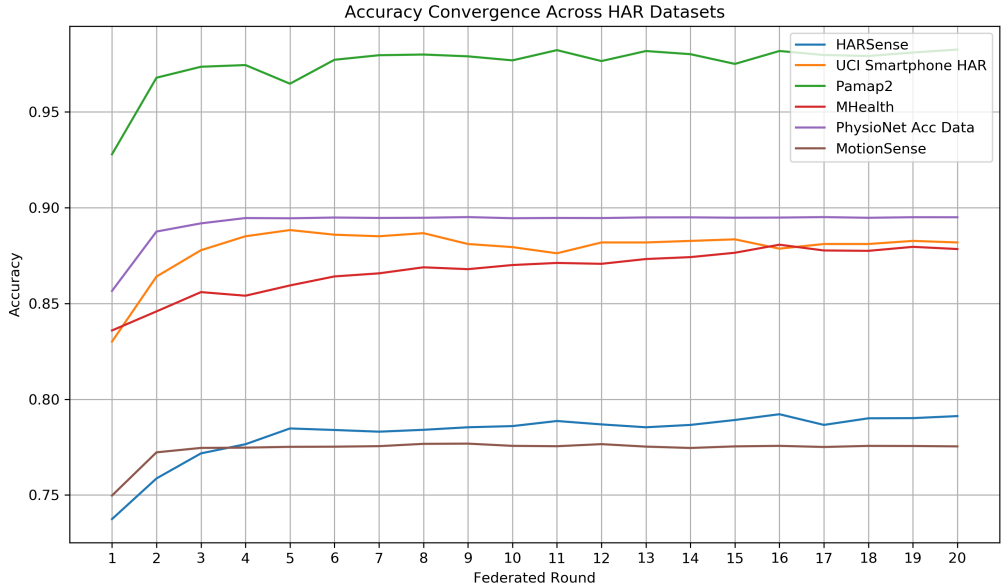
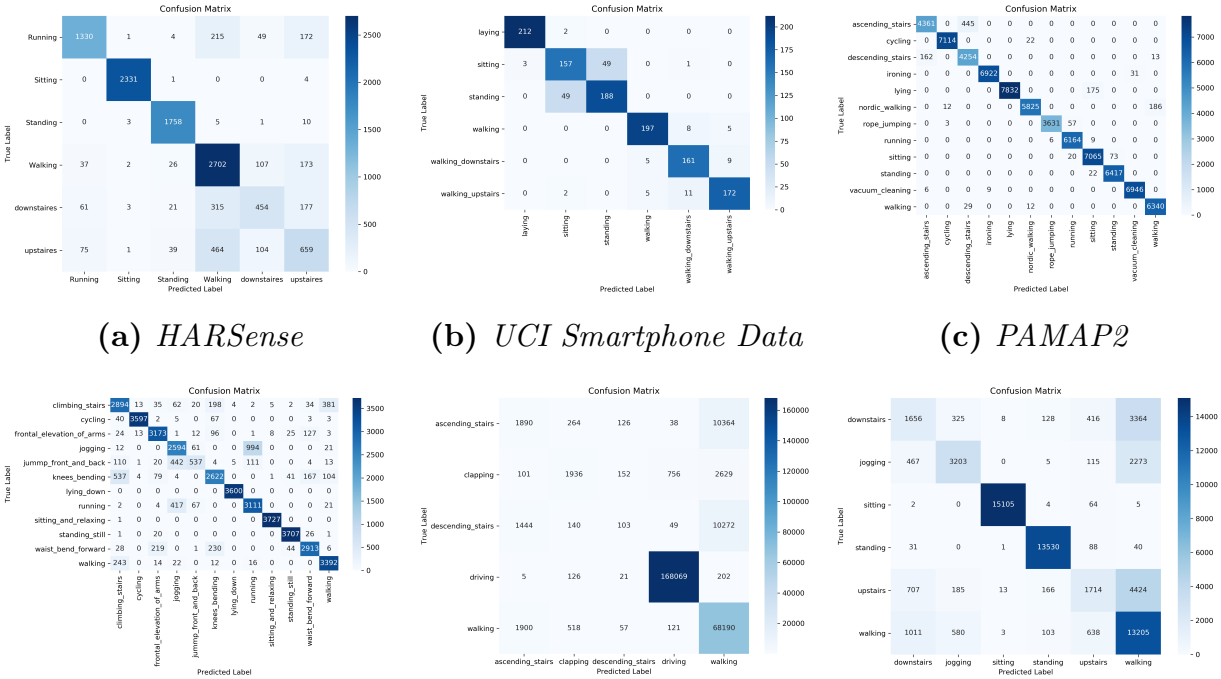


Figure 5.1. *Model Accuracy Convergence on Diverse HAR Datasets under IID Conditions*

As show in confusion matrices presented in 5.2, MotionSense and PhysioNet datasets, despite achieving relatively high overall accuracy, exhibit a clear bias toward dominant classes such as walking and sitting. Models trained on these datasets tend to misclassify less frequent activities, resulting in substantially lower macro-averaged F1-scores. In contrast, datasets with more balanced class distributions, such as PAMAP2 and MHEALTH show greater alignment between accuracy and F1 metrics, indicating that the models perform more consistently across all activity classes.

Although this thesis does not intend to assess the impact of individual dataset characteristics such as preprocessing methods or feature extraction techniques, some interesting observations emerged from this experiment. First, datasets generated from raw smartphone sensors (e.g., accelerometers, gyroscopes) resulted in poorer model performance compared to those created using richer multi-sensor configurations, even when the latter included a greater number of activity classes. For instance, MHEALTH (raw, multi-sensor) and PAMAP2 (preprocessed, multi-sensor) both consistently outperformed simpler, smartphone-only datasets. Moreover, the highest performance was achieved on PAMAP2, which combines multiple synchronized sensors, extensive preprocessing, and a well-balanced class distribution.

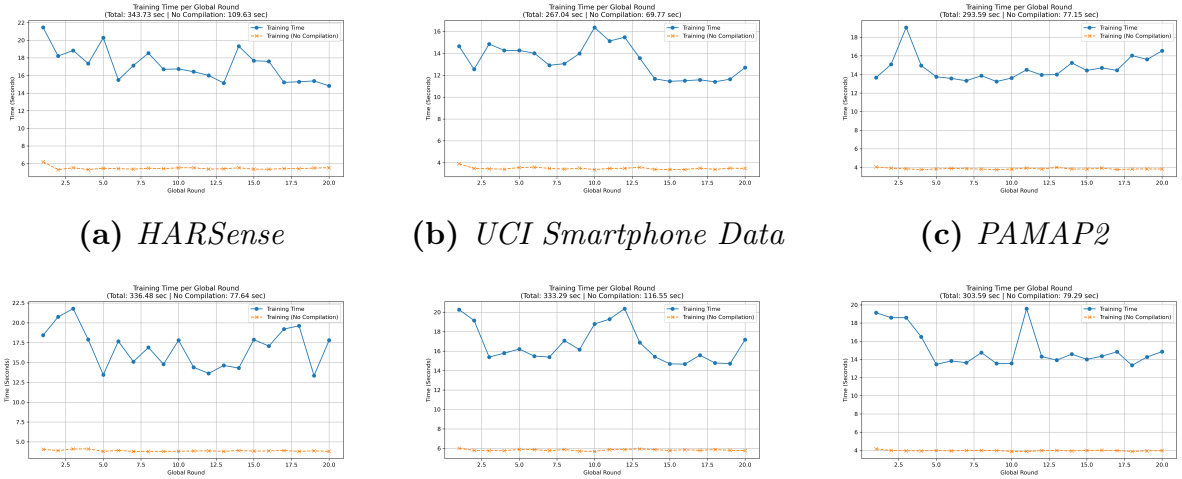
Regarding training time (see Figure 5.3), a key observation is that model compilation dominates the overall runtime especially in smaller models. Across all datasets, the



(a) *HARSense* **(b)** *UCI Smartphone Data* **(c)** *PAMAP2*

(d) *MHealth* **(e)** *PhysioNet Accel* **(f)** *MotionSense*

Figure 5.2. *Dataset Variation Experiment Confusion Matrices*



(a) *HARSense* **(b)** *UCI Smartphone Data* **(c)** *PAMAP2*

(d) *MHealth* **(e)** *PhysioNet Accel* **(f)** *MotionSense*

Figure 5.3. *Dataset Variation Experiments - Training time plots*

compilation step consumed up to 75% of total training time. Furthermore, while training time without compilation remained relatively stable, training time including compilation fluctuated significantly. This variation occurs since in contrast to the server, smartphones were dedicated exclusively for the FL task so there where no other processes consuming system resources. These figures clearly illustrate how server capabilities can impact convergence speed, even though as discussed in Chapter 4.1.1, the compilation overhead only exists because of TFLite's inability to programmatically update model weights.

5.2.2 Effect of Label Imbalance Across Clients

All skew experiments (label, quantity, and mixed) are conducted using the HARSense dataset since as shown in Experiment 5.2.1, HARSense was the overall worst-performing dataset while maintaining relatively equal accuracy across all labels. In contrast, MotionSense showed even lower overall performance but exhibited significant imbalance in label-wise accuracy, making it less suitable for controlled skew analysis. The test examines three levels of label skewness low, moderate, and extreme in which every client will be assigned with a dataset that completely misses 1,3 or 5 labels out of 6 respectively. Labels are removed in a round-robin fashion to ensure balanced distribution of missing classes across clients. For example, in the low-skew setting, if Client 1 has access to labels 2 through 6, then Client 2 will have labels 1 and 3 through 6, and so on.

As shown in Figure 5.4, under the low level of label skew, the model still converges relatively well, reaching approximately 72% accuracy, slightly below the baseline for the non-skewed configuration (79%). The corresponding confusion matrix shown in Figure 5.5 shows relatively balanced performance across most labels, though increased misclassification is observed for dynamic activities such as **walking**, **downstairs**, and **upstairs**. For instance, of the 7490 **walking** samples, 3480 are misclassified as **upstairs** while only 3640 are correctly identified. Notably, **upstairs** does not degrade significantly, likely because this label is still present in the training sets of all five clients as we can see from the distribution of test samples provided on table 5.2.

At the moderate skew level, where each client sees only three out of six labels, accuracy plateaus at around 40% and confusion across labels becomes markedly worse. As expected, **downstairs** deteriorates further, and **walking** continues to exhibit significant misclassification. Interestingly, upstairs now also becomes more error-prone, aligning with the fact that it is no longer present in every client's training data.

At the extreme level of label skew, the model fails to converge and stabilizes at approximately 22% accuracy. The confusion matrix reveals widespread misclassifications, but also highlights an interesting pattern in how predictions are distributed. Specifically, the model disproportionately favors certain labels such as **standing**, **downstairs**, and **upstairs** despite the fact that these labels have no clear advantage in sample size within either the training or test sets. One possible explanation for this behavior is that labels that frequently appear together on the same client such as **downstairs** and **upstairs** can reinforce each other during local training, leading to their overepresentation in the global model. Conversely, labels that are isolated to a single client such as **running** or **walk-**

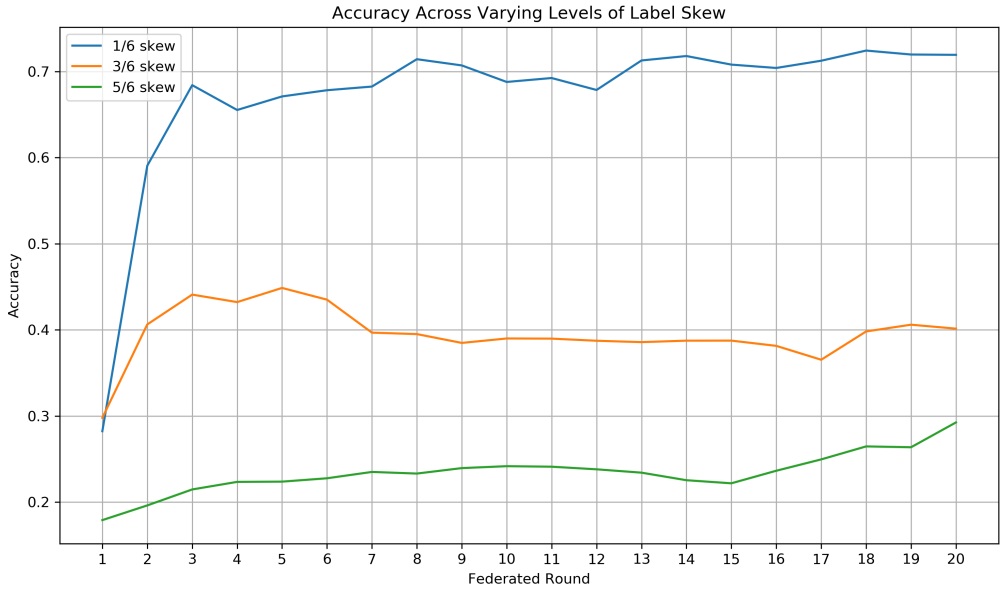
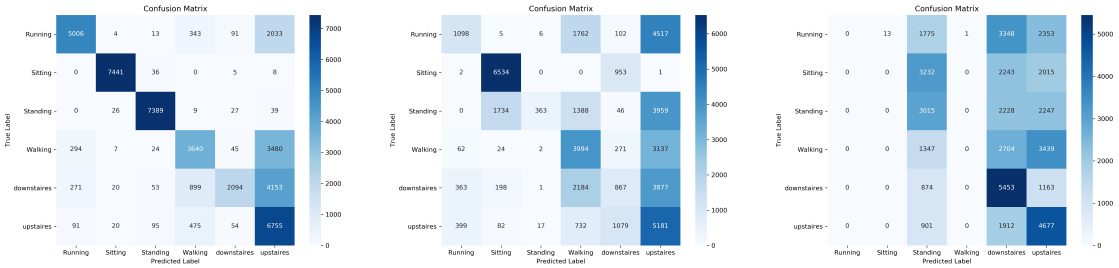


Figure 5.4. *Effect of Progressive Label Skew on Model Accuracy*



(a) *Level 1/6* **(b)** *Level 3/6* **(c)** *Level 5/6*

Figure 5.5. *Confusion Matrices Under Increasing Label Skew Intensity*

ing and lack cross-client occupation tend to be underrepresented or even ignored during prediction. However, this reasoning does not fully account for the strong prediction bias toward the standing class, which also appeared on only one client. This highlights the fact that there are numerous local training dynamics that also influence the prediction like label co-occurrence and client specific data distributions. While such extreme label distributions may be rare in practical deployments, this experiment raises an important and underexplored question. To what extent can FL systems effectively propagate label knowledge across clients in the presence of missing label distributions? Based on our observations, improving this type of cross-client knowledge transfer could significantly enhance FL performance in non-IID settings and it is an interesting direction for future research.

5.2.3 Effect of Data volume on Model Performance

Quantity skew experiments evaluate low, moderate, and extreme levels of data imbalance by assigning each client 60%, 30%, or 10% of the original training data respectively.

From the accuracy convergence plot 5.6, it is clearly observed that the amount of local data significantly influences both convergence speed and final model performance.

Table 5.2. *Per-Client Label Distribution Across Label Skew Configurations*

Level	Client	Running	Sitting	Standing	Walking	Downstairs	Upstairs
1/6	1	0	3 593	3 593	3 593	3 593	3 593
	2	3 571	0	3 571	3 571	3 571	3 571
	3	3 485	3 485	0	3 485	3 485	3 485
	4	3 516	3 516	3 516	0	3 516	3 516
	5	3 611	3 611	3 611	3 611	0	3 611
	Total	14 183	14 205	14 291	14 260	14 165	17 776
3/6	1	0	3 593	0	3 593	0	3 593
	2	3 571	0	3 571	0	3 571	0
	3	0	3 485	0	3 485	0	3 485
	4	3 516	0	3 516	0	3 516	0
	5	0	3 611	0	3 611	0	3 611
	Total	7 087	10 689	7 087	10 689	7 087	10 689
5/6	1	3 593	0	0	0	0	0
	2	0	3 571	0	0	0	0
	3	0	0	3 485	0	0	0
	4	0	0	0	3 516	0	0
	5	0	0	0	0	3 611	3 611
	Total	3 593	3 571	3 485	3 516	3 611	3 611

In the low quantity skew setting the model reaches nearly 79% accuracy by round 20 which is almost identical to the baseline case where 100% data is present in each client. With moderate skew (30%), convergence slows slightly and plateaus around 78%, while in the extreme case (10%), the model still reaches 77% but with even slower convergence. This result highlights that even with limited data, the model can maintain reasonable performance likely due to the preserved label balance and the relatively simple architecture that does not require large amount of data to generalize well. While the confusion matrices (see Figure 5.7) show a gradual decline in classification performance as the skew increases, no consistent misclassification pattern clearly emerges across all levels. Table 5.3 shows the distribution of train set label counts for each client and quantity skew level. An interesting extension to this experiment would be to explore uneven data distribution among clients in order to explore how clients with less data may benefit from those with more, and vice versa.

5.2.4 Combined Label and Quantity Skew

To further examine the challenges posed by non-IID data, mix skew experiments were conducted by combining both label and quantity skew. Specifically, two configurations were tested: a moderate level, in which each client was assigned only 30% of the original training data and 3 out of 6 labels were omitted; and an extreme level, where clients received just 10% of the training data and 5 out of 6 labels were missing. In the moderate setting, convergence was still achieved, reaching 49% accuracy by round 20. In contrast, in the extreme setting, convergence was nearly halted, with accuracy stagnating around

Table 5.3. *Per-Client Training Sample Counts in Quantity Skew Experiments*

Level	Client	Running	Sitting	Standing	Walking	Downstairs	Upstairs
60%	1	2 156	2 156	2 156	2 156	2 156	2 156
	2	2 143	2 143	2 143	2 143	2 143	2 143
	3	2 091	2 091	2 091	2 091	2 091	2 091
	4	2 110	2 110	2 110	2 110	2 110	2 110
	5	2 167	2 167	2 167	2 167	2 167	2 167
	Total	10 667	10 667	10 667	10 667	10 667	10 667
30%	1	1 078	1 078	1 078	1 078	1 078	1 078
	2	1 071	1 071	1 071	1 071	1 071	1 071
	3	1 046	1 046	1 046	1 046	1 046	1 046
	4	1 055	1 055	1 055	1 055	1 055	1 055
	5	1 083	1 083	1 083	1 083	1 083	1 083
	Total	5 333	5 333	5 333	5 333	5 333	5 333
10%	1	359	359	359	359	359	359
	2	357	357	357	357	357	357
	3	348	348	348	348	348	348
	4	352	352	352	352	352	352
	5	361	361	361	361	361	361
	Total	1 777	1 777	1 777	1 777	1 777	1 777

Table 5.4. *Per-Client Training Sample Counts in Mixed Skew Experiments*

Level	Client	Running	Sitting	Standing	Walking	Downstairs	Upstairs
Moderate	1	1 078	0	1 078	0	1 078	1 078
	2	0	1 071	0	1 071	0	1 071
	3	1 046	0	1 046	0	1 046	0
	4	0	1 055	0	1 055	0	1 055
	5	1 083	0	1 083	0	1 083	0
	Total	3 207	2 126	3 207	2 126	3 207	3 204
Extreme	1	359	0	0	0	0	0
	2	0	357	0	0	0	0
	3	0	0	348	0	0	0
	4	0	0	0	352	0	0
	5	0	0	0	0	361	361
	Total	359	357	348	352	361	361

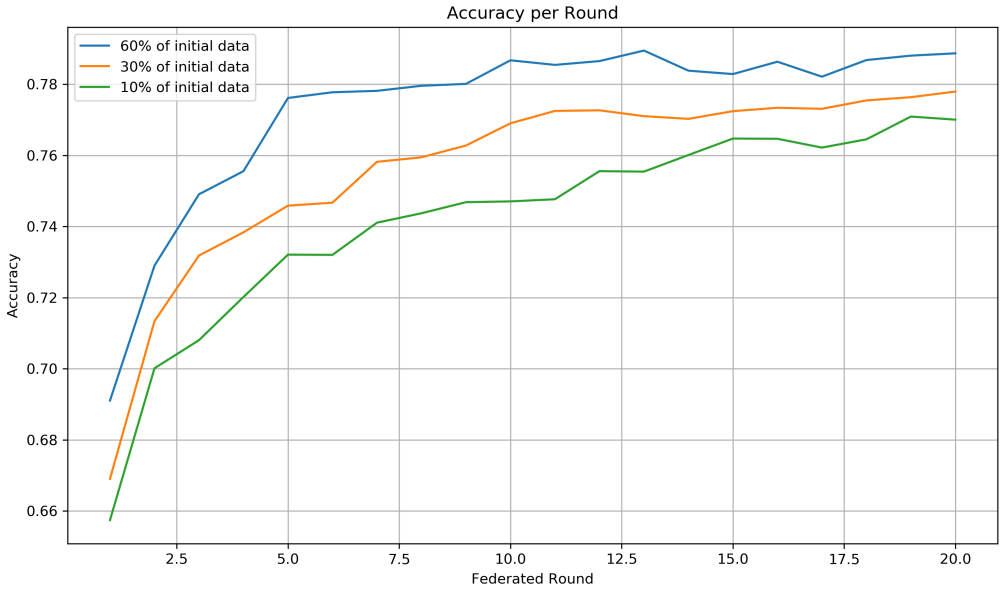
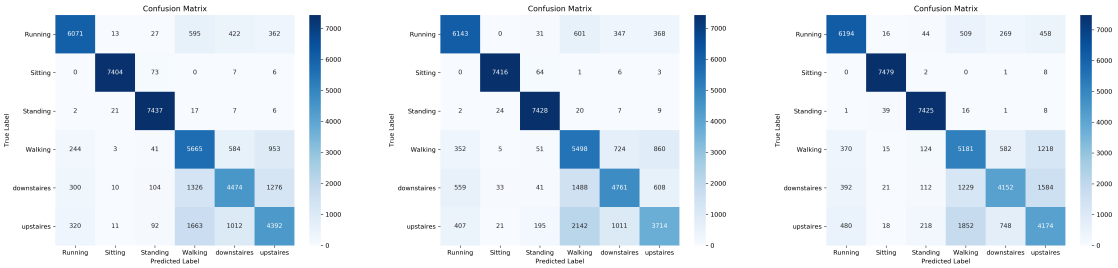


Figure 5.6. *Impact of Data Volume Imbalance on Model Accuracy*



(a) *Level 60%* **(b)** *Level 30%* **(c)** *Level 10%*

Figure 5.7. *Confusion Matrices Under Different Levels of Quantity Skew*

22%. The confusion matrices are shown in Figure 5.9, and the corresponding training-set label distributions are detailed in Table 5.4.

5.2.5 Model Behavior Under Varying Client Participation

In this experiment, the number of participating clients is progressively increased from 2 to 5 in order to evaluate how model performance scales with broader client participation. It will be conducted using the MotionSense dataset since in the dataset variation experiments it had the weakest overall performance while it was collected from a relatively large pool of 24 subjects. This subject diversity makes it ideal for examining how incremental client participation affects model generalization. In comparison, the similarly underperforming HARSense dataset includes data from only 12 subjects. To preserve the non-IID nature of the dataset, each of the five available clients train data was assigned only the portion of data associated with a single subject, as indicated by the dataset's subject_id attribute. As a result, when training with five clients, the global model was constructed using data from only 5 out of the 24 total subjects (approximately 20% of the available data). The test set, however was sampled from the entire dataset to preserve the original class distribution.

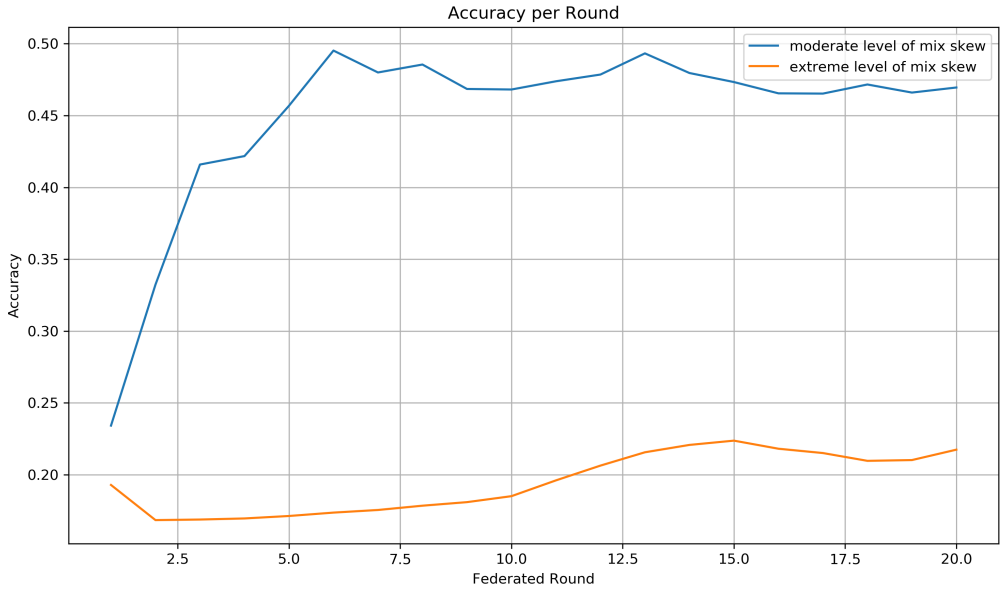
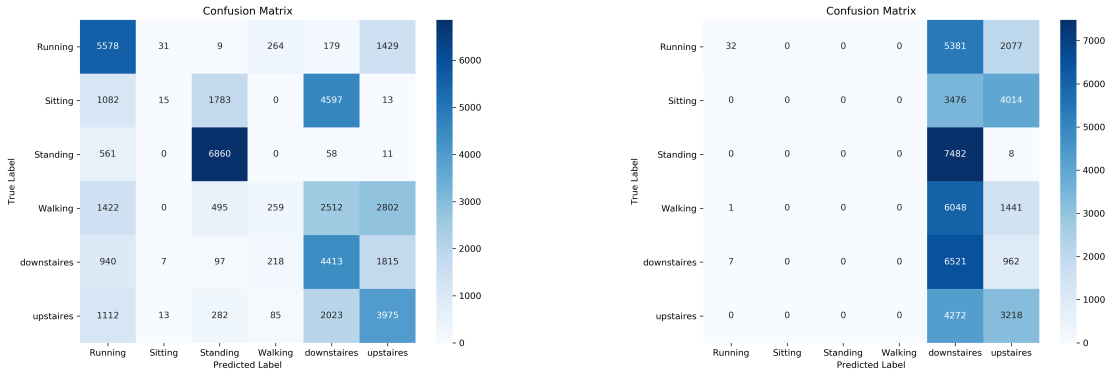


Figure 5.8. *Effect of Combined Label and Quantity Skew on Model Performance*



(a) *Moderate Level 3/6 60%* **(b)** *Extreme Level 1/6 30%*

Figure 5.9. *Confusion Matrices Under Different Levels of Combined Label And Quantity Skew*

To establish a reference for the upper bound of performance under these constraints, an additional experimentation run was conducted in which data from all 24 subjects was evenly distributed across the five clients. In order to retain the subject-level non-IID structure and have a more realistic deployment scenarios the subjects were grouped in contiguous blocks rather than being randomly shuffled. For example, subjects 1 to 5 were assigned to client 1, subjects 6 to 10 to client 2, and so on.

To provide a baseline under ideal conditions, the same client scaling analysis was also repeated using IID data, allowing a direct comparison of how scaling affects FL performance under both IID and non-IID settings.

As shown in Figure 5.10, in the IID setting, model performance improved slightly as the number of clients increased. Accuracy range from approximately 0.77 with 2 clients to just above 0.78 with 5 clients. Convergence was stable across all configurations, with very low variance between rounds. This behavior is expected since under IID conditions,

Figure 5.10. *Impact of Client Count on Model Accuracy (IID Setting)*

while increasing the number of clients adds more data per round, the underlying statistical distribution remains virtually unchanged. As a result, the global model is exposed to equivalent representative samples, leading to marginal improvements in generalization.

In contrast, Figure 5.11 reveals a more complex pattern under the non-IID setting. Here, both accuracy and convergence are highly sensitive to the number of participating clients. With only 2 clients, the model struggles to converge, fluctuating heavily and achieving only 27–30% accuracy. As the number of clients increases to 3 and 4, accuracy improves to approximately 35–40%, though the high variability between rounds persists. With 5 clients, accuracy rises further to above 40%, but the instability persists.

One likely explanation for this fluctuation is inter-subject variability as described on Chapter 3.2.1. Even within a lab-controlled HAR dataset, subjects differ in movement patterns (e.g., gait, walking speed), and phone placement may vary slightly even if scripted. In real-world settings, such variability would likely be even more pronounced. These observations demonostrate the potential value of model personalization, where the global model is fine-tuned to adapt to a specific user's data after FL training is completed. This step could mitigate some of the generalization issues caused by diverse local data.

Interestingly, the upper-bound configuration where data from all 24 subjects was evenly distributed across 5 clients, did not lead to a dramatic improvement in accuracy as originally expected. While it did provide a more stable training curve it reaches an accuracy of approximately 35%.This suggests that even with access to a broader subject pool, the model struggles to learn a truly global representation. It may indicate that 24 subjects are still insufficient to fully capture the diversity of HAR especially in non-IID settings. This experiment highlights the significant impact of data heterogeneity on model generalization, as discussed in Chapter 2.4.3. In this case, the non-IID nature of the data stems primarily from subject-level variability differences in movement patterns, behavior, or sensor placement while the datasets across clients are otherwise balanced in size and

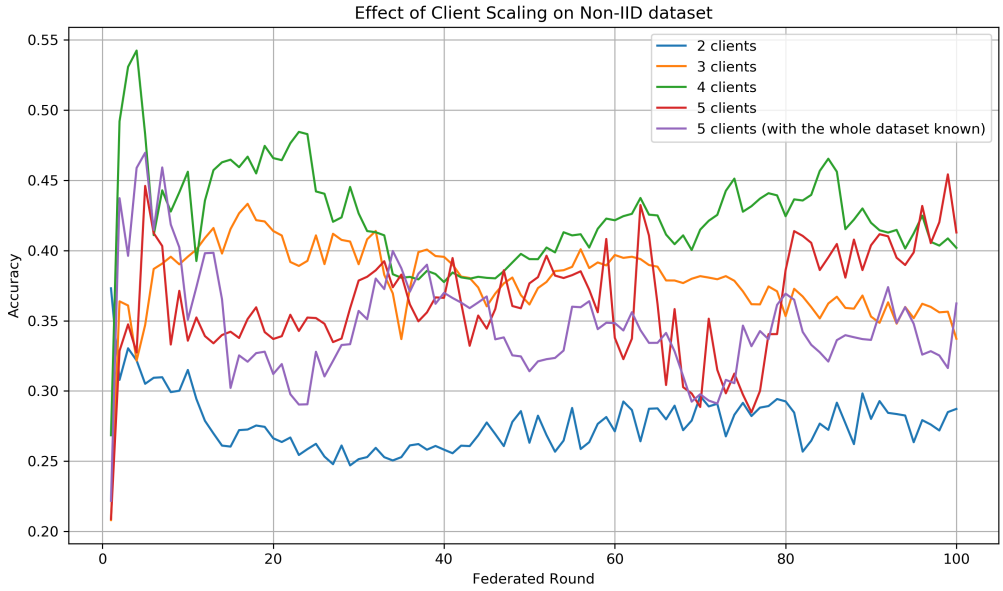


Figure 5.11. *Impact of Client Count on Model Accuracy (non-IID Setting)*

class distribution. In real-world scenarios, where both label imbalance and data quantity skew are more pronounced, this issue is likely to be even more severe. Given how important this challenge is to the practical deployment of FL for HAR, further investigation is essential. Specifically, it would be valuable to explore alternative aggregation algorithms designed to handle non-IID data more effectively such as FedProx, SCAFFOLD and to assess whether they offer better robustness and convergence in the presence of diverse client distributions.

5.3 Energy Efficiency Trade-offs in Federated Learning

A core concern when deploying FL on mobile or edge devices is energy consumption. These devices operate with inherently limited energy resources, and training over many communication rounds can result in significant battery drain. Therefore, energy efficiency is a critical consideration in the design and deployment of FL systems.

The total energy consumed by an edge device during training process can be expressed as

$$E = E_{\text{net}} + E_{\text{c}} + E_{\text{sys}}, \tag{5.1}$$

where E_{net} is the energy used for network communication, E_{c} is the energy consumed during computation, and E_{sys} accounts for system-level energy costs (e.g. OS overhead and background processes)[68].

While E_{net} and E_{c} are directly influenced by design parameters such as the number of local epochs, communication rounds and model size, E_{sys} is largely unrelated to these variables and is instead determined by hardware-specific factors. As a result, we can perform grid search over the parameters that influence E_{net} and E_{c} to identify the optimal balance between energy consumption and model performance. This tuning process is

guided by two key observations. First in wireless environments, communication energy cost (E_{net}) is not negligible and can become a significant factor especially in settings where the local model is lightweight and requires minimal computation, as is often the case in HAR tasks. So by increasing the number of local training epochs per round, the total number of communication rounds required for convergence can be reduced. This, in turn reduces communication overhead and overall energy consumption. This insight leads to the local vs. global computation trade-off, where more on-device training may reduce costly network communication. Second, larger DNN models, with more layers and parameters demand more energy for local training, but they tend to achieve better performance. This introduces the energy vs. performance trade-off, where the challenge lies in balancing computational cost with model accuracy. The objective of this section is to experimentally evaluate both of these trade-offs.

The first experiment investigates the local computation vs global communication trade-off by keeping the total number of gradient descent steps constant, while varying the ratio between local epochs and global communication rounds. For example, configurations such as 4 epochs $\times$ 25 rounds and 5 epochs $\times$ 20 rounds result in the same number of total updates, but distribute the workload differently between local computation and network communication. This allows us to assess whether increasing local training per round while reducing the number of communication rounds can lead to more energy-efficient training without compromising performance.

The second experiment examines how model size affects energy consumption, training time, and accuracy, aiming to identify the smallest model that delivers acceptable performance while consuming the least energy.

As described in Chapter 4.5.3, the energy consumption during the FL process for each round and device is calculated using equation 4.1, while the total energy required to produce the final global model is computed using equation 4.2. However, among the five client devices used in the experiment (see Table 4.1), Samsung SM-A510F did not expose charge and voltage metrics, and Xiaomi M2006C3MNG consistently produced unreliable or inconsistent readings. As a result, the average energy consumption was calculated using data from the remaining three devices. Additionally, to estimate the total energy cost across all five clients, the energy usage of the two excluded devices was approximated using the mean consumption of the three valid devices.

Both of these experiments use the same low-level label skewed dataset configuration described in Section 5.2.2 as it more accurately reflects the non-IID data distributions encountered in real-world federated learning deployments.

5.3.1 Balancing Local Computation and Communication for Energy Efficiency

In this experiment, the total number of stochastic gradient descent steps is fixed at 100, while the configuration of local training epochs and federated communication rounds is varied. The four training setups evaluated are: 1×100, 2×50, 4×25, and 10×10.

As shown in the top part of Figure 5.12, energy consumption per device decreases

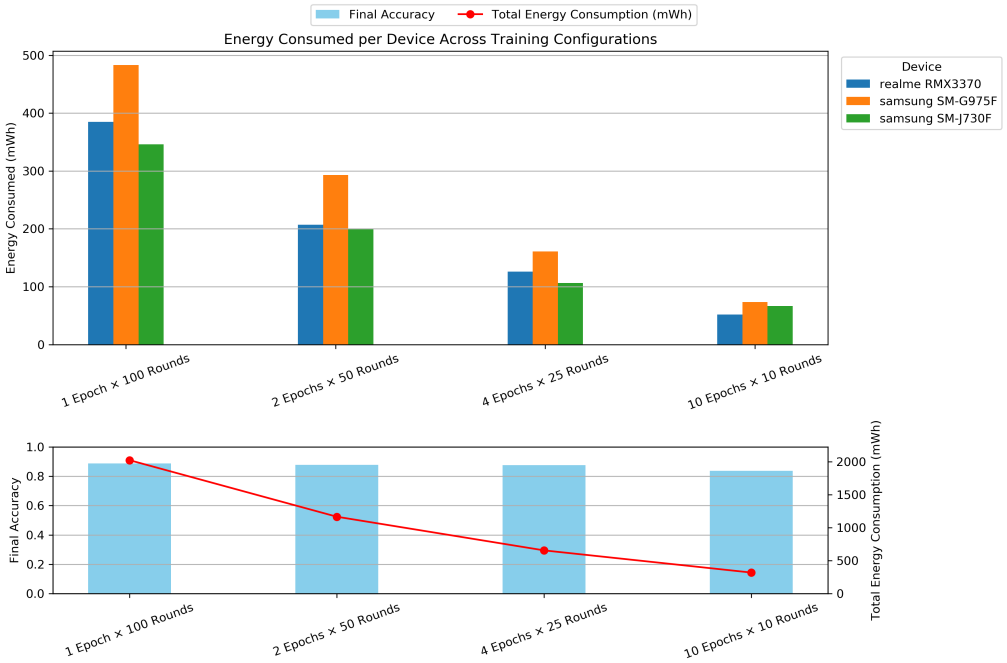


Figure 5.12. *Energy-Accuracy Trade-off: Local Epochs vs Communication Rounds.* ***Top****: Total energy consumption per device (in mWh) across varying local epoch and communication round configurations.* ***Bottom****: Final model accuracy (blue bars, left y-axis) vs. total energy consumption (red line, right y-axis).*

consistently as the number of communication rounds is reduced. This pattern is observed across all three devices with valid energy readings (Realme RMX3370, Samsung SM-G975F, Samsung SM-J730F) and aligns with expectations that fewer global updates which mean fewer network transmissions are typically more energy-intensive than local computation. Another interesting observation is that Realme RMX3370 which the newest most cabable device in the group, consumes less energy than the older Samsung models across all configurations. This highlights how hardware efficiency plays a substantial role in total energy usage in FL deployments.

The bottom part of Figure 5.12 shows the final model accuracy alongside the total combined energy consumption for each configuration. Total energy across all five clients (including estimates for the two with missing data) is as follows:

- 1×100: 2023.99 mWh
- 2×50: 1166.49 mWh
- 4x25: 655.75 mWh
- 10×10: 319.37 mWh

While the highest accuracy is achieved with 1×100 setting, the accuracy difference between this and the more energy-efficient 10×10 configuration is marginal. Thus, the energy cost of frequent communication is not justified by the relatively small performance gains. One possible explanation for why more communication rounds may lead to improved

accuracy is that more frequent model aggregation helps the global model better integrate the diverse data patterns learned by individual clients. As demonstrated in the non-IID experiments in Sections 5.2.2, 5.2.3, and 5.2.4, this effect is particularly important when client data distributions are highly non-IID.

The most important takeaway though, is the dominant role of communication rounds in energy consumption. Between the most communication-intensive configuration (1×100) and the most efficient one (10×10), total energy use drops by over 84%, while accuracy remains nearly unchanged. This demonstrates that energy savings in FL can be significantly optimized by adjusting the balance between local computation and global synchronization.

5.3.2 Impact of Model Complexity on Energy Consumption

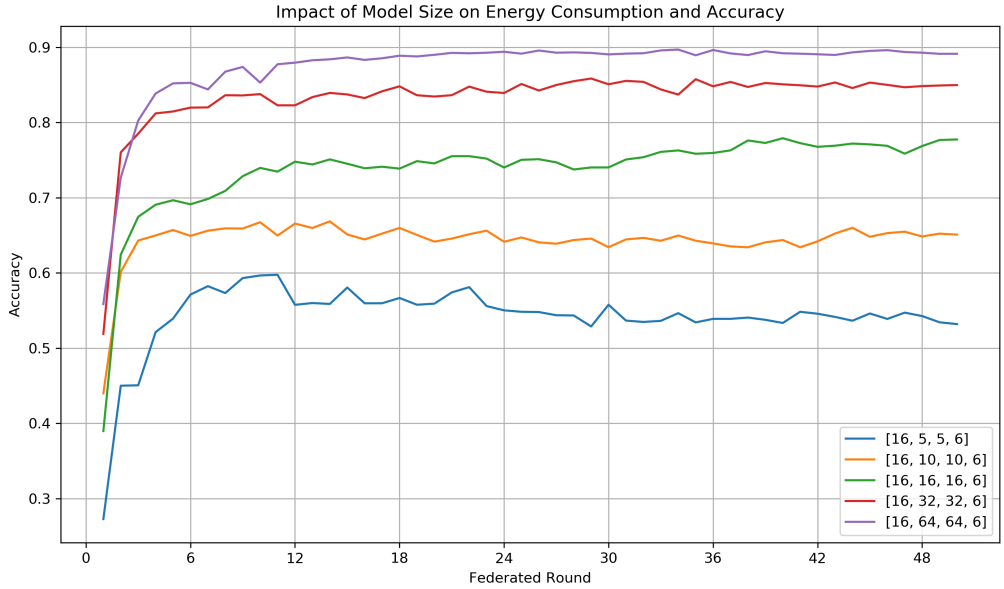


Figure 5.13. *Impact of Model size on Accuracy*

In this experiment, the impact of increasing model complexity on both energy consumption and accuracy is evaluated. Five fully connected neural network configurations are tested, with the same input and output dimensions but varying hidden layer sizes:

- model1: [16, 5, 5, 6]
- model2: [16, 10, 10, 6]
- model3: [16, 16, 16, 6]
- model4: [16, 32, 32, 6]
- model5: [16, 64, 64, 6].

Figure 5.13 shows how model accuracy evolves over federated rounds. As expected, larger models consistently achieve better performance. The smallest model converges to just over 55% accuracy, while the largest model5 reaches approximately 90%. Accuracy

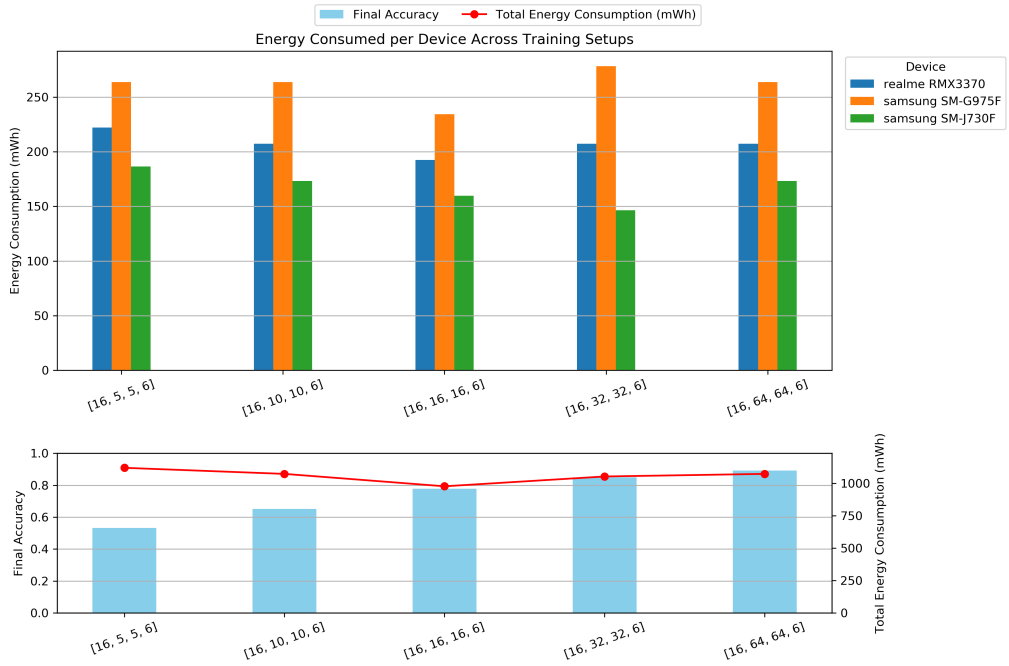


Figure 5.14. *Energy Cost vs Accuracy for Varying Model Architectures.* ***Top****: Total energy consumption (in mWh) per device for different neural network configurations.* ***Bottom****: Final model accuracy (blue bars, left y-axis) versus total energy consumption (red line, right y-axis)*

steadily improves with model size, however after the model4 gains start to wear out which indicates diminishing returns beyond this point.

The top part of Figure 5.14 illustrates the energy consumption per device across all tested model architectures, while the bottom part shows the final accuracy of the global model along with the total energy consumed to reach that accuracy. One counter-intuitive observation is that, although model accuracy improves significantly, from approximately 55% for the smallest model to 90% for the largest, the total energy consumption remains relatively stable across all configurations:

- model1: 1120.14 mWh
- model2: 1073.27 mWh
- model3: 977.59 mWh
- model4: 1053.28 mWh
- model5: 1073.27 mWh

There are two possible explanations for this. First, the increase in model size may be relatively small compared to the computational capabilities of the devices, meaning that the additional workload does not substantially affect power consumption. Second, after monitoring CPU load using Android Studio, it was observed that CPU usage clipped at around 13% for each device. This is due to restrictions imposed by the Android OS, which limits the CPU resources allocated to any single app. As a result, larger models may not

increase CPU usage but instead extend the computation time. Although this leads to higher overall energy consumption, the nature of this trade-off remains largely unexplored.

Nevertheless, these findings highlight the need for further investigation into the relationship between model architecture and energy consumption. While as seen from the accuracy improvements, model size clearly influences performance. On this specific examle its impact on energy efficiency remains less straightforward. Given the results from the previous experiment, where training configuration significantly affected energy usage, future work should explore how architectural choices can affect both the accuracy and the energy efficiency.

5.4 Robustness of FL Under Network Variability

Network conditions directly influence the convergence time of FL, as each FL round consists of both computation and communication phases. In our initial tests, we evaluated the impact of varying signal quality which is measured using RSSI values, where approximately –35dBm indicates strong signal strength and –78dBm indicates poor signal strength. However, no significant fluctuation in convergence time was observed, regardless of network quality. This outcome was expected, as the experimental setup involved relatively small models with few trainable parameters since over the course of 75 rounds, approximately 3.72 MB of data was received and 0.52 MB was transmitted by each client. Given these small communication payloads, variations in network bandwidth or signal strength were found to have a negligible impact on overall training time. As a result, attention was directed toward two more critical aspects of network conditions in FL systems. First, due to unstable connections, clients may occasionally fail to participate in a given round. This leads to the question: How robust is the global model when client availability is probabilistic rather than guaranteed? Second, the scenario of permanent client dropouts was considered, in which clients may leave the training process entirely. The key question here is: To what extent does model performance degrade as the pool of participating clients shrinks over time?

Both of these experiments use the same low-level label skewed dataset configuration described in Section 5.2.2 as it more accurately reflects the non-IID data distributions encountered in real-world federated learning deployments.

5.4.1 Impact of Intermittent Client Participation

To emulate the conditions of probabilistic client availability, each client is configured to participate in any given round with a probability$a \in \{25\%, 50\% 75\% 100\%\}$. This is implemented by modifying the server logic to probabilistically include each client's updates in the aggregation step based on the specified value of a, while still maintaining five clients as available participants throughout the experiment.

Figure 5.15 presents the model accuracy over time for each availability level as well as a corresponding chart showing the number of clients participating in each round under the different values of a. As illustrated, reduced client availability leads to slower convergence

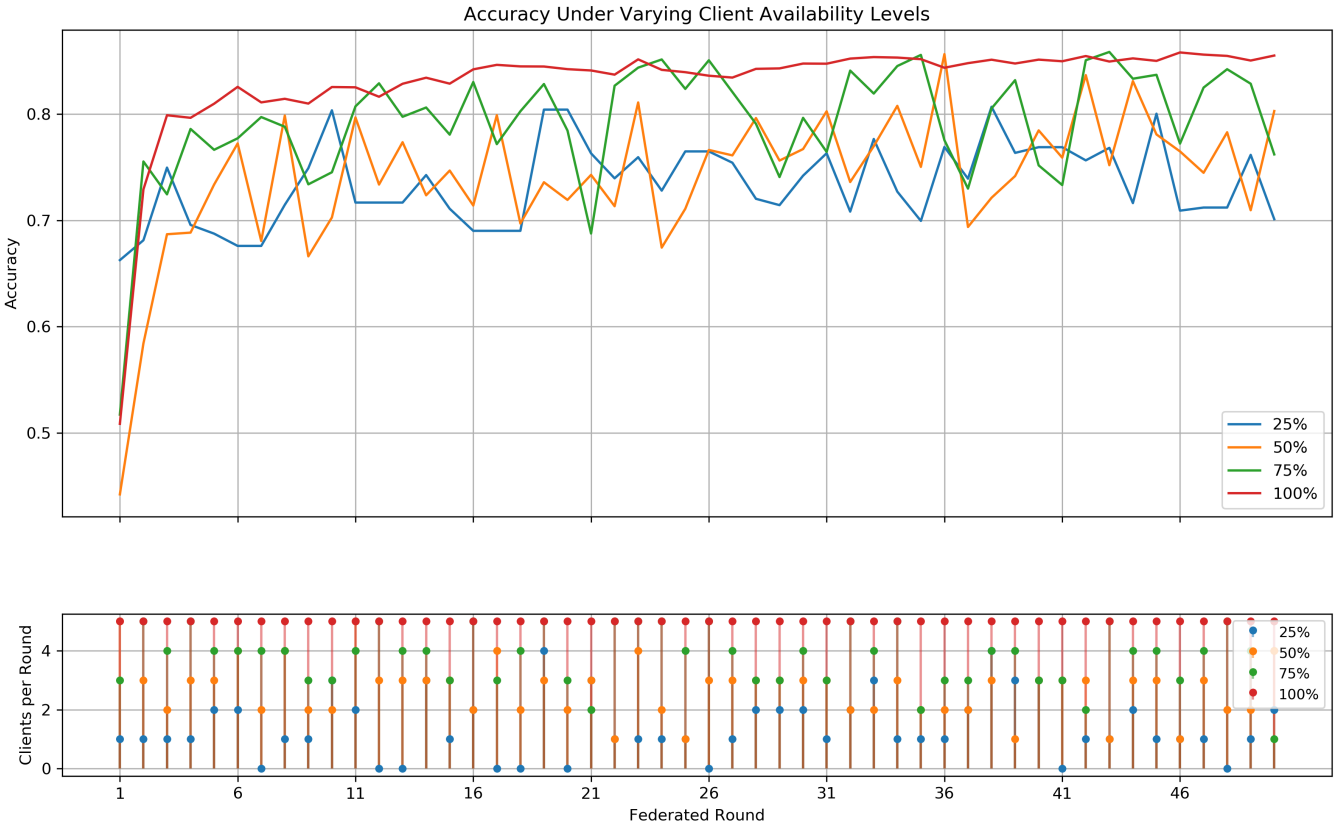


Figure 5.15. *Model Convergence Stability under Probabilistic Client Participation*

and increased instability in model accuracy. At 75% participation, fluctuations in accuracy become immediately noticeable. While these fluctuations do grow more pronounced as participation rates decrease further, the increase is not as dramatic as one might expect. Our intuition is that, because FedAvg is not inherently robust to non-IID data, the algorithm may already be operating near its limits even at 75% participation. As a result, further reductions in participation have a less noticeable impact, since the performance degradation is already substantial. It is possible that with more robust aggregation methods, such as FedProx or SCAFFOLD, we would observe a more pronounced drop in accuracy and a sharper increase in instability as client availability decreases. Understanding how different aggregation strategies respond to varying client availability in non-IID settings is an important direction for future research, as non-IID data remains one of the core challenges and open issue in federated learning bibliography.

5.4.2 Model Resilience to Permanent Client Dropout

These experiments are conducted over a total of 40 federated rounds. The permanent client dropouts are simulated by removing 1, 2, 3, and 4 clients respectively at round 40 when the model has undergone a substantial amount of training. These scenarios are compared against a benchmark case in which all clients remain active throughout the training process. To implement this setup, the server's maximum number of clients

required to initiate training is set to 5, and the minimum number of clients required to continue training is set to 1. The specified clients are then manually disconnected after round 20.

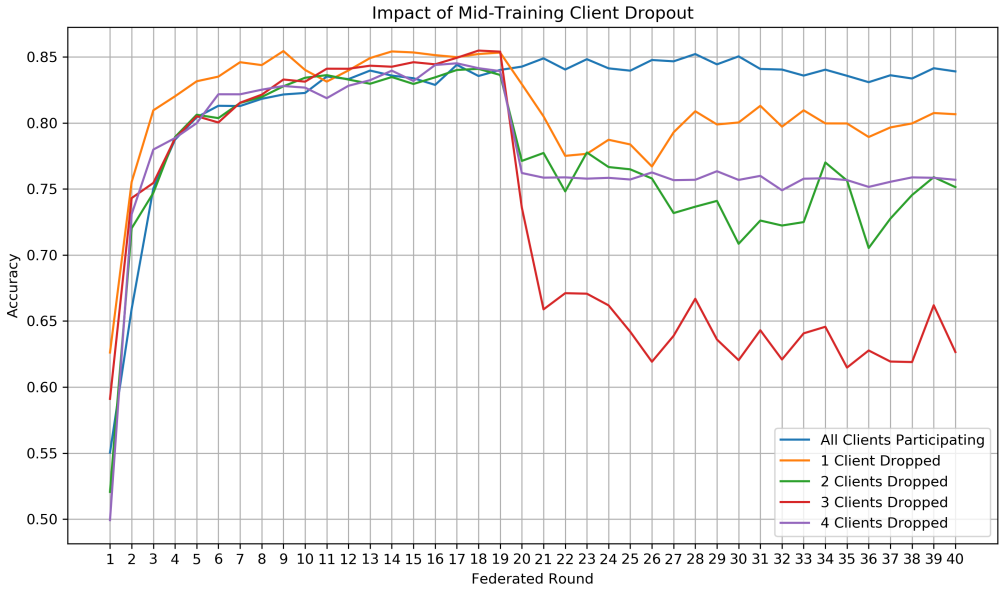


Figure 5.16. *Effect of Permanent Client Dropout on Model Accuracy and Stability*

The results are illustrated in Figure 5.16. When only one or two clients are removed after round 20, the impact on model performance is relatively modest. Although the final accuracy levels off sooner and does not continue improving as it does in the full-client scenario, the model still manages to maintain a respectable accuracy in the range of 81–83%. However, when three clients drop out—leaving only two active participants—the degradation becomes more pronounced. Accuracy declines to approximately 75%, and the model begins to exhibit greater variance across rounds, signaling reduced stability. The most severe scenario occurs when four clients are dropped, leaving only a single device to continue training. In this case, convergence effectively halts, and accuracy drops sharply to around 65%. This dramatic collapse underscores the critical importance of client diversity in federated learning: with data coming from just one user, the model lacks the heterogeneity required to generalize well, resulting in overfitting and poor performance. These findings highlight a critical challenge in federated learning: resilience to adversarial attacks. The significant performance degradation observed in the absence of honest participants underscores the system's vulnerability to malicious clients. This raises concerns about the robustness of federated learning against data poisoning or model manipulation attacks, even though this experiment focused on a completely different issue. Addressing this vulnerability is also an important direction for future research.

Chapter 6

Conclusions and Future Directions

6.1 Summary of Contributions

This thesis evaluated the application of the FL paradigm in a real cross-device scenario, focusing specifically on the HAR task. In contrast to the majority of existing FL literature, which heavily relies on simulation-based environments and synthetic datasets, this work addressed the practical constraints, trade-offs, and systemic limitations encountered when deploying FL on real mobile hardware. To support this evaluation, a complete end-to-end FL system was developed, using Android smartphones as client devices for on-device training (using TFLite) and a server implemented with the Flower framework. All devices were connected via a local Wi-Fi network, enabling real-time communication and coordination between clients and the central server during the FL process. The system was tested using six publicly available HAR datasets, selected for their diversity in sensing modalities, preprocessing techniques, and activity types. From the experiments, the following core insights were derived:

- **Impact of Non-IID Data on Model Performance:** Client scaling experiments (Chapter 5.2.5) demonstrated that even when data was relatively balanced and collected under controlled conditions, differences in subject behavior, sensor positioning, and hardware characteristics led to significant convergence variance.
- **Energy Efficiency Through Training Configuration:** Although HAR can be addressed using relatively lightweight neural networks, energy consumption remains a significant concern. Energy experiments (Chapter 5.3.1) showed that increasing the number of local training epochs while reducing the number of communication rounds reduced the total energy usage by up to 6×, without compromising model accuracy. This demonstrates a practical trade-off for energy-aware FL deployments.
- **Sensitivity to Network Variability:** Network experiments revealed that FL performance is highly sensitive to client dropout and intermittent availability. Even moderate reductions in client participation introduced instability and delayed convergence. These results underscore the need for more robust aggregation algorithms to ensure reliability in real-world scenarios.

In summary, this thesis demonstrates both the potential and limitations of FL in realistic edge-device scenarios, while through the experimental findings, it provides empirical

guidance for future research directions.

6.2 Limitations

Although this work provides valuable insights, it also has certain limitations that should be acknowledged:

- **System Heterogeneity:** Although devices used in the experiments varied in hardware capabilities and Android OS versions, no dedicated experiments were conducted to isolate or quantify the impact of system heterogeneity on convergence speed or model performance. While the study explores the three key aspects of FL (performance, energy consumption, and network behavior), this fourth dimension remains an important unexplored area. An interesting experiment would be to investigate the impact of devices with limited computational capabilities relative to the rest of the client pool on the overall system performance and convergence
- **Partial Energy Data:** Energy measurements were only available for three out of five devices. For the two devices with missing or unreliable sensor data, energy consumption was estimated using the average of the remaining devices. While this approximation provided a practical workaround, it may not reflect the true energy dynamics of those devices.
- **Limited Device Scale:** The evaluation was conducted on five Android smartphones, which is sufficient to demonstrate cross-device behavior but does not capture the complexities and scalability challenges of large-scale deployments which may involve even millions of edge devices.

6.3 Directions for Future Research

Future work can be organized into two primary directions: **i)** extensions to the existing framework and **ii)** broadening the scope of experimentation.

The first direction involves enhancing the current FL framework by incorporating more robust aggregation algorithms, such as FedProx and SCAFFOLD, and evaluating their performance against the baseline FedAvg approach within the HAR context. These methods are specifically designed to address non-IID data distributions which is the most critical challenges identified in this thesis. As shown in the client scaling experiment with non-IID data (Chapter 5.2.5), evaluating these algorithms under realistic conditions may provide deeper insight into their potential for improving convergence and system stability.

The second direction involves expanding the scope of experimentation to explore new dimensions of FL behavior. For instance, the label skew experiments (Chapter 5.2.2) raise the question of how effectively FL systems can propagate label knowledge when certain classes are completely missing from some clients. Improving this type of cross-client knowledge transfer could significantly enhance model performance in highly skewed or fragmented data settings. Similarily, the quantity skew experiments (Chapter 5.2.3) suggest

that further exploration is needed into how uneven data distributions impact collaboration among clients. In particular, it would be valuable to examine whether clients with limited local data can benefit from those with richer datasets, and how this imbalance influences global model performance.

Another key area for future research concerns the relationship between model architecture and energy efficiency. While this thesis explored varying model sizes, the connection between architectural complexity and energy consumption remains underexplored. As demonstrated in Chapter 5.3.2 future work should investigate how model architecture choices affect both accuracy and energy efficiency. Constructing energy-to-accuracy trade-off curves could be valuable tool to identify the optimal balance for resource-constrained edge deployments.

Finally, results from the network-related experiments suggest that current FL workflows are highly sensitive to partial client participation. As shown in Chapter 5.4.1, decreasing client availability due to intermittent connectivity can lead to increased instability and reduced model accuracy, especially under non-IID data distributions. Exploring more robust aggregation algorithms, such as FedProx or SCAFFOLD, may help mitigate this degradation by better handling the effects of data heterogeneity. Chapter 5.4.2 further highlights the challenge of permanent client dropout, where the removal of participants led to significant performance losses. Although the experiment did not explicitly target adversarial attacks, the results highlight FL's vulnerability to scenarios involving malicious clients, where techniques such as model poisoning can significantly degrade model performance. Addressing these robustness challenges is essential for enabling reliable FL deployment in real-world, non-IID, and potentially adversarial environments.

Bibliography

[1] Saqib Nawaz, Jahar Bhowmik, Tanya Linden και Matthew Mitchell. *Exploring the impact of smartphone dependency on real-life recreational activities: A theory of planned behaviour study*. *Entertainment Computing*, 52:100906, 2025.

[2] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[3] European Parliament and Council of the European Union. *Regulation (EU) 2019/679 of the European Parliament and of the Council*
newline on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). https://gdpr-info.eu/, 2019. Online; accessed 18-May-2025.

[4] U. S. Department of Health and Human Services. *HIPAA for Professionals*. Website, Office for Civil Rights, 2025. Accessed 18 May 2025.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson και Blaise Aguera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*Aarti Singh και Jerry Zhu, επιμελητές, τόμος 54 στο *Proceedings of Machine Learning Research*, σελίδες 1273–1282. PMLR, 2017.

[6] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari και Salman Avestimehr. *Federated Learning for Internet of Things: Applications, Challenges, and Opportunities*. arXiv preprint arXiv:2111.07494 [cs.LG], 2021. Version 4.

[7] Qinbin Li, Yiqun Diao, Quan Chen και Bingsheng He. *Federated Learning on Non-IID Data Silos: An Experimental Study*. arXiv preprint arXiv:2102.02079 [cs.LG], 2021. Version 4.

[8] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. *University of Toronto*, 2012.

[9] Ons Aouedi, Alessio Sacco, Latif U. Khan, Dinh C. Nguyen και Mohsen Guizani. *Federated Learning for Human Activity Recognition: Overview, Advances, and Challenges*. *IEEE Open Journal of the Communications Society*, 5:7341–7367, 2024.

[10] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarquede Gusmão και Nicholas D. Lane. *Flower: A Friendly Federated Learning Research Frame-*

work. arXiv preprint arXiv:2007.14390 [cs.LG], 2020. Open-source, mobile-friendly federated learning framework (version 1).

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu και Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org.

[12] Alex Krizhevsky, Ilya Sutskever και Geoffrey Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Neural Information Processing Systems*, 25, 2012.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser και Illia Polosukhin. *Attention Is All You Need*. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017. 15 pages, 5 figures.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, Georgevan den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel και Demis Hassabis. *Mastering the game of Go with deep neural networks and tree search*. *Nature*, 529(7587):484–489, 2016.

[15] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Jackson Ekstrand και others. *Advances and Open Problems in Federated Learning*. arXiv preprint arXiv:1912.04977 [cs.LG], 2019. Version 3.

[16] Andrew Hard, Kanishka Rao, Rajiv Mathews, Sreeram Ramaswamy, Françoise Beaufays, Stephen Augenstein, Hubert Eichner και others. *Applied Federated Learning: Improving Google Keyboard Query Suggestions*. arXiv preprint arXiv:1812.02903 [cs.LG], 2018. Version 1.

[17] Karen Hao. *How Apple Personalizes Siri Without Hoovering Up Your Data*. MIT Technology Review, online article, 2019. Accessed 18 May 2025.

[18] Qiang Yang, Yang Liu, Tianjian Chen και Yongxin Tong. *Federated Machine Learning: Concept and Applications*. arXiv preprint arXiv:1902.04885 [cs.AI], 2019. Version 1.

[19] Hao Guan, Pew Thian Yap, Andrea Bozoki και Mingxia Liu. *Federated learning for medical image analysis: A survey*. *Pattern Recognition*, 151:110424, 2024.

[20] Md. Saikat Islam Khan, Aparna Gupta, Oshani Seneviratne και Stacy Patterson. *Fed-RD: Privacy-Preserving Federated Learning for Financial Crime Detection*. arXiv preprint arXiv:2408.01609 [cs.CE], 2024. Version 1.

[21] Gustav A. Baumgart, Jaemin Shin, Ali Payani, Myungjin Lee και Ramana Rao Kompella. *Not All Federated Learning Algorithms Are Created Equal: A Performance Evaluation Study*. arXiv preprint arXiv:2403.17287 [cs.LG], 2024. Version 1.

[22] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar και Virginia Smith. *Federated Optimization in Heterogeneous Networks*. arXiv preprint arXiv:1812.06127 [cs.LG], 2018. Presented at MLSys 2020, version 5.

[23] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich και Ananda Theertha Suresh. *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*. arXiv preprint arXiv:1910.06378 [cs.LG], 2019. Version 4.

[24] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li και M. Hadi Amini. *A Survey on Federated Learning for Resource-Constrained IoT Devices*. *IEEE Internet of Things Journal*, 9(1):1–24, 2022.

[25] Xiaoxu Wen, Yan Wang, Menghao Yuan, Yingrui Geng, Hongnian Yu και Ge Zheng. *Enhancing Human Activity Recognition With FedPA: Focusing on Non-IID Data Challenges in Federated Learning*. *IEEE Sensors Journal*, 24(23):39230–39242, 2024.

[26] Google Open Source. *TensorFlow Federated*. https://www.tensorflow.org/federated, 2025. Accessed 18 May 2025.

[27] Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan και Qiang Yang. *FATE–LLM: An Industrial Grade Federated Learning Framework for Large Language Models*. arXiv preprint arXiv:2310.10049 [cs.LG], 2023. Version 1.

[28] Pascal Riedel, Lukas Schick, Reinholdvon Schwerin, Manfred Reichert, Daniel Schaudt και Alexander Hafner. *Comparative analysis of open-source federated learning frameworks - a literature-based survey and review*. *International Journal of Machine Learning and Cybernetics*, 15(11):5257–5278, 2024.

[29] OpenMined Community. *Let's Solve Privacy*. https://openmined.org/, 2025. Accessed 18 May 2025.

[30] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram και Salman Avestimehr. *FedML: A Research Library and Benchmark for Federated Machine Learning*. arXiv preprint arXiv:2007.13518 [cs.LG], 2020. Version 1.

[31] Chaoyang He, Murali Annavaram και Salman Avestimehr. *Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search*. arXiv preprint arXiv:2004.08546 [cs.LG], 2020. Accepted to CVPR 2020 workshop on Neural Architecture Search and Beyond for Representation Learning.

[32] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh και Dave Bacon. *Federated Learning: Strategies for Improving Communication Efficiency*. arXiv preprint arXiv:1610.05492 [cs.LG], 2016.

[33] Cong Xie, Sanmi Koyejo και Indranil Gupta. *Asynchronous Federated Optimization*. arXiv preprint arXiv:1903.03934 [cs.DC], 2019. Version 5.

[34] Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro Porto Buarquede Gusmão, Yan Gao, Daniel J. Beutel, Taner Topal, Akhil Mathur και Nicholas D. Lane. *A First Look into the Carbon Footprint of Federated Learning*. arXiv preprint arXiv:2102.07627 [cs.LG], 2021. arXiv admin note: substantial text overlap with arXiv:2010.06537.

[35] Thomas Tsouparopoulos και Iordanis Koutsopoulos. *Implementation of Federated Learning on Resource-constrained devices: Lessons learned*. *2022 IFIP Networking Conference (IFIP Networking)*, σελίδες 1–6, 2022.

[36] Akhil Mathur, Daniel J. Beutel, Pedro Porto Buarquede Gusmão, Javier Fernandez-Marques, Taner Topal, Xinchi Qiu, Titouan Parcollet, Yan Gao και Nicholas D. Lane. *On-device Federated Learning with Flower*. arXiv preprint arXiv:2104.03042 [cs.LG], 2021. Accepted at the 2nd On-device Intelligence Workshop @ MLSys 2021.

[37] Tuo Zhang, Chaoyang He, Tianhao Ma, Lei Gao, Mark Ma και Salman Avestimehr. *Federated Learning for Internet of Things: A Federated Learning Framework for On-device Anomaly Data Detection*. arXiv preprint arXiv:2106.07976 [cs.LG], 2021. Version 4.

[38] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher και Yuval Elovici. *N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders*. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[39] Dilraj N, Rakesh K, Rahul Krishnan και Maneesha Ramesh. *A Low Cost Remote Cardiac Monitoring Framework for Rural Regions*. *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*, MOBIHEALTH'15, σελίδα 231–236, Brussels, BEL, 2015. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[40] Erin E. Dooley, Natalie M. Golaszewski και John B. Bartholomew. *Estimating Accuracy at Exercise Intensities: A Comparative Study of Self-Monitoring Heart Rate and Physical Activity Wearable Devices*. *JMIR mHealth and uHealth*, 5(3):e34, 2017. Conflicts of Interest: None declared.

[41] Pietro Spadaccino και Francesca Cuomo. *Intrusion Detection Systems for IoT: Opportunities and Challenges Offered by Edge Computing and Machine Learning*. *arXiv preprint arXiv:2012.01174*, 2020. Paper submitted for publication in ITU Journal on Future and Evolving Technologies (ITU J-FET).

[42] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras και Helge Janicke. *Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning*. *IEEE Access*, 10:40281–40306, 2022.

[43] Rahim Taheri, Mohammad Shojafar, Mamoun Alazab και Rahim Tafazolli. *Fed-IIoT: A Robust Federated Malware Detection Architecture in Industrial IoT*. *IEEE Transactions on Industrial Informatics*, 17(12):8442–8452, 2021.

[44] Jessilyn Dunn, Ryan Runge και Michael Snyder. *Wearables and the medical revolution*. *Personalized Medicine*, 15(5):429–448, 2018.

[45] Alfredo J. Perez και Sherali Zeadally. *Privacy Issues and Solutions for Consumer Wearables*. *IT Professional*, 20(4):46–56, 2018.

[46] Aite Zhao, Junyu Dong, Jie Li, Lin Qi και Hui Yu. *LSTM for diagnosis of neurodegenerative diseases using gait data*. σελίδα 242, 2018.

[47] Ons Aouedi, Alessio Sacco, Kandaraj Piamrat και Guido Marchetto. *Handling Privacy-Sensitive Medical Data With Federated Learning: Challenges and Future Directions*. *IEEE Journal of Biomedical and Health Informatics*, 27(2):790–803, 2023.

[48] Akane Sano, Sara Taylor, Andrew W. McHill, Andrew J. K. Phillips, Laura K. Barger, Elizabeth Klerman και Rosalind Picard. *Identifying Objective Physiological Markers and Modifiable Behaviors for Self-Reported Stress and Mental Health Status Using Wearable Sensors and Mobile Phones: Observational Study*. *Journal of Medical Internet Research*, 20(6):e210, 2018. Conflicts of Interest: R. Picard is a co-founder of Affectiva and Empatica; E. Klerman has consulted for legal firms and Pfizer.

[49] Brandy Warwick, Nicholas Symons, Xiao Chen και Kaiqi Xiong. *Detecting Driver Drowsiness Using Wireless Wearables*. *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, σελίδες 585–588, 2015.

[50] Yuchen Zhao, Hanyang Liu, Honglin Li, Payam Barnaghi και Hamed Haddadi. *Semi-supervised Federated Learning for Activity Recognition*. *arXiv preprint arXiv:2011.00851*, 2020.

[51] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan και Yan Zhang. *Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT*. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2020.

[52] Toshiyo Tamura, Yuka Maeda, Masaki Sekine και Masaki Yoshida. *Wearable Photoplethysmographic Sensors—Past and Present*. *Electronics*, 3(2):282–302, 2014.

[53] Denisse Castaneda, Aibhlin Esparza, Mohammad Ghamari, Cinna Soltanpur και Homer Nazeran. *A Review on Wearable Photoplethysmography Sensors and Their Potential Future Applications in Health Care*. *International Journal of Biosensors & Bioelectronics*, 4(4):195–202, 2018. Conflict of interest: none declared.

[54] Gary Weiss. *WISDM Smartphone and Smartwatch Activity and Biometrics Dataset* . UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5HK59.

[55] Jeremy Zhang. *Dynamic Time Warping — Explanation and Code Implementation*, 2020. Accessed: 2025-05-23.

[56] Scott Small, Sara Khalid, Paula Dhiman, Shing Chan, Dan Jackson, Aiden Doherty και Andrew Price. *Impact of Reduced Sampling Rate on Accelerometer-Based Physical Activity Monitoring and Machine Learning Activity Classification*. *Journal for the Measurement of Physical Behaviour*, 4(4):298 – 310, 2021.

[57] Andreas Bulling, Ulf Blanke και Bernt Schiele. *A tutorial on human activity recognition using body-worn inertial sensors*. *ACM Comput. Surv.*, 46(3), 2014.

[58] Oscar D. Lara και Miguel A. Labrador. *A Survey on Human Activity Recognition using Wearable Sensors*. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, 2013.

[59] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi και Uzoma Rita Alo. *Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges*. *Expert Systems with Applications*, 105:233–261, 2018.

[60] Daniel Garcia-Gonzalez, Daniel Rivero, Enrique Fernandez-Blanco και Miguel R. Luaces. *A Public Domain Dataset for Real-Life Human Activity Recognition Using Smartphone Sensors*. *Sensors*, 20(8), 2020.

[61] Boyou Wang. *Signal Processing Based on Butterworth Filter: Properties, Design, and Applications*. *Highlights in Science, Engineering and Technology*, 97:72–77, 2024.

[62] Nurul Amin Choudhury, Soumen Moulik και Diptendu Sinha Roy. *HARSense: Statistical Human Activity Recognition Dataset*, 2021.

[63] Attila Reiss και Didier Stricker. *Introducing a New Benchmarked Dataset for Activity Monitoring*. *2012 16th International Symposium on Wearable Computers*, σελίδες 108–109, 2012.

[64] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro και Hamed Haddadi. *Protecting Sensory Data against Sensitive Inferences*. arXiv preprint arXiv:1802.07802, 2018. 6 pages.

[65] Garcia Rafael Banos, Oresti και Alejandro Saez. *MHEALTH*. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5TW22.

[66] Michal Karas, Jan Urbanek, Ciprian Crainiceanu, Jaroslaw Harezlak και William Fadel. *Labeled raw accelerometry data captured during walking, stair climbing and driving (version 1.0.0)*, 2021.

[67] Flower Labs. *Flower: Android Example for Federated Learning*. https://github.com/adap/flower/tree/main/examples/android, 2025. Accessed: 2025-06-09.

[68] Rachid El Mokadem, Yann Ben Maissa και Zineb El Akkaoui. *Federated Learning for Energy Constrained IoT Devices: A Systematic Mapping Study*. arXiv preprint arXiv:2301.03720 [cs.LG], 2023. 27 pages, version 1.

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
NN	Neural Network
SVM	Support Vector Machine
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
FL	Federated Learning
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
IID	Independent and Identically Distributed
FedAvg	Federated Averaging
FATE	Federated AI Technology Enabler
gRPC	gRPC Remote Procedure Call
HAR	Human Activity Recognition
IoT	Internet of Things
DDoS	Distributed Denial of Service

MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer
TCP	Transmission Control Protocol
MRI	Magnetic Resonance Imaging
NLP	Natural Language Processing
RAM	Random Access Memory
CPU	Central Processing Unit
SCAFFOLD	Stochastic Controlled Averaging for Federated Learning
TFF	TensorFlow Federated
ADL	Activities of Daily Living
ECG	Electrocardiogram
EDA	Electrodermal Activity
FFT	Fast Fourier Transform
IMU	Inertial Measurement Unit
LPF	Low Pass Filter
NIDS	Network Intrusion Detection Systems
PPG	Photoplethysmography
PSD	Power Spectral Density
CSV	Comma-Separated Values
FN	False Negative
FP	False Positive

TN	True Negative
TP	True Positive
UI	User Interface
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
RSSI	Received Signal Strength Indicator
TFLite	TensorFlow Lite
dBm	Decibel-Milliwatts
mAh	Milliampere-Hour
mWh	Milliwatt-Hour
OS	Operating System
CIFAR-10	Canadian Institute For Advanced Research 10-class dataset
GPS	Global Positioning System