ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ, ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

&

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΕΠΙΚΟΙΝΩΝΙΩΝ
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΗΣ ΑΝΑΠΤΥΞΗΣ

ΠΡΟΓΡΑΜΜΑ ΔΙΔΑΚΤΟΡΙΚΩΝ ΣΠΟΥΔΩΝ ΜΕ ΣΥΝΕΠΙΒΛΕΨΗ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ομόσπονδη, πολυπρακτορική, βαθιά και ενισχυμένη μάθηση

ΑΘΑΝΑΣΙΟΣ Γ. ΨΑΛΤΗΣ

ΑΙΓΑΛΕΩ

ΙΟΥΝΙΟΣ 2024

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING,
FACULTY OF ENGINEERING, UNIVERSITY OF WEST ATTICA

&

INFORMATION TECHNOLOGIES INSTITUTE,
CENTRE FOR RESEARCH AND TECHNOLOGY - HELLAS

PROGRAM OF DOCTORAL STUDIES UNDER CO-SUPERVISION

PhD Thesis

Federated, Multi-agent, Deep Reinforcement Learning

ATHANASIOS G. PSALTIS

ATHENS-EGALEO

JUNE 2024

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ομόσπονδη, πολυπρακτορική, βαθιά και ενισχυμένη μάθηση

Αθανάσιος Γ. Ψάλτης

ΣΥΝ-ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Χαράλαμπος Πατρικάκης - Καθηγητής, Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχ., ΠαΔΑ
ΣΥΝ-ΕΠΙΒΛΕΠΩΝ ΕΡΕΥΝΗΤΗΣ: Πέτρος Δάρας - Ερευνητής Α', ΙΠΤΗΛ, ΕΚΕΤΑ

ΤΡΙΜΕΛΗΣ ΣΥΜΒΟΥΛΕΥΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Χαράλαμπος Πατρικάκης - Καθηγητής, Τμ. ΗΗΜ, ΠαΔΑ
Πέτρος Δάρας - Ερευνητής Α', ΙΠΤΗΛ, ΕΚΕΤΑ
Δημήτριος Ζαρπαλάς - Ερευνητής Β', ΙΠΤΗΛ, ΕΚΕΤΑ

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

(Υπογραφή)

[illegible]

Digitally signed by
CHARALAMPOS PATRIKAKIS
Date: 2024.07.18 18:23:21 +03'00'

Χαράλαμπος Πατρικάκης,
Καθηγητής, Τμ. ΗΗΜ, ΠαΔΑ

(Υπογραφή)

Petros Daras

Petros Daras
17.07.2024 20:06

Πέτρος Δάρας,
Ερευνητής Α', ΙΠΤΗΛ, ΕΚΕΤΑ

(Υπογραφή)

[illegible]

Digitally signed by DIMITRIOS ZARPALAS
DN: c=GR, serialNumber=ERMIS-30853945,
sn=ZARPALAS, givenName=DIMITRIOS,
cn=DIMITRIOS ZARPALAS
Date: 2024.07.17 22:07:12 +03'00'

Δημήτριος Ζαρπαλάς,
Ερευνητής Β', ΙΠΤΗΛ, ΕΚΕΤΑ

(Υπογραφή)

ELENI
AIKATERINI
LELIGKOU

Digitally signed by ELENI
AIKATERINI LELIGKOU
Date: 2024.07.19
09:24:05 +03'00'

Ελένη Αικατερίνη Λελίγκου,
Καθηγήτρια, Τμ. ΜΒΣΠ, ΠαΔΑ

(Υπογραφή)

ΠΑΡΑΣΚΕΥΗ
ΖΑΧΑΡΙΑ

Digitally signed by ΠΑΡΑΣΚΕΥΗ
ΖΑΧΑΡΙΑ
Date: 2024.07.19 10:26:08 +03'00'

Παρασκευή Ζαχαριά,
Επίκουρη Καθηγήτρια, Τμ. ΜΒΣΠ, ΠαΔΑ

(Υπογραφή)

ΧΡΙΣΤΟΦΟΡ
ΟΣ ΚΑΧΡΗΣ

Digitally signed by
ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ
Date: 2024.07.19 19:12:46
+03'00'

Χριστόφορος Κάχρης,
Επίκουρος Καθηγητής, Τμ. ΗΗΜ, ΠαΔΑ

(Υπογραφή)

Athanasios
Voulodimos

Digitally signed by Athanasios
Voulodimos
Date: 2024.07.24 09:17:57 +03'00'

Αθανάσιος Βουλόδημος,
Επίκουρος Καθηγητής, Σχ. ΗΜΜΥ, ΕΜΠ

Ημερομηνία εξέτασης 21/06/2024

PhD THESIS

Federated, Multi-agent, Deep Reinforcement Learning

Athanasios G. Psaltis

CO-SUPERVISOR from UNIWA: Charalampos Patrikakis, Professor UniWA
CO-SUPERVISOR from ITI-CERTH: Petros Daras, Researcher A', CERTH-ITI

ADVISORY COMMITTEE:

Charalampos Patrikakis, Professor UniWA
Petros Daras, Researcher A', CERTH-ITI
Dimitrios Zarpalas, Researcher B', CERTH-ITI

EXAMINATION COMMITTEE

(Signature)

[illegible] Digitally signed by CHARALAMPOS PATRIKAKIS Date: 2024.07.18 18:23:48 +03'00'

Charalampos Patrikakis,
Professor UniWA

(Signature)

Petros Daras Petros Daras 17.07.2024 20:08

Petros Daras,
Researcher A', CERTH-ITI

(Signature)

[illegible] Digitally signed by DIMITRIOS ZARPALAS DN: c=GR, serialNumber=ERMIS-30853945, sn=ZARPALAS, givenName=DIMITRIOS, cn=DIMITRIOS ZARPALAS Date: 2024.07.17 22:07:47 +03'00'

Dimitrios Zarpalas,
Researcher B', CERTH-ITI

(Signature)

ELENI AIKATERINI LELIGKOU Digitally signed by ELENI AIKATERINI LELIGKOU Date: 2024.07.22 10:38:22 +03'00'

Helen C. Leligou,
Professor UniWA

(Signature)

ΠΑΡΑΣΚΕΥΗ ΖΑΧΑΡΙΑ Digitally signed by ΠΑΡΑΣΚΕΥΗ ΖΑΧΑΡΙΑ Date: 2024.07.19 10:26:39 +03'00'

Paraskevi Zacharia,
Assistant Professor UniWA

(Signature)

ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Digitally signed by ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Date: 2024.07.24 08:28:47 +03'00'

Christoforos Kachris,
Assistant Professor UniWA

(Signature)

Athanasios Voulodimos Digitally signed by Athanasios Voulodimos Date: 2024.07.24 09:18:31 +03'00'

Athanasios Voulodimos,
Assistant Professor NTUA

Examination Date 21/06/2024

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Αθανάσιος Ψάλτης του Γεωργίου, υποψήφιος διδάκτορας του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:
«Είμαι συγγραφέας και δικαιούχος των πνευματικών δικαιωμάτων επί της διατριβής και δεν προσβάλω τα πνευματικά δικαιώματα τρίτων. Για τη συγγραφή της διδακτορικής μου διατριβής δεν χρησιμοποίησα ολόκληρο ή μέρος έργου άλλου δημιουργού ή τις ιδέες και αντιλήψεις άλλου δημιουργού χωρίς να γίνεται αναφορά στην πηγή προέλευσης (βιβλίο, άρθρο από εφημερίδα ή περιοδικό, ιστοσελίδα κ.λπ.). Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

[illegible]

Περίληψη

Το τοπίο της τεχνητής νοημοσύνης (ΤΝ) αναδιαμορφώνεται από την Ομόσπονδη Μάθηση (ΟΜ), μια αποκεντρωμένη προσέγγιση στη μηχανική μάθηση (ΜΜ) που ενισχύει την ιδιωτικότητα δεδομένων και τη συνεργατική εκπαίδευση μοντέλων. Αυτή η διατριβή εξετάζει τις προκλήσεις και τις δυνατότητες της ΟΜ, επικεντρώνοντας στην βελτιστοποίηση της αποδοτικότητας επικοινωνίας, την ενίσχυση της απόδοσης μοντέλου και τη διασφάλιση της ανθεκτικότητας σε διάφορα περιβάλλοντα.

Η έρευνα περιλαμβάνει μια λεπτομερή ανασκόπηση της λογοτεχνίας και την ταυτοποίηση των βασικών προκλήσεων στην ΟΜ. Διεξήχθησαν μια σειρά από μελέτες για να αντιμετωπιστούν συγκεκριμένες πτυχές: η βελτιστοποίηση της μετάδοσης δεδομένων και η διαχείριση διάφορων αρχιτεκτονικών μοντέλων, η κατανομή δεδομένων και η επιλογή κόμβων, η μάθηση αναπαράστασης και η ομόσπονδη απόσταξη, η σταδιακή μάθηση και η διατήρηση γνώσης, καθώς και η εκπαίδευση μοντέλων με περιορισμένα δεδομένα. Κάθε επιμέρους μελέτη συνέβαλε στον τομέα αναπτύσσοντας καινοτόμους αλγορίθμους, τους οποίους δοκίμασε σε προσομοιωμένα περιβάλλοντα ΟΜ και συνέκρινε με υπάρχουσες μεθόδους.

Τα κύρια ευρήματα της έρευνας περιλαμβάνουν τη βελτίωση της αποδοτικότητας της επικοινωνίας με μειωμένες απαιτήσεις υπερφόρτωσης και εύρους ζώνης, την ενίσχυση της απόδοσης του μοντέλου στη διαχείριση ετερογενών δεδομένων και μεταβλητότητας της αρχιτεκτονικής του μοντέλου, αποτελεσματικές στρατηγικές για την αντιμετώπιση της καταστροφικής λήθης και μεθοδολογίες ευέλικτες σε περιορισμένα και διασκορπισμένα δεδομένα. Η εφαρμοσιμότητα της ΟΜ αποδείχθηκε σε πρακτικά σενάρια, επιδεικνύοντας τη δυναμική της σε διάφορους τομείς.

Συμπερασματικά, η διατριβή συνεισφέρει σημαντικά στην προώθηση της ΟΜ. Αντιμετωπίζει θεμελιώδεις προκλήσεις και αποδεικνύει την προσαρμοστικότητα και την αποτελεσματικότητα της ΟΜ σε εφαρμογές του πραγματικού κόσμου. Τα ευρήματα τονίζουν τον ρόλο της ΟΜ ως μεθόδου που εξασφαλίζει την ιδιωτικότητα, αυξάνει την αποδοτικότητα και επιδεικνύει ευελιξία στον τομέα της τεχνητής νοημοσύνης και της μηχανικής μάθησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: 11
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ομόσπονδη Μάθηση, Αποδοτικότητα Επικοινωνίας, Ετερογένεια Αρχιτεκτονικής Μοντέλου, Ετερογενή Δεδομένα, Σταδιακή Μάθηση, Ιδιωτικότητα Δεδομένων.

Abstract

The landscape of artificial intelligence (AI) is being reshaped by Federated Learning (FL), a decentralized approach to machine learning (ML) that enhances data privacy and collaborative model training. This thesis delves into the challenges and potential of FL, focusing on optimizing communication efficiency, enhancing model performance, and ensuring robustness in diverse settings.

The research encompasses a detailed literature review and the identification of core challenges in FL. A series of studies were conducted to address specific aspects: optimizing data transmission and handling diverse model architectures, data partitioning and client selection, representation learning and federated distillation, incremental learning and knowledge retention, and training models with limited data. Each study contributed to the field by developing innovative algorithms, tested in simulated FL environments and compared with existing methods.

Key findings of the research include improved communication efficiency with reduced overhead and bandwidth requirements, enhanced model performance in handling heterogeneous data and model architecture variability, effective strategies to combat catastrophic forgetting, and methodologies adept at working with limited and scattered data. The applicability of FL was demonstrated in practical scenarios, showcasing its potential in various domains.

In conclusion, the dissertation significantly contributes to the advancement of FL. It addresses foundational challenges and demonstrates the adaptability and efficacy of FL in real-world applications. The findings emphasize FL's role as a method that ensures privacy, boosts efficiency, and showcases flexibility in the field of AI and ML.

SUBJECT AREA: 11
KEYWORDS: Federated Learning, Communication Efficiency, Model Heterogeneity, Non-IID Data, Incremental Learning, Data Privacy.

To my newborn son—
for the precious hours we've missed,
and for the countless moments we are yet to share..

Ευχαριστίες

Embarking on a PhD is a journey of discovery, made possible by the guidance and support of many. At the forefront, my advisor, Petros Daras, has been instrumental in shaping my research skills with his remarkable foresight and commitment to excellence. His patience and guidance through numerous iterations have been invaluable. His mentorship has been a cornerstone of my development. I am equally grateful to Prof. Charalampos Patrikakis, my co-advisor, whose expertise and encouragement have been pivotal in refining my ideas into substantial research. Together, Petros and Charalampos have been the architects, transforming my raw ideas into research of substance, and helping me navigate the world of study with great insight. The Visual Computing Lab of CERTH has been a vital part of my PhD journey, full of inspiration and teamwork. Working with the talented people there has greatly enriched my experience. Many talks and pair-programming sessions, especially with Spyridon Thermos, have boosted my skills and sparked a strong interest in the vast potential of deep learning. I extend my gratitude to my ex-supervisors Prof. Robby T. Tan, Prof. Manolis Vavalis, and Prof. Elias Houstis, for their influential perspectives and inspiration. Lastly, my heartfelt appreciation goes to my family—my wife Mariadora, my parents, and my sister—for their unwavering support and belief in me throughout this journey.

Λίστα Δημοσιευσεων

Λίστα των δημοσιεύσεων του φοιτητή σε συνέδρια/περιοδικά που έγιναν στα πλαίσια της εκπόνησης της διδακτορικής διατριβής.
Δημοσιεύσεις σε Επιστημονικά Περιοδικά:

1. **Psaltis, A.**, Zafeirouli, K., Leškovský, P., Bourou, S., Vásquez-Correa, JC., García-Pablos, A., Cerezo Sánchez, S., Dimou, A., Patrikakis, CZ., and Daras, P. Fostering Trustworthiness of Federated Learning Ecosystem through Realistic Scenarios. Information. 2023; 14(6):342. https://doi.org/10.3390/info14060342

Δημοσιεύσεις σε Επιστημονικά Συνέδρια:

1. **Psaltis, A.**, Patrikakis, C. Z., and Daras, P. (2022, October). Deep Multi-modal Representation Schemes for Federated 3D Human Action Recognition. In European Conference on Computer Vision (pp. 334-352).

2. **Psaltis, A.**, Kastellos, A., Patrikakis, C. Z., and Daras, P. (2023). FedLID: Self-Supervised Federated Learning for Leveraging Limited Image Data. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1039-1048).

3. **Psaltis, A.**, Chatzikonstantinou, C., Patrikakis, C. Z., and Daras, P. (2023). FedRCIL: Federated Knowledge Distillation for Representation based Contrastive Incremental Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 3463-3472).

Υποβληθείσες Εργασίες σε Επιστημονικά Περιοδικά:
N/A
Υποβληθείσες Εργασίες σε Επιστημονικά Συνέδρια:

1. Chatzikonstantinou, C., **Psaltis, A.**, Patrikakis, C. Z., and Daras, P. “FedFMRL: Federated Distillation by Feature Mixing for Representation based Learning” ECCVW 2024

2. Kastellos, A., **Psaltis, A.**, Patrikakis, C. Z., and Daras, P. “FedHARM: Harmonizing Model Architectural Diversity in Federated Learning” ECCV 2024

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Πίνακας Ορολογίας

Ξενόγλωσσος όρος	Ελληνικός Όρος
Federated Learning (FL)	Ομόσπονδη Μάθηση
Knowledge Distillation	Απόσταξη Γνώσης
Representation Learning	Μάθηση Αναπαράστασης
Incremental Learning	Σταδιακή Μάθηση
Self-Supervised Learning	Αυτό-Εποπτευόμενη Μάθηση
Data Partitioning	Διαμέριση Δεδομένων
Machine Learning (ML)	Μηχανική Μάθηση
Artificial Intelligence (AI)	Τεχνητή Νοημοσύνη
Edge Computing	Υπολογιστική Στο Άκρο
Internet of Things (IoT)	Διαδίκτυο των Πραγμάτων
Data Heterogeneity	Ετερογένεια Δεδομένων
Client Selection	Επιλογή Κόμβου
Adversarial Attacks	Εχθρικές Επιθέσεις
Data Poisoning	Δηλητηρίαση Δεδομένων
Model Convergence	Σύγκλιση Μοντέλου
Communication Efficiency	Αποδοτικότητα Επικοινωνίας
Data Privacy	Απόρρητο Δεδομένων
Model Personalization	Εξατομίκευση Μοντέλου
Differential Privacy	Διαφορικό Απόρρητο
Homomorphic Encryption	Ομομορφική Κρυπτογράφηση
Blockchain Technology	Τεχνολογία Αλυσίδας Συστοιχιών
Anomaly Detection	Ανίχνευση Ανωμαλιών
Sustainability	Βιωσιμότητα
Carbon Footprint	Αποτύπωμα Άνθρακα
Ethical AI	Ηθική Τεχνητή Νοημοσύνη
Regulatory Compliance	Συμμόρφωση με Κανονιστικές Απαιτήσεις

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

ΑΕΙ	Ανώτατο Εκπαιδευτικό Ίδρυμα
ΕΛΛΑΚ	Ελεύθερο Λογισμικό Λογισμικό Ανοιχτού Κώδικα
Η/Η	Ηλεκτρολόγων/Ηλεκτρονικών
ΠΑΔΑ	Πανεπιστήμιο Δυτικής Αττικής
FLS	Federated Learning Strategies
SMC	Secure Multiparty Computation
FL	Federated Learning
AI	Artificial Intelligence
ML	Machine Learning
IoT	Internet of Things
5G	Fifth Generation (of mobile network technology)
IID	Independent and Identically Distributed
Non-IID	Non-Independent and Identically Distributed
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Unit
CLR	Contrastive Learning
BYOL	Bootstrap Your Own Latent
GAN	Generative Adversarial Network
FedAvg	Federated Averaging
FedCA	Federated Contrastive Aggregation
FedSimCLR	Federated SimCLR
MOON	Model-Oriented Online Network
FedMatch	Federated Matching
FedX	Federated X (generic term)
FedCon	Federated Consistency
SelfFed	Self-Supervised Federated Learning
Adam	Adaptive Moment Estimation
ImageNet1k	ImageNet Dataset with 1000 classes

Chapter 1

Introduction

1.1 Overview of the Dissertation Topic

Federated Learning: A Paradigm Shift in Collaborative AI

Federated Learning (FL) represents a transformative approach in the field of Artificial Intelligence (AI) and Machine Learning (ML). It is a decentralized learning technique that enables multiple participants or devices to collaboratively train a shared model while keeping their data local. This method not only addresses privacy concerns but also leverages distributed data sources efficiently [2]. FL's unique attributes – data privacy, collaborative learning, and decentralized architecture – make it an attractive solution for a wide range of applications, from healthcare and finance to autonomous vehicles and IoT devices [3]. This paradigm is particularly relevant in today's digital era, where data privacy and security are paramount, and the volume of data generated by edge devices is colossal.

In contrast to conventional methods that require data centralization, FL harnesses the power of distributed datasets across numerous devices, enabling learning directly at the data source. In FL, data remains on local devices (such as smartphones, IoT devices, or organizational servers) [4]. The ML model is sent to these devices, where it learns from the local data and then only the model updates (and not the data itself) are sent back to the central server. This approach is in absolute contrast to traditional ML, where data from all sources is usually centralized for model training. Therefore, significantly reducing the risks associated with data transfer, storage, and potential breaches. Moreover, FL is inherently designed to operate under limited bandwidth and varying network conditions, making it particularly suited for edge computing scenarios where data is generated in vast quantities at the network's edge.

One of the foremost advantages of FL is its inherent privacy-preserving nature [5]. By design, FL ensures that the raw data remains confined to its original location, typically the user's device, and never gets transmitted or centralized. This fundamental aspect of FL not only mitigates privacy risks but also fortifies data security, making it an exceptionally suitable approach for industries handling sensitive information, such as healthcare and finance.

FL facilitates collaborative yet independent learning, offering a novel paradigm in the realm of ML. It allows multiple participants, each with their distinct datasets, to collaboratively contribute to the development of a comprehensive and robust model. This balanced combination of collective intelligence and private data control is the cornerstone of FL's approach. Each participant in an FL network trains a shared model locally on their dataset, thereby contributing to the learning process without exposing their data. These local updates are then aggregated to refine and improve a global model. This mechanism ensures that the model benefits from a wide range of data inputs, encompassing diverse patterns and scenarios, which significantly enhances its performance and generalizability. As a result, FL not only fosters a cooperative learning environment but also respects and maintains the independence and privacy of individual data sources.

In real-world scenarios, data is often distributed unevenly and can be highly heterogeneous in nature [6]. FL is uniquely suited to handle such diversity, as it allows each participant to train models on their specific, local dataset, contributing to a more inclusive and representative global model. This global model benefits from the diverse insights derived from each local dataset, ensuring that it is not just informed by a single type of data but is instead representative of a wide array of data sources. This approach allows FL to create models that are more adaptable and effective in practical situations, where data diversity is the norm rather than the exception.

The focus of this dissertation is on exploring and addressing the various challenges and opportunities that FL presents. It aims to delve into strategies for optimizing communication efficiency, enhancing model performance, and ensuring robustness in diverse environments. Given the increasing importance of data privacy and the rapid growth of edge computing, FL is expected to play a crucial role in the future of AI and ML, making this research both timely and significant. This dissertation aims to contribute to the growing body of knowledge in FL, providing insights, methodologies, and applications that can help harness the full potential of this emerging paradigm.

1.2 Research Motivation and Objectives

1.2.1 Bridging Gaps and Harnessing Data Diversity

The motivation behind this research stems from the growing need for privacy-preserving AI solutions and the challenges associated with centralized data processing. FL emerges as a promising solution to these challenges, yet it is not without its own set of complexities and barriers [7]. These include issues related to data heterogeneity, communication efficiency, model aggregation, and performance optimization under varying constraints.

In an era where data is becoming increasingly valuable and sensitive, the traditional methods of centralizing data for ML are no longer viable. This has motivated the exploration of FL as a solution that respects user privacy and data security [8]. The primary objective is to demonstrate how FL can effectively train models without compromising the confidentiality of the data. Moreover, the increasing regulatory demands for data protection, exemplified by legislations like the General Data Protection Regulation (GDPR), Data Act [1]or Data Governance Act [2], further underscore the urgency of finding solutions that align with these privacy requirements. FL presents an opportunity to comply with such regulations while still leveraging the collective power of distributed data sources for ML. Another motivation behind this research is the need to address the challenges posed by data accessibility and heterogeneity. Data is often siloed within organizations and devices, making it difficult to access for centralized training. FL provides a unique opportunity to leverage this disparate data, facilitating collaborative learning without the need for data consolidation. The objective here is to showcase how FL can handle diverse data distributions and varying data quality across different nodes to collaboratively train robust and generalizable ML models.

The decentralized nature of FL poses significant challenges in terms of maintaining and improving the model performance. This is particularly true when considering the dynamic nature of model training over time, a concept central to the study of model incrementality [9]. In a federated setting, where data is continuously evolving and new information is constantly being introduced, it's crucial for models to not only retain previously learned knowledge but also efficiently incorporate new insights. This concept of model incrementality, or continual learning, is a key focus of the research. Addressing these challenges involves developing novel algorithms and strategies that not only enhance model accuracy and efficiency but also enable the models to adapt and evolve incrementally. The objective is to refine FL algorithms to handle the incremental nature of learning, where the model is continually updated and improved as new data becomes available across the network of devices.

With the rapid increase of IoT devices and edge computing, there's a growing need for efficient on-device ML. FL is inherently suited for this purpose as it allows for local model training on edge devices [10]. This research is motivated by the desire to advance the field of edge computing by developing FL techniques that are optimized for such environments. Moreover, the variability in computational architectures across devices adds another layer of complexity. Devices participating in an FL network can range from high-powered servers to edge devices with limited processing capabilities. This diversity necessitates adaptable and flexible model architectures that can efficiently operate across a wide spectrum of computational resources. The research thus also aims to address how FL can be tailored to accommodate and optimize for varying computational architectures, ensuring that the collaborative learning process is efficient and effective regardless of the hardware capabilities of each participating node.

Finally, this research is driven by the broader goal of preparing for the future landscape

[1]https://digital-strategy.ec.europa.eu/en/policies/data-act
[2]https://digital-strategy.ec.europa.eu/en/policies/data-governance-act

of AI and ML, where privacy, data security, and efficient use of resources will be paramount. By exploring and addressing the challenges within FL, this dissertation aims to contribute to the development of sustainable, secure, and efficient AI systems for the future, that align with the evolving demands of our digital society.

1.2.2 Objectives

This dissertation explores the frontiers of FL, a paradigm designed to harness decentralized data while enhancing privacy and efficiency. It delves into novel methodologies for handling data diversity, optimizing model performance in complex scenarios, and demonstrating FL's versatility across a range of real-world applications.

Exploring Efficient Learning Strategies: Investigating novel methodologies in FL to handle data diversity and enhance learning efficiency. The primary objective under this theme is to delve into the development of innovative methodologies within FL that can efficiently handle diverse data distributions. This involves creating algorithms that can learn effectively from decentralized data sources, ensuring that the models trained are as accurate and efficient as possible. The goal is to overcome the challenges posed by data and system heterogeneity and to enhance the learning efficiency of FL systems, making them suitable for a wide range of applications and environments.

Optimizing Model Performance: Developing techniques to optimize the performance of FL models in diverse and realistic scenarios. This objective is centered around the enhancement of FL model performance, particularly in scenarios that mirror real-world complexities and constraints. The primary aim is to develop algorithms that not only enhance computational efficiency and model accuracy but also address the crucial aspects of fast and stable model convergence, efficient communication, and robustness against Non-Independent and Identically Distributed (Non-IID) data. A significant focus is on devising strategies that can effectively utilize sparse data, incorporating techniques like semi-supervised, self-supervised, and transfer learning to extract maximal information from limited datasets. Additionally, adapting FL models for resource-constrained environments is key, requiring lightweight model architectures that maintain high performance despite computational and data limitations. The overarching goal is to ensure that FL models are optimized to perform seamlessly across a spectrum of settings, from data-rich to data-scarce scenarios, thus enhancing the applicability and effectiveness of FL in various real-world applications.

Addressing Real-world Applications: Demonstrating the versatility of FL across real-world challenges, this research objective focuses on applying FL to various practical tasks, such as image classification, named entity recognition, and speech recognition. These applications are pivotal across numerous industries, going beyond domain-specific boundaries and offering a horizontal applicability to different sectors. The aim is to illustrate the transformative potential of FL in handling different types of data while maintaining data privacy and efficiency. This involves deploying FL for different data modalities and evaluating its performance in these varied contexts. By successfully applying FL to these cross-domain tasks, this research underlines the adaptability and effectiveness of FL, showcasing its potential as a game-changer in industries where data is distributed and privacy is paramount.

In summary, the objectives outlined above aim to advance the field of FL by enhancing its efficiency, privacy, security, and real-world applicability. Through innovative research and practical implementations, this dissertation endeavors to establish FL as a key player in the future landscape of AI and ML.

1.3 Dissertation Structure and Summaries of Associated Research Studies

This dissertation is structured to provide an in-depth exploration of FL. It begins with an introduction that lays the foundation for the topic, detailing the dissertation's overarching themes, motivations, and objectives. This section includes a discussion on the potential of leveraging data diversity and bridging gaps in FL, followed by a clear delineation of the dissertation's objectives. Additionally, it presents an overview of the dissertation structure and a summary of the studies included. The body of the dissertation is divided into multiple detailed studies, each addressing a specific aspect of FL.

The second chapter serves as a background and literature review, transitioning from general ML concepts to the more specialized domain of distributed learning, summarizing existing studies in FL. This includes a deep dive into various strategies employed in FL, challenges, and opportunities within the field.

Subsequent chapters, from the third to the tenth, are dedicated to individual studies focusing on different facets of FL, covering principles, privacy-preserving techniques, data modality heterogeneity, representation learning, incremental learning, and model architecture variability. Each chapter is dedicated to a specific study, meticulously structured with an overview, methodology, experimental setup, results, analysis, and concludes with insightful discussions and findings. The series investigates FL's application across domains like image classification, named entity recognition, and 3D action recognition, addressing challenges from data distribution to knowledge transfer and model heterogeneity. It progresses to a practical evaluation of FL in real-world scenarios, showcasing its adaptability, efficiency, and resilience, thereby demonstrating FL's potential for dynamic, privacy-preserving, and robust distributed learning in diverse environments and applications.

The final chapter synthesizes the contributions of the dissertation, discusses limitations, and suggests future research directions. Appendices provide additional technical details on implementation tools, best practices, and guidelines, as well as a focused look at principal threats and mitigation strategies in FL, enhancing the practical applicability of the dissertation.

Before diving into the detailed chapters of this dissertation, here we briefly overview the key studies that underpin this research. These summaries provide a snapshot of each study's main focus and findings, offering a clear roadmap for the key research areas addressed in this dissertation, setting the stage for the detailed exploration and discussion in the subsequent chapters.

Study 1: The study explores the practical application of FL in two key areas: facial classification in Computer Vision and named entity recognition in Text Categorization. It begins with small-scale experiments to test the effectiveness, accuracy, and limitations of FL in various settings, focusing particularly on the challenges of data distribution, both IID and non-IID. The experimental phase is methodically structured, progressing from single-node, limited data scenarios to more complex, highly heterogeneous data distributions across federated nodes. This approach helps assess the adaptability and scalability of FL models, revealing their performance under different data conditions. The study concludes with valuable insights for advancing FL applications in fields marked by diverse and uneven data distributions.

Study 2: This comparative study investigates various FL strategies, in the context of 3D action recognition, focusing on the impact of different data modalities on system performance. Key findings include the variable effectiveness of FL strategies in handling non-IID data distributions and the challenges in multi-modal federated optimization, particularly in terms of communication overheads and data synchronization. The study reveals the limitations of traditional FL schemes under complex multi-modal scenarios. The importance of multi-modal data fusion in enhancing FL performance is emphasized, but integrating diverse datasets remains a challenge. The study highlights the need for innovative FL techniques in multi-modal human action recognition to achieve more accurate, efficient, and privacy-preserving systems.

Study 3: The third study introduces a novel method for federated distillation weight aggregation in distributed learning environments. This approach efficiently transfers knowledge across federated network nodes, using an advanced algorithm for capturing and consolidating representations from client nodes at a central server. Key innovations include an advanced federated distillation scheme, a new aggregation method using feature mixing and a trainable vector for optimal weight assignment, and extensive validation of the approach's effectiveness across diverse FL scenarios. This research significantly enhances the efficiency and effectiveness of knowledge transfer in FL systems.

Study 4: The fourth study in this series extends into incremental learning within computer vision, focusing on overcoming catastrophic forgetting in continual learning scenarios. It introduces a novel Federated Incremental Learning approach, integrating multi-scale representation learning with Knowledge Distillation techniques. This method eliminates the need for central model transfer, a common bottleneck in traditional FL. Key innovations include a rehearsal-based knowledge distillation technique for efficient knowledge transfer across nodes and a contrastive learning algorithm for retaining knowledge across different tasks. Extensive experimentation validates the effectiveness of this approach, demonstrating significant improvements in communication efficiency and robust performance in distributed incremental learning scenarios. This research represents a major advancement in FL, offering scalable and dynamic learning models for distributed environments.

Study 5: The fifth study builds on previous advancements in federated distillation and incremental learning to address challenges in FL environments with limited and unevenly distributed data. It introduces a novel approach that combines self-supervised and supervised learning within a federated framework, specifically tailored for scenarios with sparse and imbalanced data. The study employs a custom self-supervised learning strategy at the global level for unlabeled data and supervised learning at the local level for labeled data. This combination leads to a robust and versatile learning mechanism suitable for various federated settings. Key contributions include a self-supervised learning approach for effective global data utilization, a hybrid FL scheme that integrates self-supervised and supervised techniques, and extensive validation showing the superiority of this approach over fully-supervised methods in standard federated scenarios. This research marks a significant advancement in FL, particularly in handling limited and scattered data efficiently.

Study 6: This part of the research addresses the challenge of managing diverse model architectures in FL, specifically focusing on models like ResNets, EfficientNets, and MobileNets. The study introduces a model-agnostic methodology that emphasizes representation learning over traditional federated averaging. Driven by previous studies, key components include a combination of supervised and self-supervised learning across both private and shared datasets, allowing models to adapt to local and global data contexts. The central node plays a crucial role in aggregating representations from various models, moving away from Federated Averaging to a more representation-focused approach. The study also explores feature excitation techniques for aligning local models with a central model and conducts experiments across different architectures to assess their impact on FL. Overall, this research presents a flexible and efficient strategy for managing model heterogeneity in FL, enhancing the robustness of learning across federated networks.

Study 7: In the final study, the focus shifts to a practical evaluation of FL in real-world scenarios. This comprehensive study tests the FL system's adaptability and resilience in diverse environments, using a distributed hardware infrastructure to simulate real-world conditions. It introduces a system adaptation pipeline for integrating AI tools into FL, conducts experiments across edge devices, and addresses challenges like data heterogeneity and privacy. Findings reveal FL's performance under diverse conditions, its resilience to data poisoning attacks, and its potential for real-world applications. This study bridges theory and practice, highlighting FL's viability and challenges in real-world contexts, signaling its potential for trustworthy applications.

Chapter 2

Core Concepts of Federated Learning

2.1 Transitioning from Machine Learning to Distributed Learning

ML and Deep Learning (DL) are two interconnected fields that have revolutionized many aspects of technology, particularly in computer vision. ML, a subset of AI, involves the development of algorithms that can learn from and make predictions or decisions based on data. These algorithms improve their performance as they are exposed to more data over time [11]. Traditional ML techniques include linear regression, decision trees, and support vector machines, which are powerful for a variety of tasks but often require manual feature selection and tuning.

DL, a more advanced subset of ML, refers specifically to algorithms inspired by the structure and function of the brain called artificial neural networks. DL has gained immense popularity due to its ability to automatically and adaptively learn spatial hierarchies of features from data, which is particularly useful in the field of computer vision. This approach is known as the 'Deep Learning' paradigm in computer vision [12]. DL models, especially Convolutional Neural Networks (CNNs), have shown exceptional performance in multiple image analysis tasks such as object detection, concept detection, and image classification, outperforming traditional ML methods in many cases [13]. The development of DL has been further enhanced by the application of these techniques to video-classification and action-recognition problems. For instance, the use of 3-dimensional convolution networks for spatio-temporal feature learning and the application of recurrent neural networks for dynamic scene representation are notable advancements [14]. Furthermore, innovations like two-stream architectures for faster training and more effective temporal information fusion indicate the ongoing evolution and refinement of DL models in understanding and interpreting visual data [15].

The remarkable advancements in ML and DL, particularly in the realm of computer vision, hinge significantly on the availability of large volumes of data. This 'big data' phenomenon, often centralized in nature, poses unique challenges and opportunities. As the number of smart devices connected to the internet continues to grow exponentially, they collectively generate an immense stream of data [16]. This surge in data production from countless sources, including sensors, smartphones, and other IoT devices, emphasizes the need for advanced and sophisticated learning approaches. These approaches must not only efficiently process and learn from such vast datasets but also address concerns related to data privacy, security, and the ethical use of information. The big data issue [17], therefore, is not just about managing and analyzing large datasets, but also about innovatively leveraging this information to further refine and enhance ML models, ensuring that these technological advancements continue to evolve in a responsible and sustainable manner.

Access to high-quality data is crucial for the success of data-driven applications in AI, as it enables the creation of advanced models through data manipulation techniques like regularization and optimization. However, traditional ML approaches often require centralized training data, which can pose privacy concerns, especially with sensitive information (e.g., medical data) that is protected by laws like GDPR. To address these privacy issues, FL was introduced by Google [18].

FL is a form of distributed ML wherein each participant, or node, maintains data locally. It operates under a learning protocol that facilitates interaction among nodes without centralizing data. This approach primarily addresses privacy and confidentiality, adheres to regulatory compliance, and recognizes the impracticality of moving large datasets to a central location for processing [5]. In contrast to FL, traditional ML models centralize training data, either on a single machine or within a data center. This process, particularly in DL, involves stages like data acquisition, training, and model optimization, all of which traditionally rely on a degree of trust among collaborators. In scenarios where data is shared among organizations and data scientists, any breach of trust can compromise the integrity of the project [19]. In the context of DL, where extensive regularization and optimization are critical, decentralizing the lifecycle of applications presents an opportunity. FL enables collaboration in model development without direct data access, fostering a secure, trust-less environment for parties involved [20].

FL decentralizes the ML process to edge devices, allowing various authorities to collaboratively develop a shared predictive model while keeping training data local. This approach is particularly beneficial for geographically dispersed authorities, as it ensures that sensitive data, potentially containing private information, remains on local devices. FL has opened new research avenues in ML, especially considering the vast amounts of data generated by smart devices. This data, rich in personal and sensitive information, is key to building robust ML models. FL offers a way to utilize this data while preserving user privacy, posing new challenges in large-scale ML, distributed optimization, and privacy-preserving data analytics.

2.2 Introduction to Federated Learning

Originally, the concept of FL was introduced primarily for applications involving mobile and edge devices. However, there has been a growing interest in applying FL to a broader range of scenarios, some of which may include only a few, but relatively reliable, clients. The case studies chosen are representative of such scenarios, where a group of devices needs to collectively gain insights by analyzing their data, yet are unable to share this data directly. In these instances, the devices collaborate to train an ML model using their collective dataset.

2.2.1 FL Topologies and Design Principles

In FL, the training process is distributed across a network of nodes, each possessing a unique arrangement that significantly influences the training dynamics. The network topology in FL is crucial because it affects the distribution and aggregation algorithms used for the model, the required number of communication links for effective training, and the overall system design cost for setting up and maintaining the training process. FL can operate both with and without a central server, and the choice of topology plays a pivotal role in this aspect. In a Star-like topology, nodes connect to a central server that coordinates the process and aggregates model updates, making it efficient in communication but reliant on a central point. The Ring-like topology, in contrast, arranges nodes in a sequential loop, promoting direct, sequential communication between nodes without a central server, potentially increasing robustness but requiring more complex communication protocols. Lastly, the Hybrid topology combines elements of both, seeking to balance the advantages of centralized and decentralized approaches, optimizing both communication efficiency and system resilience [21].

Figure 2.1 illustrates a typical FL system comprising a global or federated server and numerous clients, denoted as 'N' clients. In this setup, clients contribute by training models locally and sending these updated models to the global server. The server's role is to aggregate these individual model updates. During each round of federation, the global server distributes the global model to each client. Clients then train this model using their own private data. Once training is completed, they transmit the updated model parameters back to the server. The server integrates these updates to form an enhanced global model. This cycle is repeated for a predetermined number of federated rounds to continually refine the model.

Star-like Topology: This represents a network configuration where each participating node is directly connected to a central server, typically a trusted third-party entity. In this setup, every node communicates individually with the server, allowing for a clear, one-to-one communication channel. The core concept of standard FL is to learn a unified global statistical model from data distributed across a wide range of remote devices, potentially numbering from tens to millions. The key constraint in this model is that the data generated by each device are processed locally, and only intermediate model

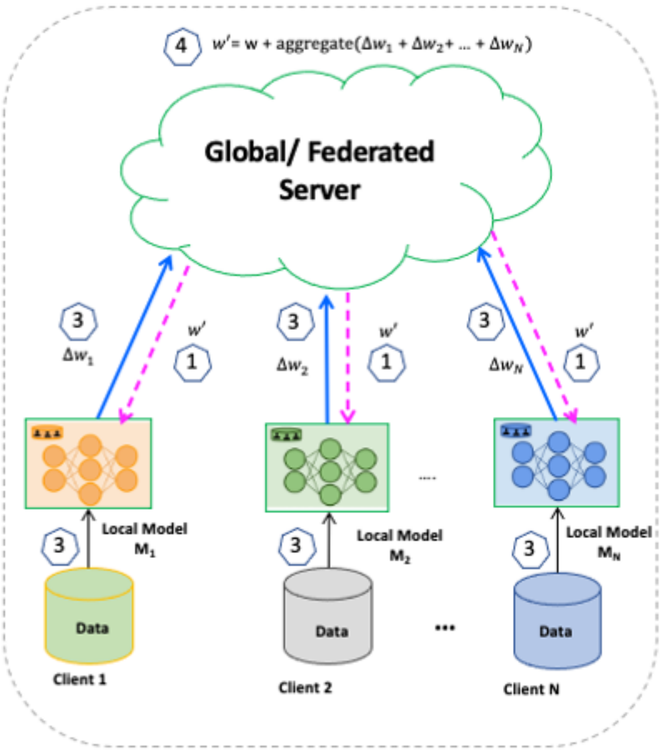


Figure 2.1: A standard FL system, in which N clients or nodes exchange model updates with a centralized global server.

updates are periodically sent to the central server.

The process unfolds as follows: Each device downloads the current global model, enhances it by learning from its local data, and then sends a summary of these improvements as a compact update back to the server. This communication is secured, often using techniques like homomorphic encryption, to ensure privacy. The central server then averages these updates to refine the shared global model. Notably, all training data remains on the local device, and the server does not retain individual updates.

The star-like topology in FL facilitates asynchronous learning, meaning the training process on one node is independent of others, eliminating the need for nodes to wait for their peers. This structure allows for easy scalability, as adding a new node to the network is straightforward. However, variability in the configuration of each edge device leads to differences in training speeds. Factors such as the computational capacity of the device and the volume of training samples available locally can influence the pace of learning. The more extensive the training data on a node, the more robust the model updates it can produce. Communication efficiency is another consideration; each node requires two communication links per training round—one to upload its updates to the server and another to download the latest global model. This dual-link setup is essential for maintaining the continuous and effective exchange of information between the central server and the nodes.

Ring-like Topology: Offers an alternative to the Star-like approach by forming a circular network where each device is connected to two others. This topology doesn't rely on a central server for management, instead using a cyclical weight transfer method. Here, each node processes the model locally and then passes its updated weights to the next node in the ring, continuing until the model converges. This synchronous training approach means each node's training is dependent on its peers, potentially lengthening the training cycle but allowing for immediate incorporation of updates from one node to the next.

This topology is more cost-effective than the Star-like topology, as it requires fewer communication links and no central server, reducing infrastructure costs. However, it demands similar computational capabilities across all nodes to prevent bottlenecks and ensure even training progress. The final, converged model is shared among all nodes at the end of the training process.

Hybrid Topology: Blends the Star-like and Ring-like topologies, integrating nodes and a server for optimized learning. Groups of nodes are formed based on specific criteria, such as geographical proximity, and are arranged sequentially within each group. The training process starts with a global model initialized by a centralized server, similar to the Star-like topology. This model is distributed to all nodes. Training within each group begins asynchronously, with each node sequentially processing and then passing model updates to the next node, reflecting the Ring-like approach.

This topology combines synchronous and asynchronous learning methods: groups start training asynchronously, while individual nodes within each group operate synchronously. Although a central server is involved, its computational burden is less than in a purely Star-like topology, as the workload is more distributed. If the number of groups equals the number of nodes, the setup essentially becomes akin to a Star-like topology. The Hybrid Topology thus offers a versatile and adaptable approach, balancing centralized coordination with localized inter-node training.

Data Partitioning Schemes

Federated Learning Strategies (FLS) are generally classified based on the distribution of data across the sample and feature spaces, falling into horizontal, vertical, or hybrid schemes [2]. In brief, horizontal FL involves datasets that share the same feature space across all devices. Vertical FL, on the other hand, employs disparate datasets with different feature spaces to collaboratively train a global model. Hybrid FL merges these two approaches, combining aspects of both feature and sample distribution.

Horizontal FL: Here, different parties' datasets share the same features but differ in samples (Figure 2.2A). Commonly in cross-device settings, parties train local models on their data and update the global model by averaging these local models. This approach, exemplified by frameworks like FedAvg [22], is often adopted when data centralization is not feasible due to legal constraints or when similar organizations aim to collaboratively

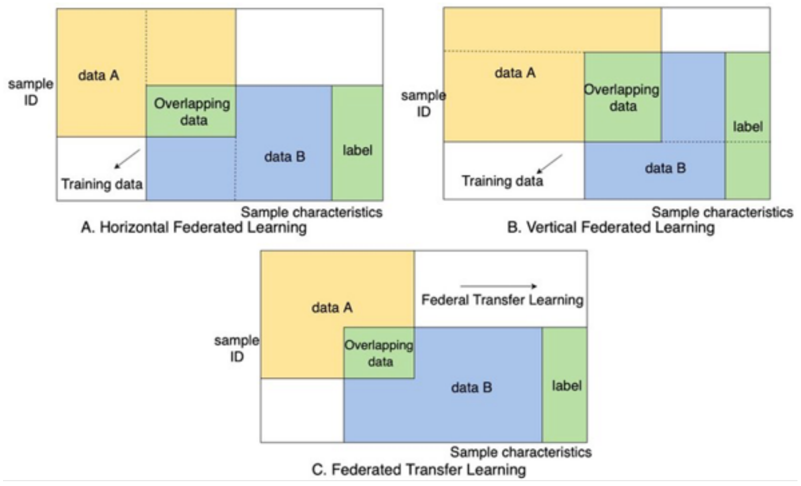


Figure 2.2: Federated Learning (FL) schemes based on data distribution.

enhance their models. Vertical FL: This type (Figure 2.2B) involves datasets with similar samples but different features across parties. Approaches in this category typically use entity alignment techniques [23, 24] to identify overlapping samples for training using encrypted methods. It's commonly used in scenarios where different companies cooperate, each contributing different feature sets to the model training. Hybrid FL: Combining elements of both horizontal and vertical partitioning (Figure 2.2C), this approach addresses more complex situations where parties have partial overlap in user or feature space. It often incorporates transfer learning techniques for collaborative model building. However, current methods, like the secure federated transfer learning system proposed by Liu *et al.* [25], tend to be limited in scope, such as being applicable to only two clients. Additional information regarding the implementation tools and various practical matters is provided in Appendix A.

2.2.2 Concepts and Terminology

The adoption of standardized terminology in FL plays a crucial role in enhancing interoperability and streamlining the exchange of information. Understanding and using a consistent set of terms is a straightforward yet vital step in grasping more complex FL concepts. Table 2.1 is a compilation of some common terminologies typically encountered in the FL framework.

2.3 Privacy-Preserving Principles

As opposed to traditional approaches, FL intrinsically enhances privacy and security as the data is never accessed or processed on central servers. Its goal is to enable different entities to collaboratively train a shared ML model while keeping all the training data on their premises; hence, decoupling the ML process from the data sources.

Table 2.1: FL related concepts and terminology.

Term	Definition / Description
Device	Refers to entities in the FL communication network like nodes or clients. In broader terms, organizations or institutions in FL are also considered as 'devices'. · Edge Devices: Points where data is generated and managed, and where the training process occurs. · Data Centre: Central locations for data creation and management, and training process execution.
Database	An organized collection of data or information. · Centralized Database: Data stored as a unified entity on a single server. · Decentralized Database: Multiple interconnected servers store and provide data without a central repository. · Distributed Database: Data is not centrally stored; each node in the network contains portions of the overall data.
Human Learning	The process of understanding problem aspects and knowing the rules to reach a solution.
Machine Learning	Involves designing systems that learn from examples. · Model: A trained file recognizing specific patterns, functioning as a parameter-tunable 'black box'. · Algorithm: Built around a modifiable mathematical function with internal parameters or weights. · Model Weights: Learnable parameters in some machine learning models. · Training: The learning process from data, where algorithms match outputs to inputs. · Inference: Making predictions using the trained model. · Validation: Evaluating a trained model with a testing dataset.
Neuron	A single learning unit in computational models, applying logic to inputs and producing outputs.
Neural network	A network of interconnected neurons influencing each other, functioning collectively in learning processes.
Artificial Intelligence (AI)	AI tools are based on machine learning algorithms that adapt and learn from data to perform specific tasks.
Deep Learning	A subset of machine learning mimicking human knowledge acquisition, effective with multiple neural network layers.
Federated Learning	A paradigm where an algorithm is trained across multiple devices or servers with local data samples, without data exchange.
Federation	A collective of computing or network providers adhering to shared operational standards. · Federated Node: An edge member of the federation, also known as a node, client, device, worker, or party. · Federated Aggregator: Coordinates trust among devices, also called a server, orchestrator, or central node. · Orchestration: Coordination activities in the application of learning technologies. · Data Aggregation: The process of combining data from distributed sources. · Data Anonymization: Removing identifiable information from data sets to maintain anonymity. · Data Re-Identification: The risk of matching anonymous data with public or auxiliary data to reveal private information. · Federated Data Analysis: Analyzing distributed datasets, also known as Federated Analysis or Federated Data Mining. · Client/Server Architecture: Shares models or statistical information with a central server in federated data analysis. · Decentralized Architecture: In federated data analysis, it allows for peer-to-peer information exchange without a central aggregating server.

In practice, FL distributes the model learning process to the users' devices, making it possible to train a global (or master) model from user-specific local models. Each user updates the local copy of the model using his/her data and sends an updated model to the server which uses the user's local model to update the master model. Instead of transferring the user data to processing servers, only the model updates are transferred. In other words, FL allows learning robust ML models from a huge amount of data distributed across isolated silos without transferring or processing it on central servers.

In FL systems, the data stays securely on the client's side, and the model training occurs locally. After training, these local models' parameters are sent to a central node for the aggregation of a global model. This process, known as Secure Aggregation, ensures that the specifics of the client's updates are not transparent to the aggregator. To maintain user privacy, secure aggregation protocols are incorporated in FL [26, 27]. These protocols enable the server to merge the local models into a global model without revealing individual local model details. The aggregation is done by calculating the combined parameters of the local models without directly accessing them. Consequently, the server cannot misuse the local model updates or deduce any private user data from them.

While FL offers flexibility and addresses issues related to data governance and ownership, it does not inherently ensure complete security and privacy. The absence of robust encryption methods could leave room for attackers to access personally identifiable information directly from the nodes or disrupt the communication flow. Furthermore, the distributed nature of the data in FL can make it challenging to maintain the data's integrity and quality, essential for reliable results. If local algorithms are not securely encrypted or the model updates aren't securely aggregated, there's a risk of data breaches or tampering. This could lead to algorithm reconstruction, theft (known as parameter inference), or leaks, which are critical concerns for many applications. Consequently, FL requires additional safeguards to defend against attack strategies like data poisoning, training data reconstruction (also referred to as model inversion), and interception of data [28].

2.3.1 Key Threats in FL

An FL system offers an initial layer of privacy protection over centralized learning models by keeping data local to the clients and only sending updated model parameters to the server. However, research, including findings from the study referenced in [27], suggests that FL doesn't fully secure privacy and data security. Even minimal exposure to original gradients might reveal significant information about local data. Furthermore, during the training phase, the exchange of model updates can potentially leak sensitive data, as indicated by studies [29] and [30]. To elevate the privacy and security standards of FL systems, it's crucial to explore potential vulnerabilities within the FL network. Additionally, implementing defense mechanisms against such attacks, as highlighted in various studies, is essential for enhancing the overall security of the FL framework.

During the training phase of FL networks, certain types of cyberattacks, known as poisoning attacks, can occur. These attacks are primarily focused on either corrupting the dataset (Data Poisoning) or manipulating the local model (Model Poisoning). The primary objective of these attacks is to alter the FL network's functionality in a detrimental manner, consequently degrading the accuracy and performance of the global ML model. In FL systems, these attacks can originate from either the central server or the FL system's client participants. As illustrated in Figure 2.3, data poisoning occurs during the local data collection phase, whereas model poisoning takes place at the stage of local model training. Additionally, FL systems are susceptible to other threats such as model inversion attacks and various backdoor attacks, including bug injections and inference attacks, which can exploit different processing steps in the system.

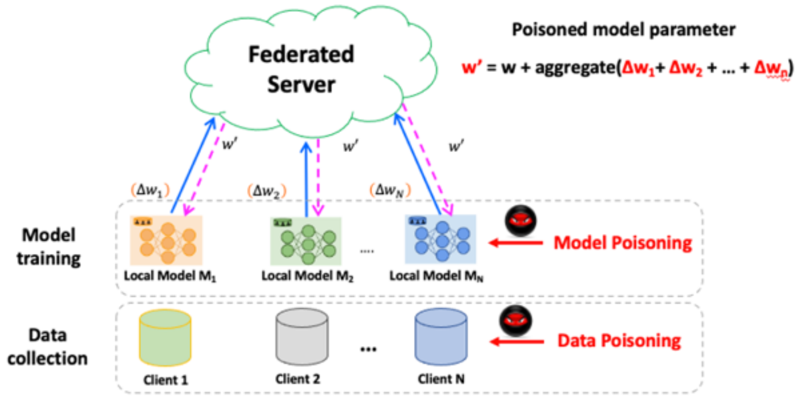


Figure 2.3: Infiltration in FL Systems: Here, an intruder targets either the model or the data in a poisoning assault.

Table 2.2 categorizes the primary risks to both the dataset - Data Poisoning (DP) and the algorithmic/computational process - Model Poisoning(MP). The table provides a detailed description and a fundamental example for each type of attack. For an in-depth examination, please see the detailed analysis in Appendix C.

2.3.2 Existing Mechanisms and Approaches

As discussed earlier, while FL allows for client-side ML, it doesn't automatically provide security and privacy guarantees. Although a significant advantage of FL is the non-sharing of private data with a central server, this doesn't preclude the possibility of adversaries extracting sensitive information from that data. The aforementioned threats necessitate the development of methods that not only provide stringent privacy guarantees but are also computationally efficient, effective in communication, and do not excessively compromise accuracy. We could broadly categorize these methods into two major groups: global methods, where model updates produced in each round are private from all untrusted third parties except the central server, and local methods, where updates

Table 2.2: Most relevant attacks within FL.

	Method	Asset threatened	Description	Example
DP	Re-identification Attack	Data privacy	Breaching data privacy by identifying individuals in a dataset that has been anonymized, utilizing additional information available within the dataset	Correlating data with other datasets where the same individuals are identifiable
	Dataset reconstruction attack	Data privacy	Inferring individual characteristics from the outputs of computations performed on a dataset, without direct access to the dataset itself. This is also known as feature re-derivation or attribute inference	Deducing individual data points using multiple aggregate statistics corresponding to a single individual
	Tracing attack	Data privacy	Discerning the presence (or absence) of an individual in a dataset without specifically identifying them, also known as membership inference	Making repeated, slightly varied queries to the dataset and employing set differencing techniques to isolate individual information.
MP	Adversarial attack	Model behaviour	Introducing manipulated training examples, effectively poisoning the model. These attacks are designed to be undetectable by human observation	Introducing biases, such as sexist or racist tendencies in model predictions, or creating backdoors, like spam filters that fail to recognize specific patterns.
	Model-inversion /reconstruction attack	Model behaviour	Analyzing the algorithm's behavior, particularly the information stored in its weights	Employing algorithms to reconstruct segments of the training data, relying on various model parameters

are kept private even from the central server. Key methodologies to bolster data security in FL encompass techniques like data anonymization, differential privacy (DP), homomorphic encryption (HE), and secure multiparty computation (SMC).

The design of a robust defence mechanism against data poisoning attacks in FL systems is a challenging task since most of them are attack-specific and have been designed for a specific type of attack and they do not work well for other types [31, 32, 33]. The case of non-IID data among FL clients introduces other challenges in the procedure of developing an efficient defence mechanism against data poisoning attacks. Specifically, the work in [34] presents that the non-IID data increases the difficulty of an accurate defence against data poisoning attacks. When the data is not identically distributed among FL participants, each client has its own data distribution, therefore bringing its unique contribution to the common FL model, which is misleading for most defence mechanisms. Regarding the model poisoning attacks, participants' private information can be extracted from the sharable weights throughout the training process. Therefore, sensitive information can be revealed either to third-party or to the central server.

Many mechanisms have been developed to enhance the privacy of FL, such as secure multiparty computation or differential privacy. The aforementioned techniques provide privacy for the cost of decreased ML model performance. A challenge that should be faced during the development of a secure FL system is understanding and balancing the trade-off between the privacy-preserving level and the achieved ML performance. The authors in [35] evaluate the performance of the proposed FL system under various settings of DP as a privacy-preserving technique and configurations of the FL nodes, investigating the trade-off between those components and achieved model performance. Specifically, they demonstrate how the model performance is affected at different levels of DP when the number of participants increases as well as in the case of imbalanced client data. Beyond providing an adequate level of privacy, it is also essential to implement computationally cheap FL methods, communication efficient as well as tolerant dropped devices without compromising accuracy.

The various privacy approaches of an FL system can be grouped into two categories, which are global privacy and local privacy. In the former, the central server is a trusted party and the model updates generated at each round are considered private to all untrusted participants except the central server. In the latter, all the participants may be malicious and, therefore, the updates are also private to the server. A very common approach to prevent leakage of private client data from the shared parameters is the utilisation of the secure aggregation mechanism of FL. In the last few years, various FL designs have been introduced that use secure aggregation protocols under various setups. Bonawitz in [36] introduces a secure aggregation mechanism for FL, which can tolerate client dropouts. This method uses Shamir's Secret Sharing [37] and symmetric encryption to prevent the server from accessing individual model updates. The limitation of this approach lies in the fact that it requires at least 4 communication rounds between each client and the aggregator in each iteration.

Bonawitz's protocol is utilised by the works VerifyNet [38] and VeriFL [39], which add an extra verification level on top of [36]. The additional verifiability guarantees the correctness of the aggregation. However, those methods require a trusted party to create private keys for all clients. Trying to reduce the overhead of [36], the algorithms in [40] and [41] present secure aggregation mechanisms with polylogarithmic communication and computation complexity. The main differences in [36] are that those methods replace the star-like topology of the FL network with random subgroups of clients as well as the secret sharing is only performed for a set of clients and not for all of them. Both methods [40] and [41] demand 3 rounds of communication interaction between the server and the clients. Another approach that reduces the communication and computation overhead compared to [36] is the Turbo-Aggregate [42]. Specifically, this method utilises a circular communication topology. The FastSecAgg [43] method uses Fast Fourier Transform multi-secret sharing for secure aggregation. FastSecAgg is robust against adversaries which adaptively corrupt clients during the execution of FL procedures.

Chapter 3

Advances and Challenges in Federated Learning

3.1 Strategies for Dealing with Federated Systems and Data

Federated Learning differs from Distributed Learning in several key aspects: firstly, FL typically experiences slower and less stable communication; secondly, it involves a diverse range of devices with varying computational powers, indicating heterogeneity; thirdly, FL places a greater emphasis on privacy and security [19]. While many studies assume trustworthiness in participants and servers, real-world scenarios often involve potentially untrustworthy actors [44]. Addressing these realities requires optimizing FL algorithms to tackle practical challenges effectively. The primary concerns for researchers in this optimization process include managing high communication costs and dealing with both statistical and structural heterogeneity[45].

3.2 Challenges and Opportunities in FL

In traditional centralized models, data from various sources must be aggregated in a single location, posing risks of data breaches and misuse [46]. FL, by allowing data to remain on users' devices and only sharing model updates, offers a more privacy-preserving approach. This method reduces the risk of exposing sensitive information while still enabling the collaborative development of robust ML models. However, FL also faces challenges related to data heterogeneity and communication efficiency [45]. Since data is not uniformly distributed across devices, the resulting models may be biased or underperforming due to non-IID (not independently and identically distributed) data. Additionally, the necessity of frequent communication between devices and a central server for model updates can lead to significant network overhead, particularly when dealing with large numbers of devices. On the opportunity side, FL opens up new avenues for leveraging data from a vast array of devices without infringing on user privacy. This approach is particularly advantageous in sectors like healthcare, forensics and finance, where data sensitivity is paramount [47]. By enabling decentralized training, FL allows for the creation of personalized models that are more tailored to individual users' needs and preferences. Furthermore, FL can lead to more inclusive AI systems [48]. Since data does not need to be centralized, models can be trained on a diverse set of data sources, potentially reducing biases inherent in many of today's AI systems. This decentralized approach also democratizes AI, allowing smaller entities to participate in and benefit from AI advancements without the need for extensive data infrastructure. The ability of FL to process data in situ also paves the way for real-time, on-device decision-making, enhancing the functionality and responsiveness of smart devices [49]. However, realizing these opportunities requires overcoming the technical complexities and ensuring robust, scalable, and secure implementations of FL systems.

3.2.1 Communication Efficiency Challenges

Managing communication overhead is a crucial challenge. Although an exhaustive review of communication-efficient distributed learning methods falls outside the scope of this study, it's worth noting some broad strategies. These can be categorized into (a) local updating methods, which focus on local computation to reduce the frequency of data transmission, and (b) compression schemes, aimed at reducing the size of the data being communicated. Given the rapid growth in dataset sizes, minimizing communication overhead is essential for maintaining the flexibility and efficiency of FL. Efforts to address this include cutting down the number of communication rounds and enhancing the speed of model uploads, which collectively contribute to decreasing the overall update time.

In traditional ML systems, algorithms like Stochastic Gradient Descent (SGD) operate on large datasets evenly distributed across cloud servers, requiring fast, efficient connections for iterative computations. However, in a FL environment, data is dispersed across numerous devices, often unevenly. To tackle this, FL strategies focus on local updates and infrequent central server communication. This approach aligns with the unique challenges of FL, including adhering to data locality requirements and managing the limited communication capacity of edge devices.

Recent advancements aim to enhance communication efficiency in FL. The goal is to minimize server-client communication to reduce data upload times. This is achieved by varying the number of local updates performed independently on each device during each communication round, offering greater flexibility in balancing computation and communication. Such methods have significantly boosted performance, often outpacing conventional distributed approaches. In FL, optimization techniques that support adaptable local updates and require minimal client involvement are increasingly preferred.

McMahan *et al.* [22] conducted pioneering research in FL, focusing on enhancing communication efficiency. Their approach involves increasing the amount of computation done by each client between communication rounds, using a method that averages local stochastic gradient descent (SGD) updates. They noted the effectiveness of increasing parallelism by encouraging more client participation in each training round. Building on this concept, Nishio and Yonetani [50] developed the FedCs framework. This system maximizes client engagement in every training round to improve practical efficiency. They integrated a maximum mean discrepancy into the FL algorithm, encouraging local models to learn more effectively from other training devices, thereby accelerating convergence. Yurochkin *et al*. [51] proposed the Bayesian Nonparametric FL framework, considered state-of-the-art due to its ability to merge local models into a federated model without additional parameters. This innovation reduces the need for extra communication rounds. Their experiments demonstrated that satisfactory accuracy could be achieved with just a single round of communication.

While optimizing communication rounds is vital, finding ways to speed up model updates remains a challenge. Initially, McMahan *et al*. [52] suggested two approaches to decrease the time taken for model updates. The first approach, structured updates, involves transmitting only a portion of the updated model. This can be done through a low-rank approximation or by using a random mask. Another example is an end-to-end neural network that compresses updated information into a lower-dimensional space, thereby easing communication burdens. The second approach, sketched updates, utilizes a compressed model for updates. In line with this strategy, Zhu and Jin [53] refined Sparse Evolutionary Training (SET) to transmit only select parameters to the server, similar to the concept of sketched updates. These methods collectively aim to reduce the amount of data transferred during model updates, thereby making the update process more efficient.

In an effort to optimize local training in each round where clients typically run a fixed number of epochs, Jiang and Ying [54] developed an adaptive approach. In this method, the number of local training epochs is dynamically determined by the server based on training time and loss. This adaptive strategy aims to reduce local training time, especially when the loss diminishes. While the aforementioned algorithms primarily rely on SGD, this approach might be less efficient in cases where the function to be optimized is anisotropic. Addressing this, Liu, Chen, Chen, and Zhang [55] implemented momentum gradient descent, which incorporates information from previous gradients in each local training epoch to quicken the pace of convergence.

Local update methods effectively decrease the frequency of communication rounds in FL, but model compression techniques like sparsification, subsampling, and quantization are key to reducing the size of data transmitted in each round. These compression strategies have been widely researched in the context of distributed training within data-center environments, both empirically and theoretically. In FL settings, unique challenges arise due to factors like low device participation rates, non-independent and identically distributed (non-iid) local data, and the application of local updating methods. To address these issues, several practical strategies have been proposed and implemented in federated contexts [52, 56, 57]. These include encouraging sparsity and low-rank structures in updating models, utilizing quantization methods that incorporate structured random rotation, employing lossy compression and dropout techniques to minimize communication from server to device, and applying Golomb lossless encoding. Each of these approaches aims to optimize data transmission efficiency while maintaining the integrity and effectiveness of FL models.

3.2.2 System-specific Challenges

In the context of federated networks, system heterogeneity presents a major challenge due to the wide variability in device characteristics across the network. Differences in hardware, network connectivity, and battery life among devices can lead to uneven training times and introduce complications such as stragglers, which are more common than in centralized systems. To address these challenges, current methods primarily concentrate on two areas: allocating resources appropriately for devices with varying capabilities and enhancing fault tolerance for devices that are more likely to go offline. These approaches aim to mitigate the impact of system heterogeneity and ensure more consistent and reliable training across the federated network.

In remote device learning, handling fault tolerance is crucial due to the likelihood of participant dropouts during training iterations. Smith *et al.* [58] addressed this by considering the impact of low participation to mitigate device dropouts. A common approach is to simply overlook device failures [59], though this could introduce bias if the dropped devices possess unique data characteristics. For instance, devices in remote areas with poor connectivity might drop out more frequently, potentially skewing the federated model towards data from devices with better network conditions. While numerous studies have explored the convergence of various FL methods, few have focused on the effects of low participation or the direct impact of device dropouts. To strengthen FL systems against participant dropouts, some researchers have developed secure aggregation protocols [60] that can tolerate arbitrary dropouts as long as a sufficient number of users remain. Lib *et al.* [61] and Wu *et al.* [62] also considered the straggler effect, with the former allowing varied local update times and the latter using a cache structure to store updates from unreliable users, thereby reducing their negative influence on the global model.

In addressing resource constraints in FL, much of the existing research has concentrated on effectively allocating resources across heterogeneous devices. Kang *et al.* [63] considered client overhead to encourage participation from higher-quality devices. Similarly, Nishio and Yonetani [50] developed device sampling policies that consider system resources, aiming to maximize device updates within a set time frame. Tran *et al.* [64] examined the effects of heterogeneous power constraints on training accuracy and convergence time, while Chai *et al.* [65] focused on how resource disparities, such as CPU, memory, and network availability, impact FL training time. To promote fair resource distribution, Li, T. *et al.* [66] introduced fairness metrics to assess device loss and implemented a q-Fair optimization goal in FL. These approaches prioritize active sampling based on system variability, but it's also beneficial to consider sampling a small yet statistically representative group of devices.

3.2.3 Data-specific Challenges

Training federated models becomes particularly challenging when dealing with data that is not evenly distributed across devices. This poses issues both in modeling the data and analyzing the convergence of training methods. Traditional ML typically assumes that data is independently and identically distributed (IID). This assumption works well in scenarios where data is centrally collected and then distributed for training. However, in federated settings, data comes from a variety of devices or institutions and often does not adhere to the IID assumption. This disparity makes training a single global model on the combined datasets of all clients more complex with non-IID data. To address this issue, common solutions involve focusing on the global model and modifying the local training approach, such as adjusting hyperparameters, or incorporating additional steps into the data preprocessing phase.

The main categories of non-IID data include i) label distribution skew, where the distribution of labels is different across different nodes, ii) feature distribution skew, where the distribution of features is different across different nodes, iii) the same label but different features, where the same label is used for different features across different nodes, iv) same features but different label, where the same features are used for different labels across different nodes and v) quantity skew where the amount of data available at local nodes is different [7]. The case 'same label but different features', known as Concept drift, is mainly related to vertical FL where the clients share overlapped sample IDs with different features.

The FedAvg algorithm initially addressed the issue of non-identically distributed data across devices by averaging local updates on each device. Mohri *et al.* [67] further improved this by enhancing the global model to better accommodate a mix of client distributions, addressing the often-overlooked aspect of fairness that could lead to a biased centralized model. On the aggregation front, Wang, X. *et al.* [68] explored the convergence challenges in FL, particularly in non-IID data environments. They proposed an adaptive method to minimize the loss function while considering resource constraints, offering insights into various convergence scenarios for FedAvg under non-IID conditions. To better understand FedAvg's performance in statistically heterogeneous settings, FedProx [69] was developed. It modifies FedAvg slightly to ensure both theoretical and practical convergence, considering system heterogeneity across devices. Regarding data preprocessing, Huang, Shea *et al.* [70] introduced a community-based FL method using clustering. This approach segregates data into different clusters for federated training, addressing non-IID challenges but is less viable for large-scale data due to high parameter conversion overhead.

For some applications, data augmentation strategies can be employed to make client data more uniform. One method is to create and globally share a small dataset, which could be sourced from publicly available data, a non-sensitive subset of client data, or a condensed version of the raw data. Approaches like MOCHA [6], have leveraged multi-task learning to utilize shared representations, personalizing models for individual devices. Similarly, some researchers have proposed sharing a small data subset among local models to address non-IID data challenges [71]. Building on this concept, Huang, Yin *et al.* [72] incorporated cross-entropy loss into the transmission process and varied local update times for clients in each training round to mitigate the Non-IID issue.

3.2.4 Security Challenges

FL systems are designed to improve data privacy since the gradient information is shared between the FL participants, while the transmission of raw data is not required. However, research studies reveal that FL does not guarantee adequate privacy and security to the system. Research has shown that even a small fraction of gradients can expose sensitive information about local data in FL, and sharing model updates during training can also pose a risk to privacy [73, 74]. For instance, there is evidence about FL attacks that are able to retrieve speaker information from the transferred gradients when training FL systems for automatic speech recognition applications [75]. To mitigate these threats and ensure the privacy and security of FL, it is important to investigate potential attacks on the FL network and develop defence mechanisms. Some relevant studies provide useful strategies for protecting FL against such attacks. During the training phase of FL networks, attacks that occur are known as poisoning attacks. These attacks can impact either the dataset or the local model, with the goal of modifying the behavior of the FL network in an undesirable way and distorting the global ML model's accuracy and performance. As is already stated, attackers can exploit the communication protocol amongst different participants to perform malicious actions, which can target either the data of each client (Data Poisoning attack) or the shared model parameters (Model Poisoning attack). Data poisoning attacks compromise the integrity of training data, while model poisoning attacks target partial or full model replacement during training. In an FL system, attacks can be performed by either the central server or the participating clients of the FL system. Figure 3.1 shows that data poisoning is performed at local data collection, while model poisoning is sourced at the local model training process.

In the case of data poisoning attacks, the attacker aims to degrade the performance of a set of target nodes by injecting corrupted/poisoned data into nodes. Label flipping is a data poisoning attack in which malicious nodes change the labels of samples either arbitrarily or with a specific pattern. In the first case, a different label is randomly assigned to a sample in such a way that the global performance of the FL model is reduced. In the second case, the malicious node assigns specific labels to a set of records with a clear purpose in mind. Due to the vast number of participants involved in the FL system, it is not guaranteed who is an honest as well as a credible entity and who is a malicious actor.

Several survey works have been published in the last past years with the aim to review and summarise the latest papers related to adversarial attacks and threats on FL systems as well as the possible defense mechanisms. The authors in [76] present an extensive review

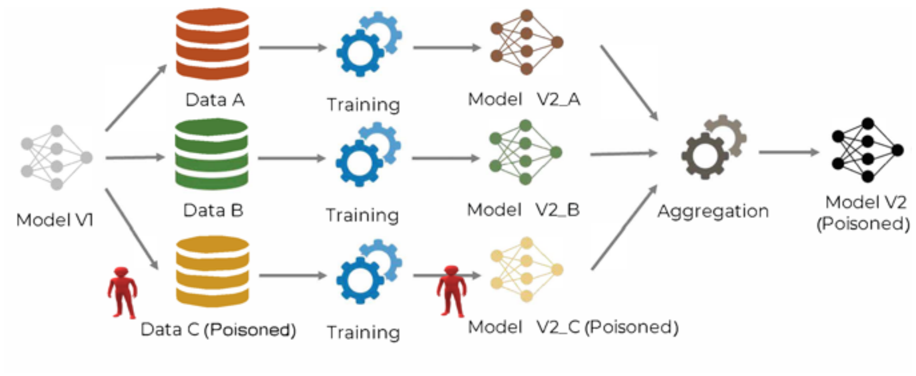


Figure 3.1: Data vs. Model poisoning attacks on FL system. In the first case, the data of the third node has been compromised in such a way as to affect the training process. In the second case, the attacker is trying to interfere with the main training process.

of the various threats that can be applied to an FL system as well as their corresponding countermeasures. Specifically, they provide several taxonomies of adversarial attacks and their defence methods, depicting a general picture of the vulnerabilities of FL and how to overcome them. The threats that can introduce vulnerabilities to trustworthy FL systems, across different stages of the development procedure are introduced in [77]. This work analyses the attacks that can be performed by a malicious participant in FL during data processing, model training, deployment and inference. Additionally, the authors of this paper aim to assist in the selection of the most appropriate defence mechanism by discussing specific technical solutions to realize the various aspects of trustworthy FL. The work [78] provides a review of the concept of FL, threat models and two major attacks, namely poisoning attacks, and inference attacks, by highlighting the intuitions, key techniques and fundamental assumptions of the attacks. Some years after, a more comprehensive survey on privacy and robustness in FL is conducted and presented in [79]. The authors of this work review the various threat models, privacy attacks and poisoning attacks as well as their corresponding defences. Finally, the paper [8] demonstrates a comprehensive study regarding the security and privacy aspects that need to be considered in an FL setup. The results obtained from this research indicate that communication bottlenecks, poisoning and backdoor attacks are the most specific security threats, while inference-based attacks are the most crucial to FL privacy.

Chapter 4

Study 1: Federated Learning in IID and Non-IID Settings

4.1 Overview of the Study

The preliminary phase of the study was dedicated to validating the practical application of FL in two distinct domains: Computer Vision, specifically in classifying faces based on skin color, and text categorization, focusing on named entity recognition. This phase was underpinned by an exhaustive literature review, complemented by an in-depth exploration of FL methodologies. Through this process, we conducted a series of targeted, small-scale experiments, each designed to delve into the specific demands and potentialities of FL. These experiments were meticulously structured with several objectives: a) to validate the sufficiency and relevance of the experimental results, b) to ascertain the accuracy and applicability of the selected workflows, c) to identify and understand any inherent limitations in addressing complex problems within the FL framework, d) to verify the practical viability of the proposed solutions and their congruence with predefined expectations, and e) to accumulate insightful experiences that could inform future research and application in this field. Additionally, one of the major objectives of these experiments was to examine the impact of both IID (Independently and Identically Distributed) and non-IID settings in FL. This aspect is crucial, as the performance of FL can vary significantly based on the nature of data distribution across different nodes. Understanding how FL models perform under these varying conditions was pivotal to assessing their robustness and adaptability in real-world scenarios.

During the experimental phase, two primary challenges were tackled: the first involved the development of a system for facial classification based on skin color, and the second focused on recognizing named entities in textual data. For each challenge, appropriate datasets were meticulously selected and subjected to thorough pre-processing to ensure optimal suitability for FL. The development stage was a crucial part of the process, involving the creation, rigorous testing, and fine-tuning of the initial models. This stage also included experimenting with various FL architectures to identify the most effective solutions. We employed a diverse range of aggregate algorithms across different nodes, evaluating their performance relative to traditional centralized training models.

The study was structured into three distinct experimental evaluations, each tailored to elucidate FL's capabilities under varied data heterogeneity scenarios within a federated setting. The approach was methodically designed to progress from simpler to more complex data distributions:

a) **Single Node with Limited Data:** The initial evaluation focused on a single node handling a constrained dataset. This setup provided baseline insights into FL's performance in a minimally distributed environment.

b) **Federated Nodes with Even Data Distribution:** The second phase expanded to incorporate federated nodes, with an emphasis on achieving an even distribution of data between them. This scenario was key in understanding FL's behavior in a relatively simple federated system.

c) **Federated Nodes with High Data Heterogeneity:** The third and most complex evaluation involved federated nodes, each with significantly less and more unevenly distributed data compared to a centralized approach. This stage was crucial for examining FL's efficiency and adaptability in handling high data heterogeneity, a common challenge in real-world applications.

This comprehensive approach not only allowed us to assess the effectiveness of different aggregation algorithms in FL settings but also provided invaluable insights into the adaptability and scalability of FL systems under varying levels of data distribution complexity. In the final phase of the study, we drew preliminary conclusions, thoroughly evaluated the outcomes from each experimental setup, and developed strategies for future advancements. This comprehensive analysis paves the way for more sophisticated applications of FL in Computer Vision and text categorization, particularly in scenarios characterized by diverse and uneven data distributions.

4.2 Methodology

In FL, optimization strategies, particularly aggregation algorithms, are crucial as they merge insights from various devices or nodes while safeguarding users' or organizations' private data. The cornerstone of this process is the Federated Averaging algorithm, known as "FedAvg," being a foundational algorithm for distributed training. Since its inception, numerous adaptations of FedAvg, like "FedProx" and "FedOpt," have been introduced to tackle the primary challenges in this field.

Delving into the specifics of the FL process, many optimization techniques rely on local client updates. Here, clients perform multiple model updates locally before communicating with the central server. This approach significantly cuts down on the communication required for model training. In this first study, three principal aggregation mechanisms were employed and meticulously examined within specific scenarios. These include the original FedAvg, along with its evolved variants FedProx and FedOpt. Each of these mechanisms was applied and analyzed to understand their effectiveness and suitability in different FL contexts.

Optimization parameters: After selecting the network topology for the nodes, various parameters of the FL process can be adjusted to enhance the learning outcomes. These parameters include: a) the number of FL rounds, b) the total count of nodes involved in the process, c) the proportion of nodes engaged in each iteration, d) the batch size for each local learning iteration, e) the number of local training iterations prior to aggregation, and f) the local learning rate. The optimization of these parameters should be tailored to fit the specific requirements and constraints of the ML application, such as the available computational resources, memory capacity, and network bandwidth.

FL operates as a protocol combining communication and training, as delineated in Algorithm 1 (outlined also in [80]). This protocol involves a network comprising several devices, referred to as clients, and a server that orchestrates the learning process. Each client possesses a local dataset that remains on the device and is never uploaded to the server. The primary objective is to develop a global model by integrating the outcomes of training performed locally by these clients. Algorithm 1 describes a training process in a FL system involving a predefined set of clients, labeled $I = \{1, \ldots, N\}$, where each client has its own local dataset. At the onset of each communication round, indexed by $t \in \{0, E, \ldots, (T-1)E\}$, the server randomly selects a subset of clients, I_t, comprising $C \cdot N$ participants. Key parameters set by the server include the client set I, the client selection ratio C per round, the total number of communication rounds T, and the number of local epochs E. In each round, the server distributes the current global algorithm state to the clients, instructing them to perform local computations based on this global state and their individual datasets. Following this, clients transmit their updates back to the server, which then updates the global model by aggregating these client contributions. This cycle continues until the training process is completed.

4.2.1 Federated stochastic gradient descent (FedSGD)

DL training mainly relies on variants of stochastic gradient descent (SGD). This method involves calculating gradients on a randomly chosen subset of the entire dataset, which are then applied to progress one step in the gradient descent process. SGD simplifies the traditional gradient descent algorithm by computing the gradient on a very small segment (known as a mini-batch) of the full dataset. In its most extreme stochastic form, a single data sample is randomly selected for each optimization step. During training the back-propagation algorithm updates each parameter w by calculating the gradient of F. For each data point i, $F_i(w)$ represents the loss, where F is the loss function. SGD updates parameters using the formula:

$$w := w - \eta \nabla F_i(w) \tag{4.1}$$

Algorithm 1 Generic Federated Learning Algorithm

```
Require: $K, T, E, \eta$    ▷ $K$: number of clients, $T$: total number of rounds, $E$: number of local epochs, $\eta$: learning rate
Ensure: $w^{global}$    ▷ Global model parameters
 1: Initialize $w_0^{global}$
 2: for $t = 1, 2, \dots, T$ do    ▷ Global training rounds
 3:     $m \leftarrow \max(\lceil K \cdot C \rceil, 1)$    ▷ Number of selected clients per round
 4:     $S_t \leftarrow$ (random set of $m$ clients)
 5:     for each client $k \in S_t$ in parallel do
 6:         $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t^{global}, E, \eta)$    ▷ Local model update
 7:     end for
 8:     $w_{t+1}^{global} \leftarrow \sum_{k=1}^m \frac{n_k}{n} w_{t+1}^k$    ▷ Aggregate the local models
 9: end for
10: return $w_T^{global}$
```

where η is the learning rate. An epoch is a full cycle through the dataset. Federated SGD (FedSGD) extends SGD to federated settings, selecting a random fraction of nodes and using their data for gradient calculations. In Algorithm 2 the server averages these gradients, weighted by the data volume of each node. Selective SGD further refines this by updating parameters with the largest gradients, based on a set threshold θ.

Algorithm 2 FedSGD Algorithm

```
Require: $N, C, T, E, \eta$    ▷ N: total number of clients, C: fraction of clients, T: global epochs, E: local epochs, $\eta$: learning rate
Ensure: $w_T$    ▷ $w_T$: model parameters after T global epochs
 1: Initialize $w_0$    ▷ Initial model parameters
 2: for $t = 1, 2, \dots, T$ do    ▷ Global training epochs
 3:     $m \leftarrow \max(C \cdot N, 1)$    ▷ Number of selected clients
 4:     $S_t \leftarrow$ (random set of $m$ clients)
 5:     for each client $k \in S_t$ in parallel do
 6:         $w_{t+1}^k \leftarrow w_t - \eta \nabla L_k(w_t)$    ▷ Local SGD update
 7:     end for
 8:     $w_{t+1} \leftarrow \frac{1}{m} \sum_{k=1}^m w_{t+1}^k$    ▷ Aggregate local models
 9: end for
10: return $w_T$
```

4.2.2 Federated Averaging (FedAvg)

Federated Averaging (FedAvg) extends FedSGD by allowing local nodes to perform multiple batch updates on their data and exchange updated weights instead of gradients. This approach is based on the idea that if all nodes begin with the same initialization, averaging gradients is effectively the same as averaging weights. Moreover, averaging weights from the same starting point does not significantly impact the performance of the averaged model. FedAvg is a more communication-efficient version of FedSGD and is the most popular optimization algorithm in FL. Unlike traditional SGD, which aggregates updates from different clients immediately after each local step, FedAvg optimizes the local model using local data and aggregates updates at a centralized device only after every $s-th$ local step, where $s \geq 1$ is typically a small number.

As depicted in Algorithm 3 FedAvg operates by initially running several epochs of stochastic gradient descent (SGD) on a subset of devices (K) in the network. These devices then send their model updates to a central server for averaging. The goal in each FedAvg round is to minimize the global model objective, which is typically a sum of the weighted average of local loss functions computed on each client. The objective function can be represented as follows:

$$\text{minimize}_w \quad F(w) = \sum_{k=1}^{K} p_k F_k(w) \tag{4.2}$$

Algorithm 3 Federated Averaging (FedAvg) Algorithm

```
Require: $K, T, E, \eta$    ▷ $K$: number of clients, $T$: number of communication rounds, $E$:
    number of local epochs, $\eta$: learning rate
Ensure: $w^{global}$    ▷ Global model parameters
 1: Initialize $w_0^{global}$
 2: for $t = 1, 2, \ldots, T$ do
 3:     $m \leftarrow \max(\lceil K \cdot C \rceil, 1)$    ▷ Number of selected clients per round
 4:     $S_t \leftarrow$ (random set of $m$ clients)
 5:     for each client $k \in S_t$ in parallel do
 6:         $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t^{global})$
 7:     end for
 8:     $w_{t+1}^{global} \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} w_{t+1}^k$    ▷ Aggregate the local models
 9: end for
10: return $w_T^{global}$
```

Here, K represents the total number of devices, p_k is a user-defined weight for each device, and F_k is the local objective function for the $k-th$ device. The round begins with a random selection of a subset of K clients. The server shares its global model w with these clients, which independently run SGD on their loss function F_k and return their updated model w_k to the server for aggregation. The server updates its global model by averaging these local models. This process repeats across n communication rounds. The number of local epochs in FedAvg is crucial for convergence. More local epochs mean more local computations and potentially less frequent communication, speeding up convergence in networks with communication constraints. However, in cases where local objectives F_k are heterogeneous, too many local epochs can cause devices to converge to their local optima rather than the global objective, potentially leading to divergence or slower convergence. A high number of local epochs also increases the risk of devices failing to complete training in a given round, necessitating their dropout.

Despite its simplicity and wide applicability, FedAvg, as a fundamental federated optimization algorithm, must be carefully managed to balance local computation, communication efficiency, and convergence to the global objective.

Algorithm 4 ClientUpdate (FedAvg) Algorithm

```
1: function CLIENTUPDATE($k, w$)
2:     $w_k \leftarrow w$
3:     for $e = 1, 2, \dots, E$ do
4:         Batch $\mathcal{B} \leftarrow$ (split client $k$'s data into batches)
5:         for each batch $b \in \mathcal{B}$ do
6:             $w_k \leftarrow w_k - \eta \nabla L(w_k; b)$                ▷ Local SGD update
7:         end for
8:     end for
9:     return $w_k$
10: end function
```

4.2.3 A federated optimization algorithm - FedProx

FedProx is an advanced version of FedAvg, designed to better handle the diverse capacities of devices in FL. It differs from FedAvg by allowing variable amounts of local work based on each device's system resources, thus adapting to network heterogeneity and reducing communication overhead. The key innovation in FedProx is the introduction of a proximal term in the local optimization objective, which ensures that local updates stay closer to the global model. This feature effectively addresses statistical heterogeneity and eliminates the need for setting a fixed number of local epochs. Importantly, FedProx includes partial solutions for slower devices, preventing their exclusion and promoting a more inclusive learning process. Its lightweight modifications to FedAvg facilitate easy integration into existing frameworks like NVIDIA Flare, with FedAvg being a special case of FedProx when the proximal term coefficient μ is zero. This adaptability makes FedProx a versatile and effective solution for the varying demands of FL environments. The objective function of FedProx, which extends the objective of FedAvg by adding a proximal term, aims to solve the following optimization problem:

$$\text{minimize}_w \quad F(w) = \sum_{k=1}^{K} p_k F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 \tag{4.3}$$

This proximal regularization term penalizes the squared Euclidean distance between the local updates and the global model parameters from the previous round, weighted by a factor μ, as depicted in Algorithm 7.

Algorithm 5 FedProx Algorithm

```
Require: $K, T, E, \eta, \mu$  ▷ $K$: number of clients, $T$: number of communication rounds, $E$: number of local epochs, $\eta$: learning rate, $\mu$: proximal term coefficient
Ensure: $w^{global}$                                  ▷ Global model parameters
 1: Initialize $w_0^{global}$
 2: for $t = 1, 2, \ldots, T$ do
 3:     $m \leftarrow \max(\lceil K \cdot C \rceil, 1)$            ▷ Number of selected clients per round
 4:     $S_t \leftarrow$ (random set of $m$ clients)
 5:     for each client $k \in S_t$ in parallel do
 6:         $w_{t+1}^k \leftarrow$ ClientUpdate($k, w_t^{global}, \mu$)
 7:     end for
 8:     $w_{t+1}^{global} \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} w_{t+1}^k$          ▷ Aggregate the local models
 9: end for
10: return $w_T^{global}$
```

Algorithm 6 ClientUpdate (FedProx) Algorithm

```
 1: function ClientUpdate($k, w, \mu$)
 2:     $w_k \leftarrow w$
 3:     for $e = 1, 2, \ldots, E$ do
 4:         Batch $\mathcal{B} \leftarrow$ (split client $k$'s data into batches)
 5:         for each batch $b \in \mathcal{B}$ do
 6:             $w_k \leftarrow w_k - \eta(\nabla L(w_k; b) + \mu(w_k - w))$  ▷ Local SGD update with proximal term
 7:         end for
 8:     end for
 9:     return $w_k$
10: end function
```

4.2.4 Federated Optimisation (FedOpt)

FedOpt represents a significant evolution in FL optimization techniques compared to its predecessors like FedAvg, FedProx, and FedAvg's variants. Its primary distinction lies in the flexibility of employing a diverse range of gradient-based optimizers for both client-side and server-side (ServerOptimizer) computations. Unlike earlier algorithms that predominantly use SGD, FedOpt incorporates adaptive optimizers like ADAM, YOGI, and ADAGRAD. This adaptability allows for more nuanced, client-specific optimization, leveraging globally aggregated statistics for more informed updates. Additionally, FedOpt introduces the concept of varying learning rates across different rounds, accommodating dynamic learning rate schedules. Its integration into advanced frameworks like NVIDIA FLARE highlights its ability to maintain communication efficiency while enhancing convergence capabilities, a notable advancement over the more rigid structures of prior algorithms. The global objective function in FedOpt can be represented as:

$$\text{minimize}_w \quad F(w) = \sum_{k=1}^{K} p_k F_k(w) \tag{4.4}$$

The objective function of the FedOpt algorithm, as illustrated in Algorithm 6, similar to other FL algorithms, generally aims to minimize a global loss function that is a weighted sum of local loss functions. However, the key distinction of FedOpt lies in the optimization strategy, particularly on the server side, where more sophisticated algorithms (like Adam, SGD with momentum, etc.) are used instead of simple averaging.

Algorithm 7 FedOpt Algorithm

```
Require: $K, T, E, \eta$, ServerOptimizer    ▷ $K$: number of clients, $T$: number of communication rounds, $E$: number of local epochs, $\eta$: learning rate, ServerOptimizer: server-side optimizer
Ensure: $w^{global}$    ▷ Global model parameters
 1: Initialize $w_0^{global}$ and ServerOptimizer
 2: for $t = 1, 2, \ldots, T$ do
 3:     $m \leftarrow \max(\lceil K \cdot C \rceil, 1)$    ▷ Number of selected clients per round
 4:     $S_t \leftarrow$ (random set of $m$ clients)
 5:     Initialize $\Delta w^{global} \leftarrow 0$    ▷ Global model update
 6:     for each client $k \in S_t$ in parallel do
 7:         $\Delta w_k \leftarrow$ ClientUpdate($k, w_t^{global}, E, \eta$)
 8:         $\Delta w^{global} \leftarrow \Delta w^{global} + p_k \cdot \Delta w_k$    ▷ Weighted sum of client updates
 9:     end for
10:     $w_{t+1}^{global} \leftarrow$ ServerOptimizer($w_t^{global}, \Delta w^{global}$)    ▷ Server optimization step
11: end for
12: return $w_T^{global}$
```

Algorithm 8 ClientUpdate (FedOpt) Algorithm

```
 1: function ClientUpdate($k, w, E, \eta$)
 2:     $w_k \leftarrow w$
 3:     for $e = 1, 2, \ldots, E$ do
 4:         Batch $\mathcal{B} \leftarrow$ (split client $k$'s data into batches)
 5:         for each batch $b \in \mathcal{B}$ do
 6:             $w_k \leftarrow w_k - \eta \nabla L(w_k; b)$            ▷ Local SGD update
 7:         end for
 8:     end for
 9:     return $w_k - w$                                        ▷ Return model update
10: end function
```

4.3 Experimental Setup and Data Description

This section delves into a technical analysis of the FL framework, specifically within the NVFLARE ecosystem, detailing the performance and outcomes of models trained under various scenarios. It encompasses a series of experiments focused on image classification and named entity recognition, aiming to explore the influence of FL on the entire training and evaluation process. These experiments shed light on how various learning algorithms and alterations in data distribution affect the overall performance of the global model.

4.3.1 Experiments details

Image Classification

The image classification task in this study involves categorizing each input image into one of several predefined categories. We utilized a DL model trained and evaluated on the CIFAR-10 dataset for both FL and centralized learning approaches. CIFAR-10, a popular dataset in computer vision, comprises 60,000 low-resolution (32x32) RGB images across 10 varied classes (illustrated in Figure 4.1). The choice of CIFAR-10 for our experiments is driven by its manageable image resolution, facilitating multiple FL scenarios to analyze the effectiveness of different learning strategies.

For the DL model, we employed a Convolutional Neural Network (CNN) with a structure comprising three blocks of convolutional layers. Each block contains two 3x3 convolutional layers and a max-pooling layer, followed by three fully connected layers. While this model doesn't represent the cutting-edge for the CIFAR-10 dataset, it effectively illustrates performance variations essential for our study. The primary metric for evaluating the model's effectiveness is classification accuracy.

In the centralized training setup for this preliminary study, the entire CIFAR-10 dataset is consolidated at a single node. This setup utilizes $50,000$ images for training and the remaining 10,000 for testing. The performance of the centralized training model serves as a benchmark for comparing against various FL algorithms. Regarding the FL

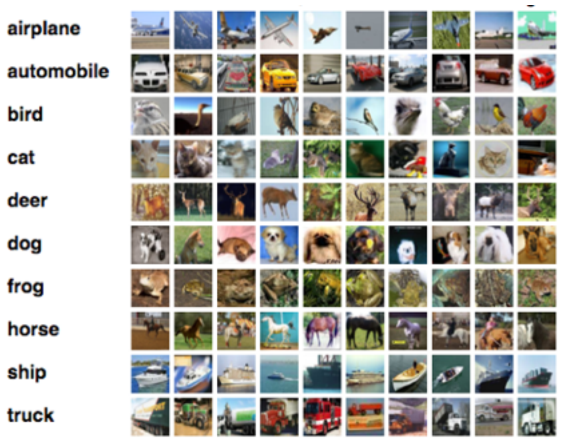


Figure 4.1: Benchmark dataset: CIFAR-10 dataset (an example of an IID type). Source: https://www.cs.toronto.edu/ kriz/cifar.html.

experiments, we initially focus on analyzing diverse learning strategies from the previous section under the assumption of evenly distributed data among participants. Subsequently, we explore the impact of non-uniform data distribution on classification performance, emphasizing how different aggregation methods can enhance outcomes in scenarios with less homogeneous data. It's important to note that CIFAR-10 was originally designed for traditional centralized learning algorithms and is not inherently suited for FL. To adapt CIFAR-10 for FL, we implement dataset partitioning strategies that mimic realistic data distributions (as depicted in Figures 4.2 and 4.3). Following Wang's approach, we employ a Dirichlet sampling algorithm where the heterogeneity level in terms of data size and class distribution is governed by a parameter α. For consistent comparisons across different setups, we use the original CIFAR-10 test set, comprising 10,000 images, as the universal test set.

To prepare a non-IID/real-world dataset for image classification tasks, one can adopt the following procedure (also depicted in Figure 4.4):

1. Hyper-parameter Configuration: Define key parameters like the number of classes per client, total number of clients, and batch size.

2. Dataset Preparation: Start with downloading or acquiring the original dataset. Split this dataset for training and testing, typically using an 80/20 or 70/30 split.

   (a) Data Transformation: Implement data transformations to create a random distribution for clients, ensuring each client has a varied number of images.

   (b) Client-wise Image Distribution: Distribute the images among 'n' clients, creating a split that mimics real-world dataset characteristics.

   (c) Non-IID Dataset Split: Establish a similar split for the non-IID dataset.

   (d) Data Shuffling: Shuffle the images within each client's dataset to ensure randomness.

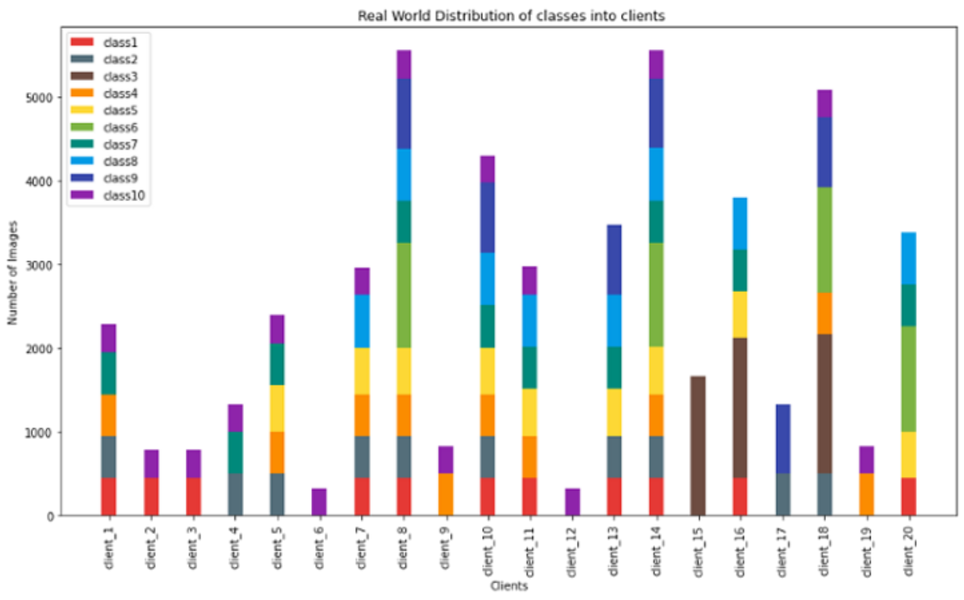


Figure 4.2: CIFAR-10 real-world data partition among clients.

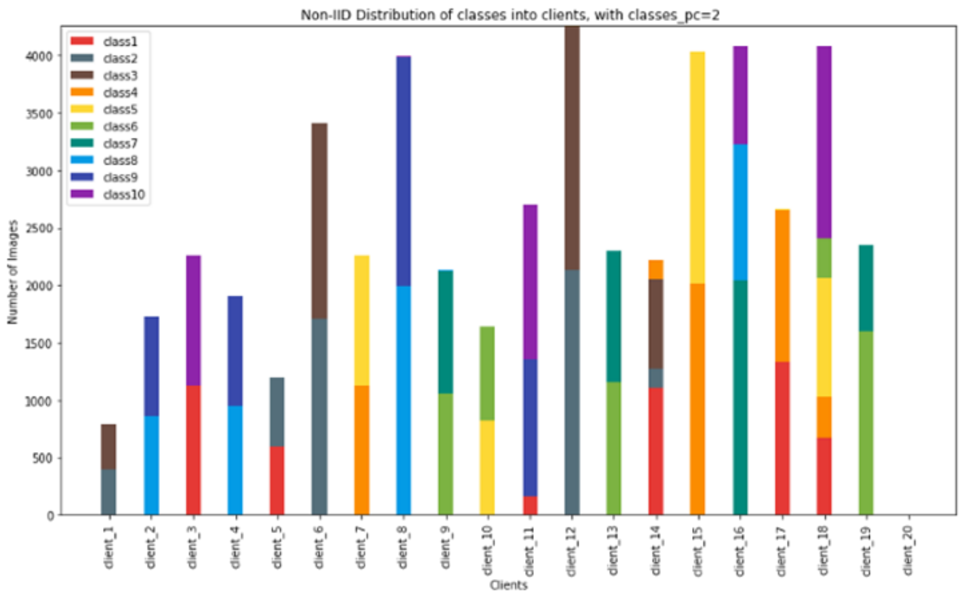


Figure 4.3: An extreme highly heterogeneous dataset partition.

(e) Data Loading: Convert the split into a data loader, incorporating image augmentation, to serve as input for model training.

(f) FL Training: Utilize these datasets in FL to develop advanced models.

(g) Baseline Retraining: Reserve a set of images on the global server for retraining clients' models before aggregation. This retraining approach addresses the challenges posed by non-IID/real-world datasets, ensuring more uniform model training.

This comprehensive method ensures that the datasets are effectively prepared for FL, simulating realistic conditions and variations in data distribution.

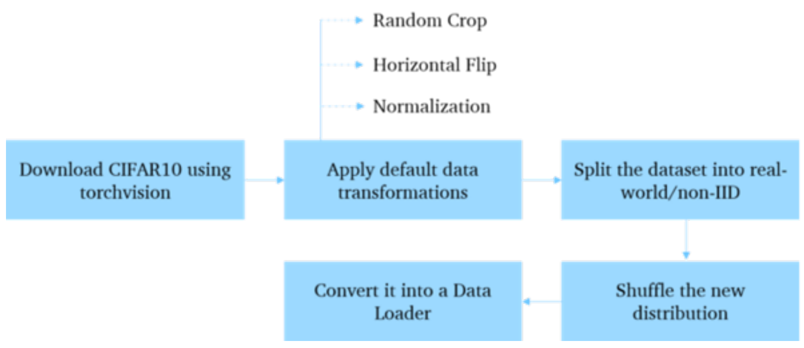

Random Crop
Horizontal Flip
Normalization
Download CIFAR10 using torchvision
Apply default data transformations
Split the dataset into real-world/non-IID
Shuffle the new distribution
Convert it into a Data Loader

Figure 4.4: Data flow diagram.

Name entity recognition

Named Entity Recognition (NER), a subtask of information extraction, focuses on identifying and classifying entities in unstructured text into predefined categories. For this task, the DL model is trained on the CoNLL-2003 dataset, a prominent open-source dataset specifically designed for NER. This dataset comprises 20,745 sentences with 301,415 labeled tokens distributed across 9 classes. It's divided into a training set with 20,345 sentences and a test set with 400 sentences. The NER model employs a Bidirectional Long Short-Term Memory (LSTM) architecture. It starts with an embedding layer, followed by a bidirectional LSTM layer, and a fully connected layer processes the LSTM's output. The model has a total of $236,809$ trainable parameters. Adam stochastic gradient descent is used as the optimizer, along with a sparse categorical cross-entropy loss function. Model performance in the FL setup is evaluated using the F1 Score. Both FL and centralized learning approaches are used for training and experimentation. In the centralized approach, the entire training dataset is consolidated on a single node. The performance of this centrally trained NER model serves as a baseline for comparing subsequent experimental results. The first FL experiment involves training the NER model with 10 clients, each receiving an equal portion of the training dataset, thus creating a homogeneous data distribution. The second experiment introduces a heterogeneous data split, distributing the training dataset unevenly among the 10 clients.

4.4 Results and Analysis

4.4.1 Image Classification

Building on the CIFAR-10 dataset preparation discussed earlier, we examine the performance of the learning algorithms described in Section 4.2 across various data distribution scenarios, ranging from identical to non-identical. To streamline the process, these experiments are carried out using NVFLARE's virtual nodes on a single machine, which effectively reduces the time required for model transmission and overall training duration.

Centralized Learning

The CIFAR-10 dataset is located on a single node, where the model undergoes training across 25 epochs. Figure 4.5 displays the classification accuracy achieved by this centralized training approach, recording an accuracy of 0.8275.

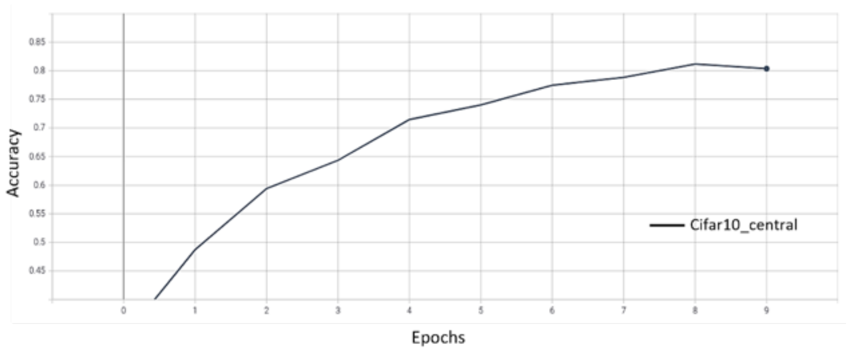


Figure 4.5: Centralised learning performance on CIFAR-10.

FedAvg Learning with non-identical participants

This experiment aims to compare the performance of the global model between centralized and distributed training setups, using FedAvg, the most basic aggregation method, under varying degrees of data heterogeneity. The heterogeneity is controlled by the parameter α, with larger α values indicating more uniform data distribution among participants, and smaller α values leading to each participant having data predominantly from one class. In our setup, there are 10 participants, and both the local epochs and communication/aggregation rounds are set to 10. This ensures a comparable number of iterations across participants as seen in the centralized baseline. We test the model with four different α values $[1, 0.5, 0.3, 0.1]$ to create a range of data distributions from identical to highly non-identical. Figure 4.6 illustrates how classification performance varies with the Dirichlet parameter α. It shows that test accuracy remains close to the centralized baseline for $\alpha = 1$ but declines significantly for smaller α values, where participants have more varied data distributions. This outcome suggests that the FedAvg algorithm struggles with datasets that are non-identically distributed and highly heterogeneous.

Advanced Learning Strategies with non-identical participants

The experiments we conducted demonstrate that with increasingly heterogeneous data, more sophisticated FL algorithms are needed. To explore this, we ran experiments

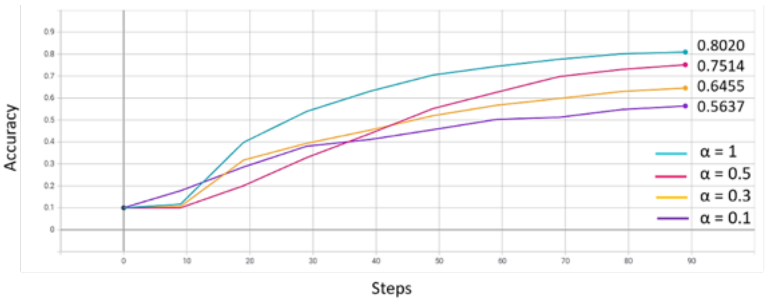


Figure 4.6: FedAvg learning curves for different α.

using the FedProx and FedOpt learning strategies, focusing on their effectiveness in enhancing the global model's performance and convergence in scenarios of high data distribution heterogeneity. For these experiments, we set the heterogeneity parameter α to 0.1, keeping all other training parameters consistent with the FedAvg experiment. As depicted in Figure4.7, FedProx matches FedAvg in terms of performance. However, FedOpt significantly surpasses FedAvg, achieving better results under the same α value and number of training iterations. FedOpt accomplishes this improvement by employing a Stochastic Gradient Descent algorithm with momentum for the global model update, as detailed in Section 4.2. This adaptation showcases the effectiveness of FedOpt in managing highly heterogeneous data distributions within FL environments.

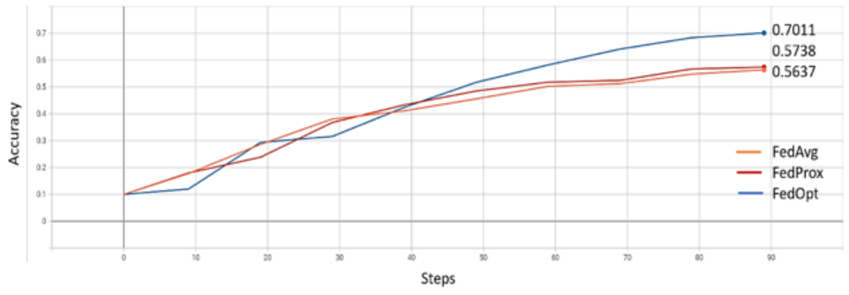


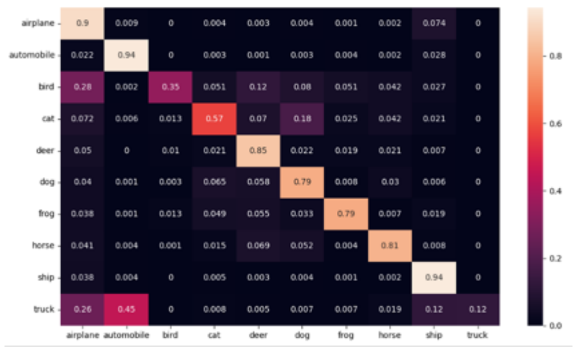
Figure 4.7: FedAvg, FedProx, FedOpt learning curves for $\alpha = 0.1$.

In this study using the FedProx algorithm, we carried out several experiments across a range of α values. The purpose was to underscore the benefits of FL compared to models trained in isolation. Table 4.1 illustrates the dataset distribution with a moderate level of heterogeneity, achieved by setting α to 1 for two randomly selected participants. This distribution results in participants possessing samples across most classes, albeit with a limited number of samples for some. Figure 4.2 displays the confusion matrices for participants 0 and 1. These matrices assess the performance of participant models trained solely with their respective local datasets, as detailed in Table 4.1. The results indicate that both participants attain commendable performance across most classes. However, performance dips are noticeable in classes with fewer samples, such as 'truck' for Participant 0 and 'airplane' for Participant 1, highlighting the impact of sample distribution on model accuracy.

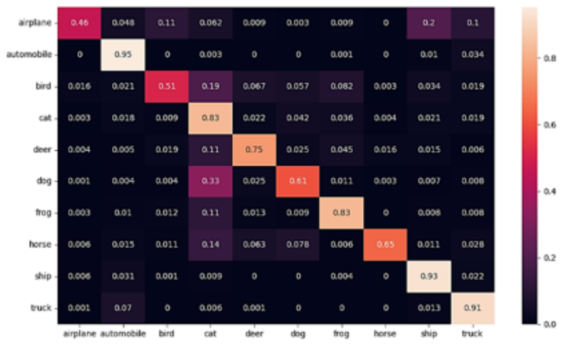
Table 4.1: Closer look at clients' data distribution for medium (a =1) heterogeneity variation.

Class/ Participants	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	1940	433	126	193	840	191	299	243	1284	0
1	22	733	86	786	494	508	876	154	535	209

Table 4.2: Comparative confusion matrices of participants 0 and 1 in a medium-heterogenity scenario.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.9	0.009	0	0.004	0.003	0.004	0.001	0.002	0.074	0
automobile	0.022	0.94	0	0.003	0.001	0.003	0.004	0.002	0.028	0
bird	0.28	0.002	0.35	0.051	0.12	0.08	0.051	0.042	0.027	0
cat	0.072	0.006	0.013	0.57	0.07	0.18	0.025	0.042	0.021	0
deer	0.05	0	0.01	0.021	0.85	0.022	0.019	0.021	0.007	0
dog	0.04	0.001	0.003	0.065	0.058	0.79	0.008	0.03	0.006	0
frog	0.038	0.001	0.013	0.049	0.055	0.033	0.79	0.007	0.019	0
horse	0.041	0.004	0.001	0.015	0.069	0.052	0.004	0.81	0.008	0
ship	0.038	0.004	0	0.005	0.003	0.004	0.001	0.002	0.94	0
truck	0.26	0.45	0	0.008	0.005	0.007	0.007	0.019	0.12	0.12



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.46	0.048	0.11	0.062	0.009	0.003	0.009	0	0.2	0.1
automobile	0	0.95	0	0.003	0	0	0.001	0	0.01	0.034
bird	0.016	0.021	0.51	0.19	0.067	0.057	0.082	0.003	0.034	0.019
cat	0.003	0.018	0.009	0.83	0.022	0.042	0.036	0.004	0.021	0.019
deer	0.004	0.005	0.019	0.11	0.75	0.025	0.045	0.016	0.015	0.006
dog	0.001	0.004	0.004	0.33	0.025	0.61	0.011	0.003	0.007	0.008
frog	0.003	0.01	0.012	0.11	0.013	0.009	0.83	0	0.008	0.008
horse	0.006	0.015	0.011	0.14	0.063	0.078	0.006	0.65	0.011	0.028
ship	0.006	0.031	0.001	0.009	0	0	0.004	0	0.93	0.022
truck	0.001	0.07	0	0.006	0.001	0	0	0	0.013	0.91

Figure 4.8 and Figure 4.9 illustrate the comparative performance of models trained locally versus a model aggregated globally, particularly under conditions of medium data heterogeneity ($\alpha = 1$). While the locally trained models demonstrate high classification accuracy, the globally aggregated model notably outperforms them. This comparison underscores the advantage of FL, where collective learning from multiple sources enhances overall model accuracy beyond what individual local models achieve.

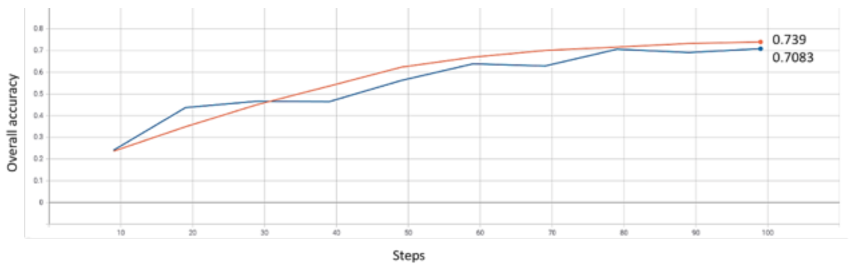


Figure 4.8: Medium-heterogeneity: Local isolated federated nodes learning curve.

Continuing our exploration, we conducted experiments with a higher level of data heterogeneity, setting α to 0.1. Table 4.3 illustrates that under this setting, participants predominantly have a substantial number of samples from one or two classes, while having few or no samples from the other classes. Figure 4.4 reveals that the models trained locally under these conditions exhibit poor performance across most classes. Notably, these models display a tendency toward bias, as they preferentially predict classes that

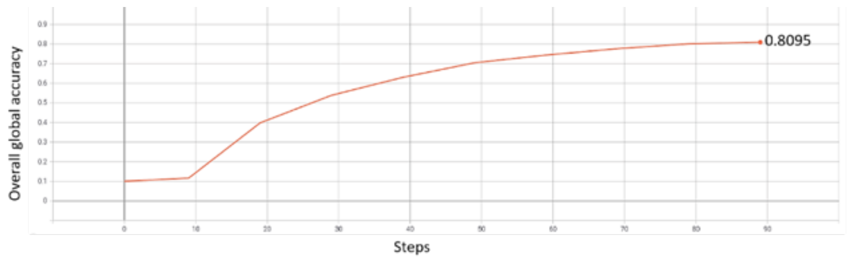


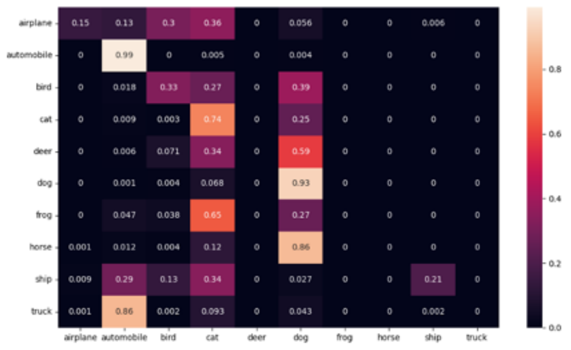
Figure 4.9: Medium-heterogeneity: Global node learning curve.

were more prevalent in their training data. This outcome highlights the challenges and limitations of local training in scenarios with highly skewed data

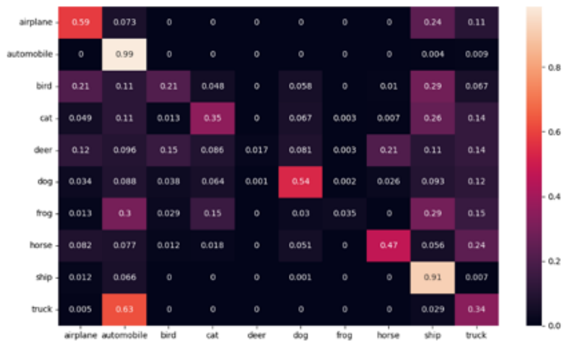
Table 4.3: Closer look at clients' data distribution for high (a =0.1) heterogeneity variation.

Class/Participants	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	0	814	212	3735	0	2737	0	0	0	0
1	46	291	0	4	0	0	0	0	401	123

Table 4.4: Comparative confusion matrices of participants 0 and 1 in a high-heterogeneity scenario.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.15	0.13	0.3	0.36	0	0.056	0	0	0.006	0
automobile	0	0.99	0	0.005	0	0.004	0	0	0	0
bird	0	0.018	0.33	0.27	0	0.39	0	0	0	0
cat	0	0.009	0.003	0.74	0	0.25	0	0	0	0
deer	0	0.006	0.071	0.34	0	0.59	0	0	0	0
dog	0	0.001	0.004	0.068	0	0.93	0	0	0	0
frog	0	0.047	0.038	0.65	0	0.27	0	0	0	0
horse	0.001	0.012	0.004	0.12	0	0.86	0	0	0	0
ship	0.009	0.29	0.13	0.34	0	0.027	0	0	0.21	0
truck	0.001	0.86	0.002	0.093	0	0.043	0	0	0.002	0



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.59	0.073	0	0	0	0	0	0	0.24	0.11
automobile	0	0.99	0	0	0	0	0	0	0.004	0.009
bird	0.21	0.11	0.21	0.048	0	0.058	0	0.01	0.29	0.067
cat	0.049	0.11	0.013	0.35	0	0.067	0.003	0.007	0.26	0.14
deer	0.12	0.096	0.15	0.086	0.017	0.081	0.003	0.21	0.11	0.14
dog	0.034	0.088	0.038	0.064	0.001	0.54	0.002	0.026	0.093	0.12
frog	0.013	0.3	0.029	0.15	0	0.03	0.035	0	0.29	0.15
horse	0.082	0.077	0.012	0.018	0	0.051	0	0.47	0.056	0.24
ship	0.012	0.066	0	0	0	0.001	0	0	0.91	0.007
truck	0.005	0.63	0	0	0	0	0	0	0.029	0.34

Finally, we again compared locally trained models with a globally aggregated model, particularly in a scenario with higher data heterogeneity ($\alpha = 0.1$). Figure 4.10 and Figure 4.11 demonstrate that the locally trained models under this condition have significantly low overall classification accuracy, approximately 30% and 50% respectively. In stark contrast, the globally aggregated model attains a much higher accuracy, nearly 70%. This stark difference in performance clearly underscores the enhanced value provided by the FL paradigm, especially in managing and mitigating the challenges posed by highly skewed or heterogeneous data distributions.

4.4.2 Named entity recognition

This subsection is dedicated to assessing the effectiveness of different Named Entity Recognition (NER) model training approaches in an FL environment, utilizing various data

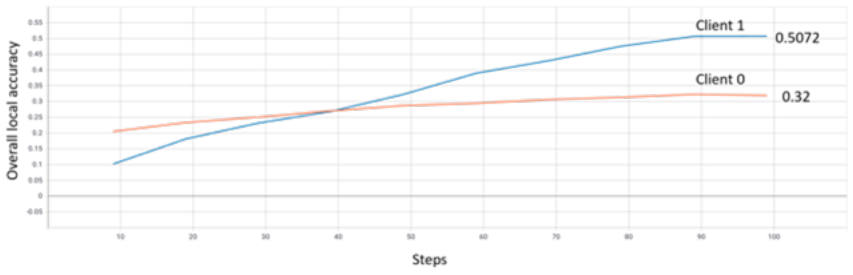


Figure 4.10: High-heterogeneity: Local isolated federated nodes learning curve.

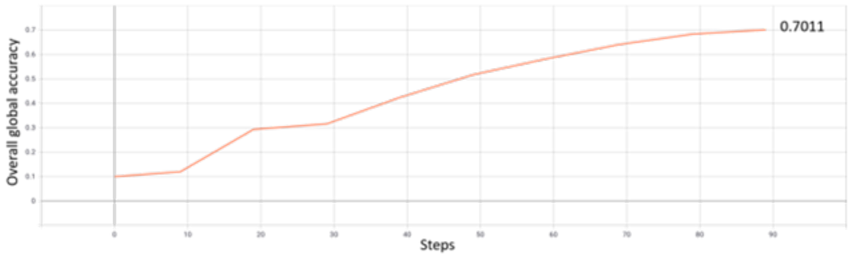


Figure 4.11: High-heterogeneity: Global node learning curve.

partitioning techniques. To optimize the process, these experiments are carried out using NVFLARE's virtual nodes on a single machine, a method that effectively minimizes the time required for model transmission and consequently reduces overall training duration.

Centralized Learning

For the centralized learning experiments, we consolidated the dataset on a single node and trained the NER model over 10 epochs. Figure 4.12 displays the F1 Score achieved by this centralized training approach. The recorded F1 Score for the centralized model reaches 85.57%, which serves as a benchmark for evaluating the performance of models trained under FL setups.

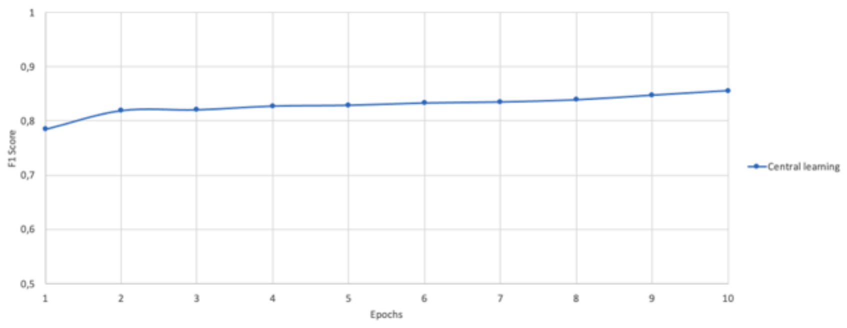


Figure 4.12: Performance of Centralized Learning on Named Entity Recognition Task.

FedAvg Learning with identical and non-identical participants

Initially, for our FL setup, we divided the training dataset evenly among 10 participants, allotting each client 2,034 training samples. Subsequently, we implemented an unequal distribution of the training set among the clients. This unequal split was conducted in three varying degrees of intensity: mild (split #1), moderate (split #2), and intense (split #3). Table 4.5 outlines how the training dataset is allocated to each client under these different splitting scenarios, showcasing the range of data distributions tested in the experiments.

Table 4.5: Number of training samples per client for different data partition strategies.

Client split	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
Equal split	2034	2034	2034	2034	2034	2034	2034	2034	2034	2034
split #1	1999	2078	2077	1996	2034	1996	2067	2004	2118	1976
split #2	1717	2267	2356	1689	2234	2167	1689	2195	2289	1742
split #3	1035	3080	2075	1569	2895	809	898	3997	2459	1528

To provide a clearer picture of how the training sets are distributed among clients, Figure 4.13 is presented. This illustration helps in understanding the variance in dataset allocation across different splitting strategies. In split #1, the number of training samples for each client closely resembles the evenly distributed scenario. In split #2, there's a slight deviation from this uniform distribution. However, split #3 shows a significant departure, with a much more uneven distribution of training samples compared to the original equal splitting. This visual representation in Figure 4.13 effectively highlights the differences in dataset allocation across the three split types.

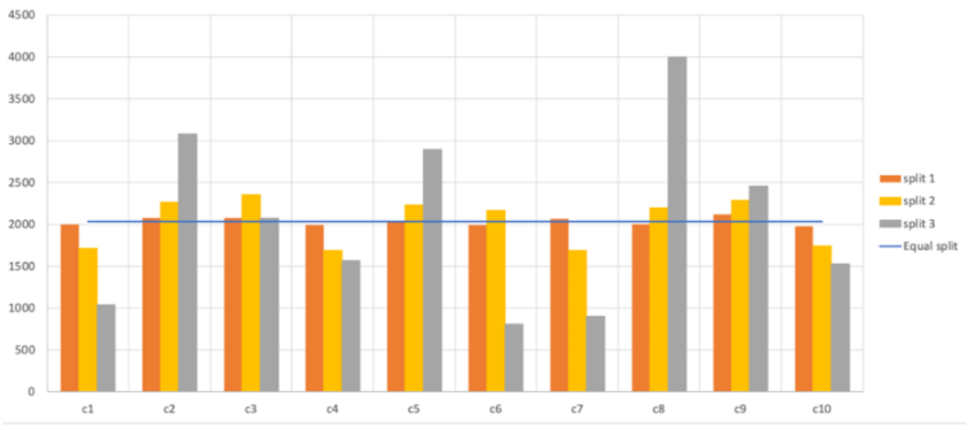


Figure 4.13: Number of training samples per client for different data partition strategies.

Figure 4.14 showcases the F1 Scores obtained from the four distinct experiments focusing on data splitting strategies. Notably, the F1 Score for Split 1 closely aligns with that of the heterogeneous splitting scenario. However, as the data distribution among clients becomes more diverse, the performance of the federated models deteriorates. This pattern indicates that the FedAvg algorithm struggles with highly heterogeneous datasets. This conclusion is consistent with the findings discussed previously, where similar observations were made and illustrated in Figure 4.10. This trend underscores the challenges FedAvg faces in adapting to scenarios with significant data variability across clients.

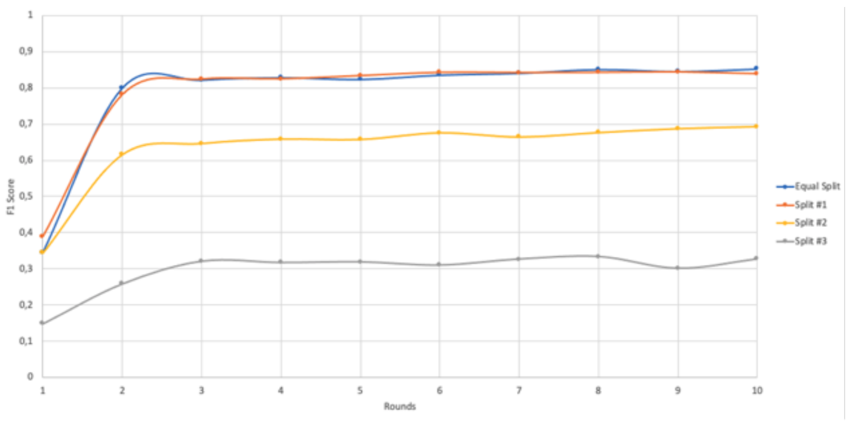


Figure 4.14: FedAvg F1 score curves for different data partition strategies.

Analyzing the performance of the federated model trained on the Split 2 dataset allows us to compare the overall F1 Metric values of the FL model, resulting from Secure Aggregation, with those of individual clients. For instance, Figure 4.15 demonstrates the performance of the global FL model alongside that of Client 1 and Client 7 across each federated round. It's noticeable from this figure that the global FL model consistently outperforms the local models in each round of the federated training. This observation highlights the effectiveness of the federated approach in enhancing model performance through collaborative learning and aggregation, even in scenarios with non-uniform data distribution among clients.

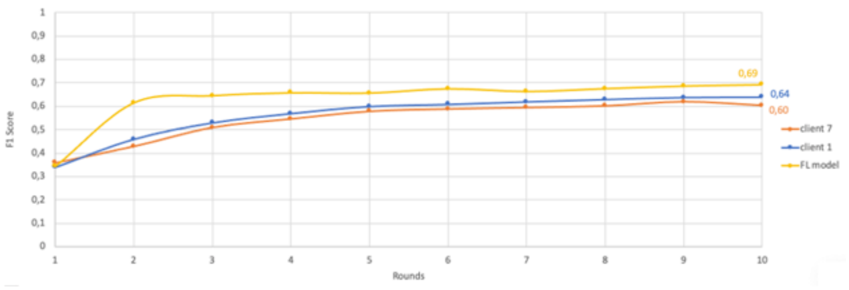


Figure 4.15: Local vs Global performance for two random clients.

4.5 Discussion and Findings

Within this study, we have designed, developed and evaluated a set of FL strategies under different learning scenarios. We have extended the insightful analysis of FL aspects carried out in the previous section, providing several training schemes for the FL framework. We discuss how the FL framework can be applied to various existing datasets successfully followed by a complete evaluation of state-of-the-art aggregation mechanisms. Reflecting on the experimental process and integrating findings from the analysis conducted, the following insights about FL have emerged:

i. Training Time Efficiency: FL significantly accelerates training time, as evidenced by parallel computations across nodes facilitating faster algorithm convergence. This was particularly noticeable in experiments with homogeneous data distribution.

ii. Inference Speed: Using local models for inference, as seen in the NER experiments, leads to quicker inference times compared to centralized approaches.

iii. Robust Privacy Preservation: FL ensures confidentiality through mechanisms like Secure Aggregation, demonstrating a strong commitment to privacy preservation, a key advantage over centralized training.

iv. Easier Collaborative Learning: The distribution of data management and annotation tasks across federation nodes streamlines collaborative learning. This was illustrated in the experiments with different data-splitting strategies, where collaborative learning improved model performance.

v. Accuracy in Varied Data Environments: The accuracy of FL models remains competitive with centralized approaches, even in diverse data environments. This was evident in the CIFAR-10 experiments, where FL models showed resilience in both homogeneous and heterogeneous data distributions.

vi. Handling Data Heterogeneity: The experiments highlighted that simple FL algorithms like FedAvg struggle with highly heterogeneous data, pointing to the need for more sophisticated algorithms in such scenarios.

vii. Trade-offs in Model Performance: FL requires balancing model accuracy with device performance, as seen in experiments where model performance varied with different levels of data heterogeneity.

viii. Advantage in Large, Distributed Datasets: FL is particularly beneficial in situations with large, distributed datasets, as its decentralized nature allows for efficient data handling and processing, surpassing the capabilities of centralized training methods.

In summary, while FL shows immense promise, particularly in privacy preservation and handling distributed datasets, it faces challenges in managing highly heterogeneous data and balancing accuracy with computational efficiency. These findings underscore the need for continued innovation and refinement in FL methodologies.

Chapter 5

Study 2: Addressing Data Modality Heterogeneity in Federated Learning

5.1 Overview of the Study - Introduction to Data Heterogeneity in FL

Following the continuation of my previous work, I completed the research on global literature and the current state in the field of FL, with a focus on areas that attract global research interest such as data and system heterogeneity in image analysis applications, including image classification, object detection, etc. As a result of this research, I shifted my focus to developing new methodologies and tools for investigating the problem of data type heterogeneity in Federated multi-modal 3D Human Action Recognition scenarios. From this research, the following conclusions were drawn:

5.1.1 FL methods

As stated in the previous section, unlike traditional ML approaches that gather distributed local data to a central server, FL solution transfers only the local-trained models, without data exchange, to a centrally located server to build the shared global model. FL relies on an iterative learning procedure. In particular, at each round, every client independently estimates an update to the current NN model, based on the processing of its locally stored data, and communicates this update to a central unit. In this respect, the collected client-side updates are aggregated to compute a new global model. FL inherently ensures privacy and security as the data resides in the owner's premises and is never accessed or processed by other parties.

Addressing heterogeneity

Significant challenges arise when training federated models from data that is not identically distributed across devices, which does not happen in the case of traditional DL techniques, where the data distribution is considered as identically independent. In a naive centralized approach, the generated model could generalise enough and applied to similar devices, since the data used can explain all variations in the devices and their environment. However, in reality, data is collected in an uneven fashion and certainly do not follow an Identically Independent Distribution (IID), making the above solution harder to be applied. To address this, research was devoted to strategies of DL model building under the FL paradigm, including the parameters of model exchange among nodes, local training, model updates based upon secure aggregation and decision on the weighted average [22, 81, 67, 82, 83].

In particular, McMahan *et al.* [22] was the first to propose an algorithm to deal with the constraints imposed by distributed data-centers using a naive averaging approach, termed as FedAvg. Being by far the most commonly used FL optimization algorithm, vanilla FedAvg can be viewed as a distributed implementation of the standard Stochastic gradient descent (SGD) with low communication overhead, wherein the local updates are aggregated at the end of each round on each node directly. Instead of averaging the gradients at the server-side, FedAvg optimizes the local version of the model solely on the local data. An extension to FedAvg was made by Wang *et al.* [82] proposing a federated layer-wise learning scheme which incorporates matching and merging of nodes with similar weights. Towards creating a fair aggregator, Li *et al.* [81] and Mohri *et al.* [67] extended the FedAvg algorithm utilizing data from different nodes at a time, selected in such a way that the final outcome is tailored to the problem to be solved, thus mitigating the learning bias issue of the global model. Several researchers have also studied the convergence behavior in statistically heterogeneous settings. In [83], authors make a small modification to the FedAvg method to help ensure convergence. The latter can also be interpreted as a generalized, re-parameterized version of FedAvg that manages systems heterogeneity across devices. Recently, [84, 85] applied FL to action recognition tasks, where the training data may not be available at a single node and nodes have limited computational and storage resources.

It is worth noting that despite the significant strides in all areas, only a few studies address the heterogeneity in data modalities [86, 87, 88, 89], mainly due to the multitude of sensors in edge devices, while the FL-based multi-modal collaborative approach for human action recognition left unexplored. This observation underscores a pivotal and yet underexplored area in the current research framework of FL, particularly in its application to multi-modal human action recognition.

One of the foremost challenges in the domain of FL is the management and integration of heterogeneous data modalities. This heterogeneity is not confined merely to the variations in data distributions across different devices (commonly referred to as non-IID data). It also encompasses the diversity in data types, especially pertinent in scenarios involving multi-sensor edge devices. The integration of varied data forms, such as visual inputs from cameras and kinetic measurements from accelerometers or gyroscopes, necessitates distinct methodologies for processing and analysis. The complexity in FL systems is thus twofold: it involves not only the aggregation of diverse datasets but also their effective interpretation and processing, all while maintaining stringent standards of privacy and computational efficiency.

5.1.2 DL-based 3D Action Recognition

DL techniques have recently been applied in the field of 3D human action recognition aiming at efficiently modeling the observed complex motion dynamics. These have been experimentally shown to significantly outperform the corresponding hand-crafted-based approaches. DL methods can be roughly divided into the following main categories, depending on the type of information that is being used: a) single-modality methods (skeleton-, surface- and flow-based), and b) multi-modal methods.

Skeleton-based methods: Current DL methods mainly focus on the use of skeleton-tracking data, *i.e.* they make extensive use of domain-specific knowledge (employed skeleton-tracker) and relatively straight-forward algorithmic implementations. Several approaches have been proposed using variants of Recurrent Neural Networks (RNNs), which adapt the architecture design towards efficiently exploiting the physical structure of the human body or employ gating mechanisms for controlling the spatio-temporal pattern learning process [90, 91, 92, 93]. Despite the suitability of RNNs in modeling time-evolving procedures, recently CNN-based architectures have also been introduced [94, 95]. Other studies focus on Graph Convolutional Networks (GCNs), which generalize CNNs to non-Euclidean feature spaces [96].

Surface-based methods: Inevitably, DL techniques have also been applied to depth-based action-recognition problems. Wang *et al.* [97] present three simple, compact, yet effective representations of depth sequences, termed Dynamic Depth Images (DDI), Dynamic Depth Normal Images (DDNI) and Dynamic Depth Motion Normal Images (DDMNI). Additionally, Yanghao *et al.* [98] describe a multi-task end-to-end joint classification-regression RNN for efficiently identifying the action type and performing temporal localization. In [99], a human pose representation model is proposed, making use of CNNs and depth sequences that transfers human poses acquired from multiple views to a view-invariant high-level feature space.

Flow-based methods: Despite the remarkable performance improvements achieved by the inclusion of optical flow in the 2D action recognition task [100], the respective motion information in 3D space has been poorly examined. With the advent of real-time 3D optical flow estimation algorithms [101], this signal has become suitable for real-world applications since it enables the focus of the analysis procedure to be put on the areas where motion has been observed. Seeking new means for exploiting the 3D flow information more efficiently, alternative representations of 3D flow fields have been investigated [102, 103].

Multi-modal methods: From the above analysis it can be deduced that the 3D action recognition-related literature has in principle concentrated on single-modality analysis, with skeleton-tracking data being by far the most widely used features. However, from the reported experimental evaluation, it has also become obvious that significant performance improvements and truly robust 3D action recognition schemes can only be obtained using multi-modal information. Until now, multi-modal DL schemes for 2D action recognition have in principle focused on modeling correlations along the spatial dimensions (*e.g.* two-stream CNN [104], trajectory-pooled deep-convolutional descriptors [105], feature map fusion [106]).

On the other hand, for the particular case of 3D actions, Shahroudy *et al.* [107] propose a deep auto-encoder that performs common component analysis at each layer (*i.e.* factorizes the multi-modal input features into their shared and modality-specific components) and discovers discriminative features, taking into account the RGB and depth modalities. The latter again puts emphasis on spatial domain analysis, relying on the initial extraction of hand-crafted features, while the method is not applicable to view-invariant recognition scenarios. Zhao *et al.* [93] propose a two-stream RNN/CNN scheme, which separately learns an RNN model, using skeleton data, along with the convolution-based model, trained using RGB information, and lately fuses the obtained features. Moreover, a graph distillation method is presented in [108], which assists the training of the model by leveraging privileged modalities dynamically.

In the field of human action recognition, the utilization of multi-modal data promises a more holistic understanding of human behaviours. The amalgamation of disparate data sources, such as visual and motion data, offers a richer, more nuanced perspective than what could be achieved through unimodal data. However, the application of such a multi-modal approach within a FL framework remains largely uncharted territory. This gap may stem from the inherent complexities associated with developing algorithms capable of harmoniously integrating and learning from such varied data sources. These complexities are not only technical, involving the synchronization and fusion of multiple data streams, but also ethical, about the safeguarding of privacy across diverse and potentially sensitive information streams.

Developing FL systems for multi-modal data requires addressing several challenges. Firstly, there's the need for robust algorithms that can handle data synchronization and fusion from various modalities. Secondly, ensuring data privacy while processing such diverse data sources is critical, especially when dealing with sensitive information like human actions. Thirdly, the computational and communication overheads in FL systems increase with the incorporation of multiple data types, requiring efficient strategies to handle these increases without significantly impacting performance.

Despite these challenges, exploring FL-based multi-modal collaborative approaches for human action recognition holds significant potential. Such systems can lead to more accurate and comprehensive models, as they would be trained on a diverse range of data sources reflecting real-world scenarios more closely. Moreover, these models could be more generalizable and robust against overfitting to a specific data type. Therefore, addressing this unexplored area could lead to advancements not only in human action recognition but also in the broader field of FL and its applications.

Observations: Taking into account the above analysis, it can be observed that certain rich information sources (namely 3D flow) and their efficient combination with currently widely used ones (*i.e.* skeleton-tracking and depth), have barely been studied. In particular, very few works have concentrated on the problem of multi-modal analysis for reaching truly robust action recognition results. Moreover, FL-based approaches have shown considerable potential in single modality analysis, while their potential in multi-modal scenarios has not been explored yet, with this study being the first, to the authors' knowledge, to address this problem by performing modality fusion at different levels of granularity in different FL settings.

The main contributions of this work are summarized as follows:

a) **a new methodology for enabling the incorporation of depth and 3D flow information** in DL action recognition schemes,

b) **design of multiple Neural-Network (NN) architectures for performing modality fusion at different levels of granularity (early, slow, late),** including a new method that reinforces the Long Short-Term Memory (LSTM) learning capabilities, by modeling complex multi-modal correlations and also reducing the devastating effects of noise during the encoding procedure,

c) **exploration of federated aggregation strategies for incarnating cross-domain knowledge transfer in distributed scenarios**, by either treating each modality as a unique FL instance, or by performing Federated modality fusion (aligning or correlating modalities at the local level) and,

d) **introduction of a new large-scale 3D action recognition dataset**, particularly suitable for DL-based analysis under the FL configuration.

5.2 Methodology and Approach for Handling Data Modality Heterogeneity

5.2.1 Singe-modality analysis

Human actions inherently include a temporal dimension (the so-called 'action dynamics'), the capturing and encoding of which is of paramount importance for achieving robust recognition performance.

Skeleton-tracking analysis: With respect to the skeleton modality, a literature approach is adopted [91], where spatial dependencies among joints and temporal correlations among frames are modeled at the same time, using a so-called Spatio-Temporal

LSTM (ST-LSTM) mechanism. The latter aims at encoding the temporal succession of the representation of its internal state while shifting the recurrent analysis towards the spatial domain; hence, modeling the action dynamics along with spatial dependency patterns. The ST-LSTM approach was selected in this work, due to its relatively reduced implementation complexity, while exhibiting increased recognition performance.

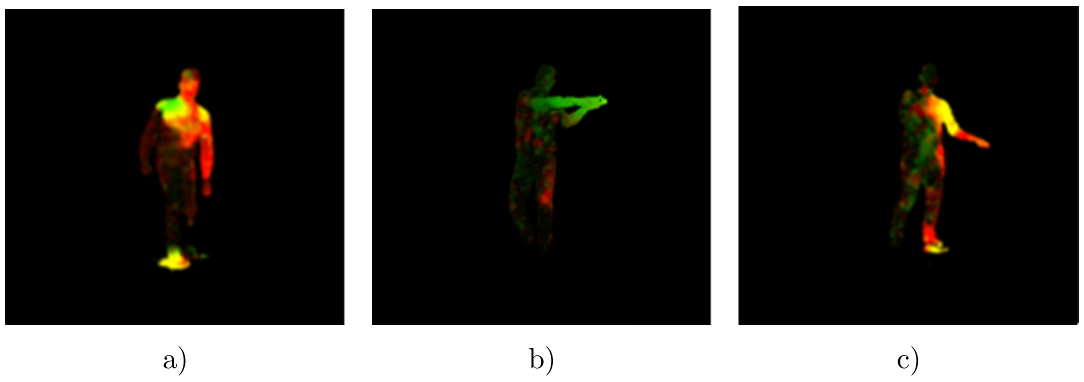


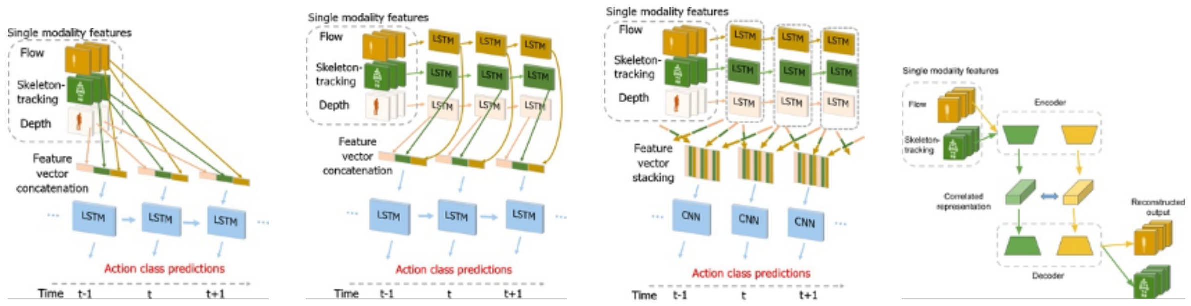
Figure 5.1: 'Colorized' 3D flow fields for actions: a) 'Bowling', b) 'Baseball swing', and c) 'Tennis forehand'. In (b) the green color indicates that there is an intense motion of both arms towards the Y direction, which is the case when someone hits the ball with the baseball bat.

Flow and depth analysis: A new methodology for processing and representing flow (and depth) information is described in this section. One of the major challenges, regarding 3D flow estimation, concerns the corresponding computational requirements, which have hindered its widespread use so far. However, computationally efficient 3D flow estimation algorithms have recently been introduced with satisfactory flow computation accuracy. In this work, the algorithm of [101] has been employed, which exhibits a processing rate equal to 24 frames per second (fps). For computing a discriminant 3D flow representation for each video frame, while taking advantage of the DL paradigm, 3D CNNs are employed, due to their increased ability to model complex patterns at multiple scales along the spatial and spatio-temporal dimensions [109]. In particular, as an initialization step, the 'transfer learning' approach has been followed in this work, similarly to the works of [105, 100] that employ RGB pre-trained models for the case of 2D flow-based analysis. In order to enable the use of pre-trained 3D CNNs (*i.e.* models that originally receive as input RGB information) in the estimation of the proposed 3D flow representation, an appropriate transformation of 3D flow to RGB-like information is required, as can be seen in Fig. 5.1. For depth-based analysis, an approach similar to the flow-based one described above is followed.

Action recognition realization: For performing action recognition, an individual LSTM network is introduced for every considered modality, namely skeleton-tracking, depth and flow data. Color information is neglected since it is considered that the type of the observed action does not depend on the color of the subjects' clothes. Regarding the composite 3D CNN-LSTM architecture, the introduced flow and depth representations are computed by considering the spatio-temporal features from the last FC layer of the 3D CNN model [110]. For every action instance, the video is split into 16-frame clips with 8 frames overlap, where a constant number of T clips is selected. For each clip, depth and flow features are extracted above. The developed single-modality LSTMs are trained to predict the observed action class at every video segment, while for estimating an aggregated probability for each action for the entire video sequence, simple averaging of all corresponding probability values of all clips is performed.

5.2.2 Multi-modal analysis

Different modalities exhibit particular characteristics with respect to the motion dynamics that they encode. To this end, a truly robust action recognition system should combine multiple information sources. In this respect, NN architectures, which realize different early, slow, and late fusion schemes of the modalities discussed in Section 5.2.1, are presented in this section. The proposed modality fusion schemes are generic and expandable, *i.e.* they can support a varying number of single-modality data streams in the form of observation sequences. Particular attention is paid to adapting the FL paradigm to the multi-modal setting, by proposing a set of federated aggregation strategies tailored to the cross-domain knowledge transfer scenario.



a) Early fusion b) Slow fusion c) Late fusion d) AE-based fusion

Figure 5.2: Proposed multi-modal fusion architectures: a) concatenation of the different single-modality features at every time instant, b) concatenation of the different single-modality features at the LSTM state space, c) simple stacking of the LSTM state signals, and d) shared/correlated latent representation features.

Early fusion: The first investigated modality fusion scheme essentially constitutes a straight-forward early fusion one, where simple concatenation of the different single-modality features is performed at every time instant and the resulting composite feature vector is subsequently provided as input to an LSTM (Fig. 5.2). In this case, the LSTM is assigned the challenging task of estimating correlations among feature vectors of diverse type and unequal dimensionality. In order to account for the difference in the nature of the single-modality features, every element of each vector is normalized in order to eventually have a mean value equal to zero and a standard deviation equal to one, before being provided as input to the LSTM. The same feature normalization procedure is followed in all fusion schemes presented in this section.

Slow fusion: In order to better estimate the correlations among data coming from different sources and also to handle the problem of the varying dimensionality of the corresponding feature vectors, a slow fusion scheme is introduced. The aim is to combine the unimodal data not in the original feature spaces, but in the single-modality LSTM networks' state space. For achieving this, the LSTM state signals $\mathbf{H}_n(t)$, $n \in \{s, d, f\} \equiv \{skeleton, depth, flow\}$, are considered. Then, a composite multi-layer LSTM is developed, by introducing additional layer(s) on top of the single-modality ones; the additional LSTM layer(s) receive(s) as input a composite vector that results from the simple concatenation of the state signals $\mathbf{H}_n(t)$, as can be seen in Fig. 5.2.

Late fusion: The early and slow modality fusion schemes rely on feature vector concatenation techniques. In this section, a novel NN-based late fusion scheme (Fig. 5.2) is introduced, which exhibits the following advantageous characteristics: a) it follows a CNN-based approach for efficient modeling of complex cross-modal correlations, and b) it simultaneously takes into account multi-modal information from multiple frames for predicting the action class for the currently examined one, contrary to the common practice of updating the class prediction probabilities by considering only the current frame; the latter renders the proposed scheme more robust in the presence of noise and enables the incorporation of the temporal dimension during the estimation of the multi-modal fusion patterns.

Prior to the application of the proposed late fusion scheme, single-modality analysis is realized for each information source as detailed in Section 5.2.1. Then, the LSTM state signals $\mathbf{H}_n(t)$ are again considered. Subsequently, for every frame t, the state vectors $\mathbf{H}_n(t)$ of all frames that lie in the interval $[t-\tau, t+\tau]$, $\tau > 0$, are stacked, according to the following expression:

$$\begin{aligned}\mathbf{H}^M(t) = [\mathbf{H}_s(t-\tau)\ \mathbf{H}_d(t-\tau)\ \mathbf{H}_f(t-\tau)\ \dots \\ \dots\ \mathbf{H}_s(t)\ \mathbf{H}_d(t)\ \mathbf{H}_f(t)\ \dots \\ \dots\ \mathbf{H}_s(t+\tau)\ \mathbf{H}_d(t+\tau)\ \mathbf{H}_f(t+\tau)],\end{aligned} \tag{5.1}$$

where $\mathbf{H}^M(t)$ is the resulting 2D matrix, containing multi-modal information. For modeling the correlations among the multi-modal data, while simultaneously taking into account information from multiple frames, a CNN is introduced, which receives as input the above-mentioned $\mathbf{H}^M(t)$ matrix and estimates for every frame a corresponding action class probability vector $\mathbf{P}(t)$. The developed CNN consists of two convolutional layers, which model the correlations among the multi-modal features, and two fully connected layers, for computing vector $\mathbf{P}(t)$. For estimating the action class probabilities of a whole video sequence, the average of all $\mathbf{P}(t)$ values is calculated, taking into account all video frames.

The developed early and slow fusion schemes rely solely on the use of LSTMs, rendering them prone to the presence of noise in the input signal, missing also valuable information from neighboring frames. By comparing equations (5.2) and (5.3), it can be easily observed that the proposed CNN-based late fusion scheme allows the modeling of significantly more detailed and complex multi-modal fusion patterns,

$$\mathbf{P}(t) = \mathbf{W}_{hp}\mathbf{H}(t) + \mathbf{B}_p \tag{5.2}$$

$$\mathbf{P}(t) = \Phi_{CNN}(\mathbf{H}^M(t)) \tag{5.3}$$

where $\Phi(.)$ denotes the transformation learned by the CNN model, $\mathbf{P}(t)$ the corresponding target output, internal state vector $\mathbf{H}(t)$, $\mathbf{W}$ the learnable weight matrices and $\mathbf{B}$ the biases.

Multi-modal Federated optimization: Optimization strategies and in particular aggregation algorithms play an important role in FL as they are responsible for combining the knowledge from all devices/nodes while respecting data's privacy. Prior to the application of the proposed multi-modal FL scheme, single-modality FL analysis is realized for each information source, as already detailed in Section 5.2.1. In particular, two main aggregation mechanisms have been followed, examined in detail and applied to specific scenarios, namely, the FedAvg, and FedProx.

FedAvg: In each round, the algorithm performs a series of local SGD model updates on a subset of clients, followed by a server-side aggregation task, trying to minimize the following objective function, which is actually the sum of the weighted average of the clients' local errors, where F_k is the local objective function for the kth device and p_k specifies the relative impact of each device:

$$\min_w = \sum_{k=1}^{N} p_k F_k(w) \tag{5.4}$$

FedProx: At each step, the algorithm adaptively selects amounts of work to be performed locally across devices based on their available systems resources and then aggregates the partial solutions received so far. The aim here is to minimize the following objective function h_k, which, as in the previous case, takes into account local losses while constraining local updates to be closer to the previously seen global model, where μ is the control parameter as described in detail in [80]:

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}||w - w^t||^2 \tag{5.5}$$

Although adapting FL to uni-modal nodes seems to be a trivial task, shifting to more complex architectures that fuse multi-modal streams, such as the ones described above, may not be as easy as it seems. In this respect, a naive yet synchronous approach was followed that first deploys different fusion schemes locally, and then aggregates the local updates by applying the optimization mechanisms (*i.e.* FedAvg, FedProx) as in the simple case of uni-modal data. However, the latter fails in cases where some nodes have access to multi-modal local data while others only to uni-modal, hence limiting the scalability of the system.

Inspired by the works of [107, 111, 112, 89], a modular multi-modal fusion scheme is proposed, that utilizes Auto-Encoders (AE) from different data modalities at the local level, enabling more advanced FL aggregation schemes at the global level. The main idea of AE for multimodal learning is to first encode data from a source stream as hidden representations and then use a decoder to generate features for the target stream. To this end, the proposed single modality schemes in section 5.2.1 have been adapted to meet the encoder-decoder requirements (*i.e.* utilized 3D CNN and LSTM AEs for flow, depth and skeleton modalities respectively). For clarity, three different fusion schemes using AEs were evaluated: a) the Multi-modal AE, where the input modalities were processed separately with a non-shared encoder-decoder part, after which these hidden representations from the encoder are constrained by a distance metric (*e.g.* Euclidean distance), b) Variational AE (VAE) [113], where the input modalities were projected into a shared latent space with the objective to minimize the reconstruction loss of both streams and c) the Correlation AE (CCAE) [114], where feature correlations are captured by cross-reconstruction with similarity constraints between hidden features, with the objective to minimize both the reconstruction loss while increasing the canonical correlation between the generated representations. In a federated setting, at each round, the nodes will locally train their models to extract non-shared, shared or correlated representations. Then, local updates from all types of representations are forwarded to the server and aggregated into a global model by the selected multi-modal optimization algorithm [89]. Since the representation features have a common structure, the aggregation algorithm is able to combine models trained on both unimodal and multimodal data. Finally, through a supervised learning process, in which the resulting encoder is used to extract internal representations from an annotated dataset, a cross-modal classifier is produced. The proposed multi-modal aggregation algorithm aims to minimize the following expression, which is actually the sum of the partial errors of the unimodal and multimodal parts respectively, balanced by a parameter λ, where $F_k(w_A)$ is the reconstruction error of modality A for the kth device.

$$\min_{w} = \sum_{k=1}^{N_A} F_k(w_A) + \lambda \sum_{k=1}^{N_{AB}} F_k(w_A) \tag{5.6}$$

5.3 Experimental Setup and Data Description - Experimentation and Results

5.3.1 3D Action Recognition Dataset

The fundamental prerequisite for the application of any DL technique constitutes the availability of vast amounts of annotated training data. In this context, further details about the main public 3D action recognition datasets currently available can be found [115]. In order to further boost research in the field, a large-scale dataset, significantly broader than most datasets indicated in [115], and in Table D.1 of Appendix D, has been formed and made publicly available [1].

[1]https://vcl.iti.gr/dataset/

The formed dataset exhibits the following advantageous characteristics: i) It can be used for reliable performance measurement (only NTU RGB+D dataset[1] exhibits such wealth of information in the literature). ii) The later[1] is indeed broader in terms of overall action instances, however, the introduced one involves three times more different human subjects (namely 132 participants) though; hence, rendering it highly challenging, with respect to the exhibited variance in the subjects' appearance and the execution of the exact same actions. iii) It can further facilitate the application of DL techniques in the field of 3D action recognition, where the prerequisite is the presence of multiple, ever larger and diverse data sources. iv) It contains a large number of sport-related and exercise-related actions, which do not exist on any other benchmark 3D action dataset, as [1] dataset mainly supports daily and health-related actions; hence, highlighting its significance. v) The availability of 3D flow information in the formed dataset can significantly boost the currently largely untouched field of motion-based 3D action recognition. vi) Provides a set of validated dataset partition strategies, which ensure that the data splits follow a close-to real-world distribution, making it suitable for FL applications.

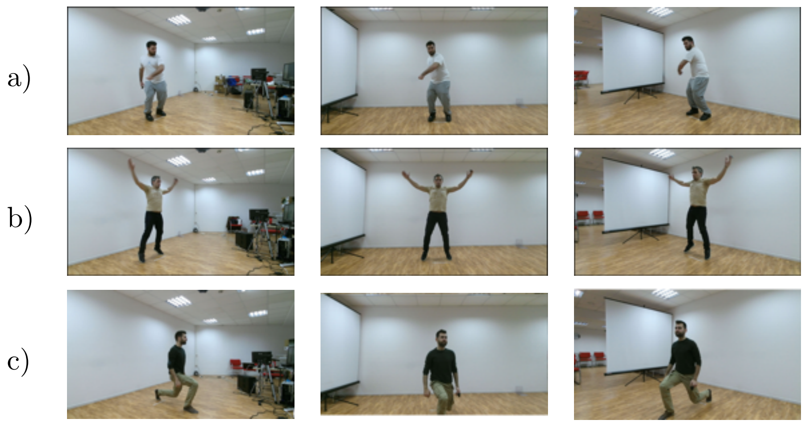


Figure 5.3: Examples of the formed dataset of multi-view action capturings for actions: a) 'Tennis backhand', b) 'Jumping jacks' and c) 'Lunge'.

The formed dataset was captured under controlled environmental conditions, *i.e.* with negligible illumination changes (no external light source was present during the experiments) and a homogeneous static background (all participants were standing in front of a white wall). For realizing data collection, a capturing framework was developed, which involved three synchronized Microsoft Kinect II sensors positioned in an arced configuration. The sensors were placed at a distance of approximately 2.5-3 meters from the performing subjects. One sensor recorded a frontal view of the subjects, while the other two captured diametrically opposed views (both forming a 45^o angle with respect to the frontal view direction). The formed dataset contains the following information sources: a) RGB frames, b) depth maps, c) skeleton-tracking data, and d) 3D flow fields. Snapshots of the captured video streams are depicted in Fig. 5.3.

Regarding the type of supported human actions, a set of 50 classes was formed.

Specifically, the set includes 18 sport-related (*e.g.* 'Basketball shoot', 'Golf drive', 'Tennis serve', *etc.*), 25 exercise-related (*e.g.* 'Running', 'Jumping jacks', 'Stretching', *etc.*) and 7 common daily-life actions (*e.g.* 'Clapping', 'Drinking', 'Waving hands', *etc.*). It needs to be mentioned that there is a distinction between left- and right-handed actions, *i.e.* they correspond to different classes.

During the capturing phase, a total of 132 human subjects, in the range of 23-44 years old, participated. All participants were asked to execute all defined actions. The latter resulted in the generation of approximately 8545 performed unique actions and a total of 25635 instances, considering the capturing from every individual Kinect as a different instance. The length of every recording varied between 2 and 6 seconds.

Additionally, certain dataset partitions are required to make the introduced dataset suitable for FL application, utilising strategies, which ensure that the data splits follow a close to real-world distribution. The strategy described by Wang *et al.* [82] was followed, to simulate both homogeneous and heterogeneous partitions. This approach is based on a Dirichlet sampling algorithm, where a parameter α controls the amount of heterogeneity with respect to data size and classes' distribution. Overall, the involved human subjects were randomly divided into training, validation and test sets, each comprising 20%, 30% and 50% of the total individuals, respectively. Further information regarding the dataset, encompassing pre-processing steps and extra samples for each modality, is available in Appendix E.

Data distribution and implementation details: In this section, experimental results from the application of the proposed 3D action recognition methods are presented. For the evaluation, the introduced dataset was used as a benchmark, and further, it was converted into two non-IID dataset variants to support the FL scenarios. One is replicating the real-life data, *i.e.* real-world dataset, while the other is the extreme example of a non-IID dataset (denoted as D_1 and D_2 respectively). For both datasets, a set of 50 action classes was defined and a total number of 3 nodes was chosen among which actions are to be distributed. As mentioned above, the parameter $\alpha > 0$ controls the identicalness among participants. Different α values were tested, where with $\alpha- > \infty$, all participants have identical distributions and $\alpha- > 0$, each participant has examples from only one class. To support D_1 the set was divided with medium heterogeneity by setting $\alpha = 1$. Therefore, a node can have actions from any number of classes. This type of dataset replicates the real-world scenario in which different clients can have different types of action. In contrast, for the case of the non-IID set D_2, the original dataset was divided, with a higher level of heterogeneity by setting $\alpha = 0.5$. Here, nodes tend to have a significant number of samples from some classes and few or no samples for the other classes. Each node randomly sampled $\frac{1}{3}$ of training, validation and test set data respectively. The experiments reported in Tables 5.1 and 5.2, highlight the impact of different learning algorithms and data distribution variations on local as well as global model's performance.

Regarding implementation details, the single-modality and early fusion LSTMs

Table 5.1: Action recognition results in D_1: a) Single-modality analysis, b) Multi-modal fusion. Methods indicated with superscripts 's', 'c', 'd' and 'f' incorporate skeleton, color, depth, and flow data, respectively. Accuracy obtained at server level (Acc_S), mean Accuracy obtained at node level ($mAcc_C$). In the centralized scenario, all data is gathered in one node, where $mAcc_C$ is '-', the exact opposite happens in the local (isolated) scenario where Acc_S is '-', while in the federated scenarios, access is permitted to both local and global data.

$Acc_S/mAcc_C$	Method	Centralized	Local	FedAvg	FedProx
a)	Depth	79.54% / –	– / 57.84%	72.63% / 66.18%	73.06% / 66.43%
	Skeleton	80.27% / –	– / 57.32%	74.12% / 66.45%	74.58% / 66.79%
	Flow	85.49% / –	– / 60.53%	78.35% / 70.21%	78.83% / 70.55%
b)	Early^{sdf}	82.75% / –	– / 61.28%	76.72% / 68.94%	77.20% / 69.26%
	Slow^{sdf}	87.12% / –	– / 64.87%	78.84% / 70.82%	79.31% / 71.05%
	Late^{sdf}	87.95% / –	– / 65.26%	79.25% / 71.48%	79.57% / 71.61%
	VAE^{sdf}	84.39% / –	– / 62.11%	76.96% / 69.04%	77.25% / 69.47%
	CCAE^{sdf}	85.43% / –	– / 62.57%	77.31% / 69.18%	77.93% / 69.82%

consisted of three layers, while the slow fusion scheme required the addition of one more layer on top. In all LSTM configurations, each layer included 2048 units. A set of 32 frames were uniformly selected for feature extraction, which roughly corresponds to one-third of the average number of frames per action. For the particular case of 3D CNN, each video sequence was split into 16-frame clips with 8 frames overlap. Prior to feature extraction, simple depth thresholding techniques were used to maintain only the subjects' silhouettes. Additionally, for depth and flow feature extraction, data transformation (e.g. cropping, resizing etc.) techniques have been applied.

With respect to the implemented fusion approaches, the 'Torch[2]' scientific computing framework and 4 Nvidia GeForce RTX 3090 GPUs were used at each node. Zero-mean Gaussian distribution with a standard deviation equal to 0.01 was used to initialize all NN weight and bias matrices. All class predictions (both at the node and server-side) were passed through a softmax operator (layer) to estimate a probability distribution over the supported actions. The batch size was set equal to 256, while the momentum value was equal to 0.9. Weight decay with a value of 0.0005 was used for regularization. For the federated setting, a total number of 100 communication rounds was initially selected. In each communication round, training on individual clients takes place simultaneously. In this respect, for the early fusion case, the training procedure lasted 80 epochs, while for the slow, late and attention-based fusion ones, 30 epochs were shown to be sufficient.

5.4 Results and Analysis

5.4.1 Single-modality Evaluation

In Tables 5.1 and 5.2, quantitative action recognition results are given in the form of the overall classification accuracy, *i.e.* the percentage of all action instances that were correctly classified, for different federation scenarios based on data availability, namely the centralized, the local(isolated) and the federated. For the centralised training, the resulting models are based on the combined training dataset at a central node. The

[2]https://pytorch.org

performance of these models will be used as a benchmark in a thorough evaluation against the proposed FL algorithms. In the local learning setup, the results models were trained solely on local datasets at the node level. In all setups, the global test set of all nodes was used. From the first group (a) of the provided results (*i.e.* single-modality ones), it can be seen that the introduced 3D flow representation achieves the highest recognition rates, among the single-modality features. The latter demonstrates the increased discrimination capabilities of the flow information stream. It should also be stressed that in all single modality cases, the centralized approach outperforms the others (both local and federated ones), in some cases even by a large margin (over 25%), but what can be pointed out is that the aggregation techniques show a significant improvement compared to the isolated local version ones (*i.e.* relative improvement of more than 18%). Summarizing the findings, it can be stated that federated strategies provide acceptable performance recognition compared to centralized ones while giving the feeling that it can be increased even further by incorporating more sophisticated averaging techniques.

5.4.2 Multi-modal Evaluation

Concerning the proposed modality fusion schemes [second (b) group of experiments in Tables 5.1 and 5.2], it can be observed that the introduced late fusion mechanism exhibits the highest overall performance. This is mainly due to the more complex and among multiple frames cross-modal correlation patterns that the developed CNN encodes. Surprisingly, the early fusion scheme (despite its usual efficiency in multiple information fusion tasks) is experimentally shown not to be effective in the current case; this is probably mainly due to the relatively high dimensionality and significant diversity of the involved uni-modal features. Examining the behavior of the multi-modal AE schemes in more detail, it is shown that performance is maximized in the case of the CCAE, highlighting the added value of the correlation analysis performed at the latent representation layer among the participating modalities. Despite their potential, none of the proposed AE-based methods can approach the performance of the slow and late fusion approaches, resulting in a reduction of the overall classification efficiency by about $2-3\%$. However, what makes these methods unique is the ability to train using more than one data stream, while evaluating with just one. Similarly to the findings of the single mode experiments, the FL optimization algorithms can provide a reliable solution to the problem, showing a small drop of 8%. By comparing results obtained in D_1 and D_2 it is obvious that more heterogeneous data splits require more advanced FL algorithms. Further analysis, indicates that the locally trained models achieve very poor performance for most of the actions and they tend to be biased since visibly prefer predicting the classes that they presented at training. In contrast to the global aggregated models that exhibit acceptable performance in both (a) and (b) sets, around 70% on average; thus, clearly highlighting the added value of the FL paradigm in high-heterogeneous distributions.

Table 5.2: Action recognition results in D_2: a) Single-modality analysis, b) Multi-modal fusion.

$Acc_S/mAcc_C$	Method	Centralized	Local	FedAvg	FedProx
a)	Depth	79.54% / –	– / 37.26%	64.45% / 60.02%	65.12% / 60.37%
	Skeleton	80.27% / –	– / 37.25%	66.23% / 61.38%	66.59% / 61.85%
	Flow	85.49% / –	– / 40.69%	71.02% / 66.19%	71.44% / 66.65%
b)	Early^{sdf}	82.75% / –	– / 41.49%	68.43% / 62.84%	68.84% / 63.24%
	Slow^{sdf}	87.12% / –	– / 44.73%	70.15% / 65.28%	70.62% / 65.68%
	Late^{sdf}	87.95% / –	– / 45.37%	72.42% / 66.63%	72.94% / 67.11%
	VAE^{sdf}	84.39% / –	– / 42.34%	68.77% / 62.98%	69.11% / 63.95%
	CCAE^{sdf}	85.43% / –	– / 42.71%	69.21% / 63.24%	69.47% / 64.36%

5.5 Discussion and Findings

In this comparative study, we delve into the performance of various FL strategies, such as FedAvg and FedProx, within the domain of 3D action recognition. Our research predominantly focuses on the impact of different data modalities, namely depth, skeleton, and flow, on the overall system performance. The examination reveals significant insights into the functionality and limitations of these FL strategies under varied network conditions and data heterogeneity.

One of the critical findings is the varying effectiveness of FL strategies in handling non-IID data distributions. The challenges encountered in multi-modal federated optimization, particularly regarding communication overheads and data synchronization, are highlighted. This study also explores the strengths and weaknesses of each data modality and its fusion strategies, including early, slow, and late fusion. It becomes evident that the fusion of these modalities plays a pivotal role in enhancing the recognition performance.

Through our experimental data, we observe a distinct performance disparity between the FL algorithms. FedAvg, while being the most commonly used, shows certain limitations in its ability to handle complex multi-modal scenarios effectively. In contrast, FedProx offers an extension with improved performance, particularly in managing systems heterogeneity across devices. This is crucial in FL environments where data is distributed across various nodes with different computational and storage capacities. Additionally, our analysis sheds light on the effectiveness of multi-modal data fusion in FL. By integrating various types of data, such as visual inputs from cameras and kinetic measurements from accelerometers or gyroscopes, FL systems can achieve a more nuanced understanding of human actions. However, the complexity of integrating these diverse datasets and interpreting them effectively, all while maintaining privacy and computational efficiency, remains a significant challenge.

In conclusion, this study underscores the importance of advancing FL techniques in multi-modal human action recognition. It highlights the need for innovative solutions to address the challenges of data heterogeneity and fusion strategies, paving the way for more accurate, efficient, and privacy-preserving FL systems in real-world applications.

Chapter 6

Study 3: Representation Learning and Federated Distillation in FL

6.1 Overview of the Study

Building upon the insights gained from the initial explorations in computer vision and text categorization, and the subsequent deep dive into data type heterogeneity in federated multi-modal 3D human action recognition, the third study in this series introduces a novel federated distillation weight aggregation method. This innovative approach is tailored for distributed learning environments, where it addresses the challenge of efficiently extracting and transferring knowledge across multiple nodes in a federated network. Central to this study is the development of an algorithm that adeptly captures meaningful representations from various client nodes and models this information into actionable knowledge. This knowledge is then effectively consolidated at a central server using a feature mixing technique. The aggregated knowledge is subsequently disseminated back to the client nodes for further refinement and distillation. A series of comprehensive experiments were conducted to evaluate the efficacy of this method. These experiments shed light on the effectiveness of the proposed algorithm in an FL context, particularly highlighting its capacity to maintain robust performance while significantly curtailing communication costs. This study, therefore, marks a critical advancement in FL, optimizing the knowledge transfer process through representation learning and enhancing the overall efficiency and effectiveness of distributed learning systems.

6.1.1 Challenges and Solutions in Representation Learning

The exponential growth in smart device usage precipitates a marked augmentation in data transmission volumes, presenting a significant challenge in the realm of distributed data management. A paramount concern in this domain is the facilitation of effective training protocols for decentralized clients, avoiding the necessity for inter-client data exchange. To overcome this challenge, FL [22] has been developed as a pivotal methodology, enabling training of a centralized model whilst ensuring the retention of users' sensitive data exclusively within their personal devices.

Nevertheless, the FL approach involves a communication overhead being proportional to model sizes. Particularly, models of large size prove to be impractical for deployment. Furthermore, the necessity for model aggregation in the FL training process imposes a lack of flexibility in the architecture of client models, as each client is required to train a model with an identical architectural framework.

Towards this direction, a novel technique known as Federated Distillation (FD) has been put forward. This approach diverges from the traditional model aggregation method, opting instead for the exchange and aggregation of client model logits. These aggregated logits are then utilized in the model distillation phase to disseminate the knowledge acquired by local datasets among all clients. This alternative methodology offers advantages such as reduced communication costs, greater flexibility in client model architecture, and improved handling of non-IID (independently and identically distributed) data.

This technique was first introduced by Jeong *et al.* [116], where they proposed using a Generative Adversarial Network (GAN) as a central model. In this approach, clients upload per-label averaged soft targets to train the GAN. Subsequently, clients download the GAN generator and produce samples for underrepresented labels, aiming to achieve an IID dataset. A prevalent approach in the literature involves the use of a common dataset accessible to all clients for local distillation, leading to improved model performance. In [117], Li *et al.* presented a scenario where each client possesses a small labeled dataset alongside a larger public dataset accessible to all. The training process involves initial training on the public dataset, followed by training on the private dataset. Logit vectors are then transmitted to the server for aggregation. In their study [118], Itahara *et al.* introduced a semi-supervised method that employs a labeled private dataset and a shared unlabeled dataset. They altered the aggregation step by proposing an Entropy Reduction Aggregation (ERA), demonstrating that using a temperature lower than one when applying softmax to aggregated logits reduces the entropy of global soft targets. This approach is particularly beneficial in non-IID settings.

A subset of methods employ the server solely as an aggregator (similar to conventional FL methods) for locally computed model logits. More recent strategies have incorporated a server distillation step, which distills a server-side model that can be used to construct global logits (or soft targets) for broadcasting. For instance, Cheng *et al.* [119] employ both a public shared dataset and a private dataset, utillizing smaller client models alongside a larger server model. The server categorizes instances in the public dataset into correctly and incorrectly predicted subsets to enhance convergence. In [120], a one-shot distillation method is introduced. The proposed technique combines distillation and aggregation mechanisms to support FL. In this approach, client models are fully trained prior to being transmitted to the server, which then aggregates per-class attention maps.

The methods previously discussed predominantly utilize averaging of client logits as their aggregation technique. However, there has been limited exploration of alternatives to this approach. The proposed technique delves into the fusion process of client logits during the aggregation process, which we contend is a crucial factor affecting the performance of the global model. Consequently, this paper introduces an innovative aggregation method designed to refine the selection criterion. Drawing inspiration from the work of Parvaneh *et al.* [121], a feature mixing aggregation method is proposed, though with a different objective. In [121], the potential label change resulting from feature mixing identifies the images that require labeling. Conversely, in our approach, any potential label change is considered undesirable and leads to a reduction in the weight factor. A trainable alpha vector for the weighted aggregation of client logits is introduced, while a global model is maintained on the server side to train this vector. The main contributions of this work are as follows:

a) The introduction of an innovative federated distillation scheme, which leverages a global server model to guide the aggregation process, encompassing a more advanced aggregation of model outputs.

b) The proposal of a novel aggregation method is outlined. This method utilizes feature mixing and incorporates a trainable vector, designed to assign reduced weight factors to images that are likely to induce label changes.

c) The effectiveness of the proposed approach is validated through extensive comparative analysis on one of the most comprehensive publicly accessible benchmarks, demonstrating its superiority and robustness across a diverse array of FL scenarios.

6.2 Methodology and Approach

6.2.1 Problem statement

In a federated system, data are inherently confined to individual clients, and their exchange with other clients is stringently prohibited. Following a common literature practice in FD systems, each client possesses a distinct local dataset. Additionally, there exists a public dataset that is shared among all clients. On the server side, a global model is maintained, which is trained using this global dataset.

A main challenge in the federated distillation systems lies in the aggregation of the client logits on the server side. The prevailing method in the literature is to simply average these logits. However, this approach fails to consider that data heterogeneity or varying sizes of local datasets might lead some local models to transmit lower-quality logits to the server, which could subsequently decrease local models' performance during the distillation process. To address this issue, the proposed method adopts a strategy where client logits are gradually added, with a specific focus on penalizing those logits that result in a label change.

6.2.2 Local client training

The local clients undergo supervised training, where each client is assigned a distinct segment of the dataset. This allocation remains consistent throughout all federated rounds, with clients training exclusively on their respective segments. The training subset of the dataset is used for model training purposes, whereas the validation subset serves to assess and verify the performance of the models trained by each client. During the training phase, the Cross-Entropy loss function is utilized:

$$L(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} y_{mk}\log p_{mk} \tag{6.1}$$

where θ represents the model parameters, M denotes the total number of images, K the total number of classes, y_{mk} the label of the m, k image and p_{mk} the probability of the class.

Aside from each client's training and validation sets, there is a shared communal dataset accessible to all clients, following the common literature practice ([117], [119]). This dataset acts as a reference point for addressing the issue of data heterogeneity across clients. Each client produces model outputs for each sample in the communal dataset at the end of each federated round. These outputs are then transmitted to the server for aggregation. To ensure knowledge alignment across all clients, in the subsequent federated round, each client engages in knowledge distillation using the global anchors derived from the communal dataset. The knowledge distillation term is computed using the following expression:

$$\mathcal{L}_{KL} = ||\Phi(o_r^c) - \Phi(o_{r-1})|| \tag{6.2}$$

where Φ represents the network function, o_r^c signifies the model output of client c at the federated round r and o_{r-1} denotes the global anchors' aggregation of the previous round.

The limited amount of data that is communicated from the clients to the data in the federated distillation scheme is a disadvantage, as only the output from the model's last fully connected layer is sent to the central server. To mitigate this limitation, a multi-scale knowledge distillation strategy is implemented, drawing inspiration from the research of [122]. Specifically, each client transmits certain intermediate outputs (also known as hidden representations), computed over the communal dataset, in addition to the model's output; the number and the position of these intermediate outputs can vary depending on the specific requirements of the problem. Thus, the multi-scale knowledge distillation term is calculated as follows:

$$\mathcal{L}_{multi-scale_{KL}} = \frac{1}{L}\sum_{l=1}^{L}\Big||\Phi(z_{l_j}^{N}) - \Phi(z_{l_{j-1}})\Big|| \tag{6.3}$$

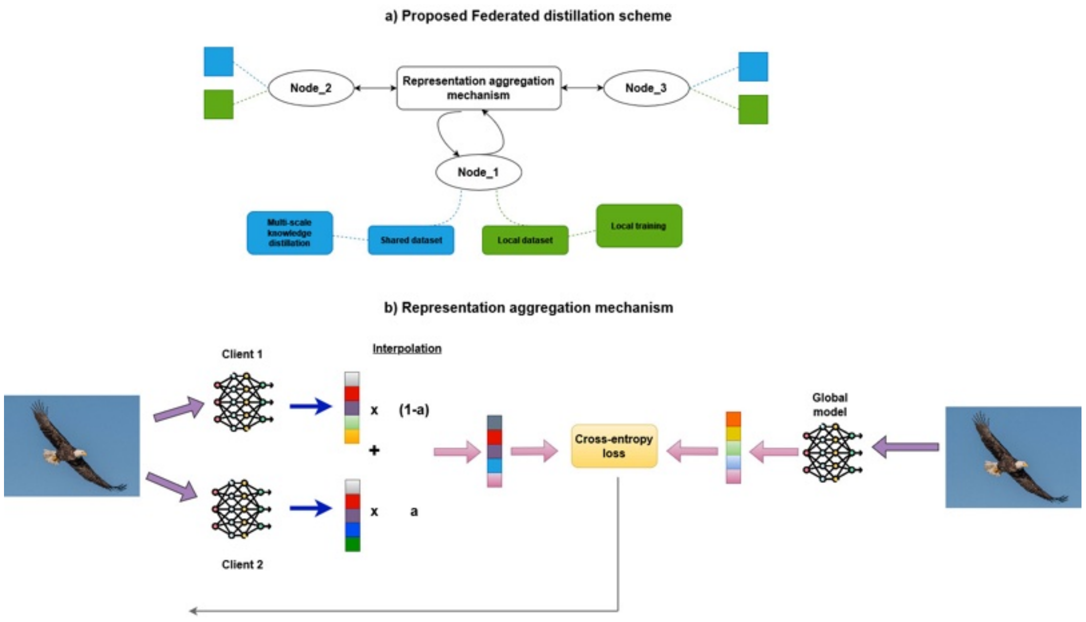

a) Proposed Federated distillation scheme
Node_2
Representation aggregation mechanism
Node_3
Node_1
Multi-scale knowledge distillation
Shared dataset
Local dataset
Local training
b) Representation aggregation mechanism
Client 1
Interpolation
x
(1-a)
+
x
a
Client 2
Cross-entropy loss
Global model

Figure 6.1: (a) The outline of the proposed algorithm. (b) The representation aggregation mechanism on the server-side.

where $z_l^N = f_l^N(\cdot), l \in (1, ..., L)$, l represent the different model layers, z_l^N signifies the intermediate output at layer l for the client N, with size $H * W * C$ and $f_l^N(\cdot)$ denotes the subnetwork for feature extraction up to the layer l.

6.2.3 Global representation aggregation

A feature mixing aggregation method is proposed in this work, implemented by linear interpolation, inspired by [121]. For each client, a tensor of random numbers is generated from separate normal distributions. The mean and standard deviation of these distributions are determined by a hyperparameter denoted as a_{cap}. The dimension of the tensor is equal to the model output, calculated as $n_{classes} * n_{samples}$, where $N_{classes}$ represents the number of the dataset classes and $n_{samples}$ represents the samples' number of the communal dataset. This tensor is utilized to perform interpolation between the current client's model output and a convex combination of the model outputs from the previous clients.

The interpolation process begins with the first client's model output being combined with the second client's model output. Subsequently, the model output of the third client is interpolated with the convex combination of the model outputs from the first two clients, and so on. The linear interpolation is computed using the following equation:

$$LA = (1 - a)\, o^* + a\, o^n \tag{6.4}$$

where a represents the generated tensor, o^* signifies the convex combination of the clients' model outputs (or the first client) and o^n denotes the model output of the client n, $n \in [0, n_{clients}]$, $n_{clients}$ represents the number of clients.

On the server side, there exists a server model that undergoes training using the communal dataset. The server model output is then compared with the feature mixing of c^* and c^n. The Cross-Entropy loss function is utilized as the loss function of the global model:

$$L_G(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} y_{mk}\log p_{mk} \tag{6.5}$$

Regarding the loss of the learnable a tensor, it is calculated as follows:

$$L(a) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} LA_{mk}\log p_{mk} + c_n norm(a) \tag{6.6}$$

where a represents the alpha tensor and the norm refers to the Frobenius norm of the tensor. After training the a tensor and prior to the feature mixing step, it is limited to the range $[0, 1)$. Thus, the Frobenius norm penalizes large values of a.

The process outlined in this section is repeated $n_{clients} - 1$ times to integrate each client's model output for a specified number of epochs (in this work, is set to 5). To prevent overfitting of the global model, the model gradients are scaled by $1 \oslash O_{diff}$, where $\oslash$ denotes the element-wise division, as indicated below:

$$(\frac{\partial L_G}{\partial W})' = 1 \oslash O_{diff} \otimes \frac{\partial L_G}{\partial W}, \tag{6.7}$$

where W represents the model's parameters and $\otimes$ denotes the element-wise multiplication.

6.2.4 FedFMRL Algorithm

This section presents the training flow of the FedFMRL procedure, which is organized into four main algorithms. In Algorithm 9, the main outline of the proposed method is provided. Algorithm 10 describes the training and update steps within each local client, while Algorithm 11 details the distillation process carried out by the local clients. Lastly, in Algorithm 12, the methodology for aggregating global representation is presented.

Algorithm 9 FD Algorithm

```
Require: $T$ is the number of communication rounds, $\mathcal{N}$ is the total number of clients, $\theta_l^i$ represents the parameters of the local models, $\theta_g$ represents the parameters of the global model, $\mathcal{D}_l^i$ are the separate datasets for the local clients, $\mathcal{D}_{cm}$ is the dataset of the communal dataset, $\mathcal{D}_{test}$ is the common testing dataset, $e_g$ are the epochs of the global model, $\eta$ is the learning rate and $a_\eta$ is the learning rate of the alpha tensor.
 1: for each $i$ from 1 to $N$ do
 2:     Initialize local models $\theta_l^i$
 3:     Prepare local datasets $D_l^i$
 4: end for
 5: Initialize global models $\theta_g$
 6: initialize tensor $a$
 7: Prepare communal dataset $D_{cm}$
 8: Prepare common global dataset for evaluation $D_{test}$
 9: Server executes:
10: for each Federated round $j = 0, 1, 2, \ldots$ do
11:     for each client in parallel do
12:         $\theta_{l_j} \leftarrow \text{ClientUpdate}(D_l, \theta_{l_j})$
13:         $z_j, .. \leftarrow$ Extract representations from $(\theta_{l_j})$
14:         Send representation to server
15:     end for
16:     $a_j \leftarrow \text{AnchorAggregation}(D_{cm}, \theta_g, z_j, N, a, e_g, a_e ta)$
17:     Distribute the updated global Anchors
18:     for each client in parallel do
19:         $\theta_{l_{j+1}} \leftarrow \text{KDUpdate}(D_{ex}, \theta_{l_j}, a)$
20:     end for
21: end for
22: Validate the updated distillated models on $D_{test}$
```

Algorithm 10 ClientUpdate Function

```
1: ClientUpdate($D_l$, $\theta$):                              ▷ Run on specific client
2: for each local epoch $i$ from 1 to $E$ do
3:     for each batch in $D_l$ do
4:         $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l, \text{labels})$      ▷ Update the client model with Eq. 6.1
5:     end for
6: end for
7: return $\theta_l$
```

Algorithm 11 DistillationUpdate Function

```
1: KDUpdate($D_{ex}$, $\theta_l$, $a$):                        ▷ Run on specific client
2: for each batch in $D_{ex}$ do
3:     $l_{mul}$                          ▷ Compute Multi Scale loss with Eq. 6.3
4:     $l_{dis}$                          ▷ Compute Distillation loss with Eq. 6.2
5:     $L = m_c * l_{mul} + l_{dis}$
6:     $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l)$     ▷ Update the client model with aggregated loss $L$
7: end for
8: return $\theta_l$
```

Algorithm 12 AnchorAggregation Function

```
1: ClientUpdate($D_cm, \theta_g, z_j, N, a, e_g, a_\eta$):
2: for each $i$ from 1 to $N - 1$ do
3:     for each $j$ from 1 to $e_g$ do
4:         $\theta_g \leftarrow$ ClientUpdate($D_{ex}, \theta_g$) $o_g \leftarrow theta_g$
5:         $LA$                   ▷ Compute linear interpolation with equation 6.4
6:         $a \leftarrow a - a_\eta \nabla L(a, theta_g)$      ▷ Update the $a$ tensor with Eq. 6.6
7:     end for
8: end for
9: return $a_j$
```

6.3 Experimental Setup and Results

6.3.1 Dataset settings

The experimental results of the proposed FedFMRL system, utilizing the CIFAR-10 and CIFAR-100 image classification datasets [123] as a benchmark, are reported in this section. These datasets were adapted to simulate both IID and non-IID scenarios, incorporating various FL parameters. In the IID setting, the data was balanced, ensuring an even distribution among clients. Conversely, in the Non-IID setup, we deliberately created a scenario with extreme data imbalance. Similarly to prior works [51], a concentration parameter *beta* is used to produce the Non-IID data partition among clients. A total number of 10 participating nodes were involved, with images distributed among them for experimentation.

IID and Non-IID Settings

As previously mentioned, the parameter $\alpha > 0$ plays a pivotal role in regulating the degree of identicalness among participants. In our experimentation, we examined various values of α, with $\alpha \to \infty$ signifying that all participants possess identical data distributions, and $\alpha \to 0$ indicating that each participant exclusively contains examples from a single class. Values ranging from 0.05 to 1 were selected for α to assess the performance of the proposed method across different levels of data heterogeneity. In the experimental setup, each node randomly sampled $\frac{1}{10}$ of training and validation data respectively, while the test set data were reserved for the final system evaluation, both at the global and local levels.

6.3.2 Implementation Details

Architecture and Parameters

The architecture of the proposed method is grounded in a distributed structure, employing the ResNet56 [124] model, for both the local clients and the server model. The local clients undergo training using a multi-loss approach introduced in [122], which involves three intermediate outputs corresponding to the outputs of three ResNet layers. The SGD [125] technique is employed as the optimization technique. The learning rate is set to 0.1, with a weight decay of $1e-3$, a batch size of 64 and m_c equals to 0.1. To ensure comprehensive learning and convergence, the local models are trained for 300 epochs. The server model is trained for 5 epochs on the communal dataset for each feature mixing step. The learning rate is set to 0.1, with a weight decay of $1e-5$, and c_n equal to 0.01. Regarding the learning rate of the a tensor, it is 0.1 for the first $frac23$ of the global epochs and then increased to 10 for the remaining training process. The DL models are implemented in Python 3.8 within the PyTorch (version 2.0) [126] environment. The code for this research will be made publicly available for reference and further exploration.

Federated setting

The training paradigm for the proposed FL system, comprising a central server and 10 local clients, encompasses 6 federated rounds. In each round, local clients are separately trained for 50 epochs, with no inter-client communication during training. The final evaluation of the local models is conducted using the test set of the CIFAR-10 and CIFAR-100 datasets. The training set is divided into a common set that is accessible to all clients, constituting 20% of the training set, and the remaining 80% is distributed among the clients. Each client retains 10% of its local data as its local validation set and uses the rest for its local training set. The training was executed on a single computer outfitted with 4 GeForce RTX 3090 (VRAM 24GB each) and furnished with 128GB of RAM.

Baselines

To comprehensively assess the proposed method's efficacy, a comparative performance analysis was conducted using two distinct configurations. Initially, benchmark tests were carried out using a FedAvg framework, encompassing both IID and non-IID scenarios.

Furthermore, the proposed method was compared with a scheme of locally isolated clients (with no communication between them) and a standard federated distillation strategy.

6.3.3 Performance Evaluation

Global aggregation schemes

In this section, the impact of various global aggregation schemes is evaluated. Three different methods are subjected to testing:

a) This method involves weighted averaging of the client outputs. For each client, the factor $(1 - a) * o_r^c$ (representing the model output of client c at the federated round r) is added to the soft logits tensor. The final soft logits tensor is then divided by the product of the number of clients and the a_cap

b) In this scheme, a linear interpolation is computed, but the o^* of the Eq. 6.4 defines the mean model output of all clients except client n. This is followed by division by the number of clients.

c) This method follows the process described in Section 6.2.3.

The results (as presented in Table 6.1) demonstrate that linear interpolation is a better method for representation aggregation compared to weighted averaging. The scheme (c) achieves better results in both the client and the server sides and has been chosen as the aggregation scheme for the subsequent experiments.

Scheme	Mean Client Accuracy	Global Model Accuracy
A	1.00	19.50
B	29.63	19.50
C	**29.83**	**20.88**

Table 6.1: Comparative evaluation of different global aggregation schemes.

Alpha calculation methods

The a tensor can be computed using a closed-form solution or by drawing values from a Gaussian distribution with a predefined mean and standard deviation for the tensor a. The closed-form solution introduced in [121] is employed in this regard. Alternately, in the case of Gaussian-generated a, the tensor can be employed without modifications, or the method described in Section 6.2.3 for a learnable a tensor can be utilized. As demonstrated in Table 6.2, a learnable a strategy proved to be more effective in capturing the diverse properties of each client and assigning the appropriate weight factor to each client. This strategy has been adopted for the remaining experiments.

Alpha calculation	Mean Client Accuracy	Global Model Accuracy
Closed-form	28.73	N/A
Gaussian generated	29.14	N/A
Gaussian generated with learnable a	**29.83**	**20.88**

Table 6.2: Experiments with different methods for alpha calculation.

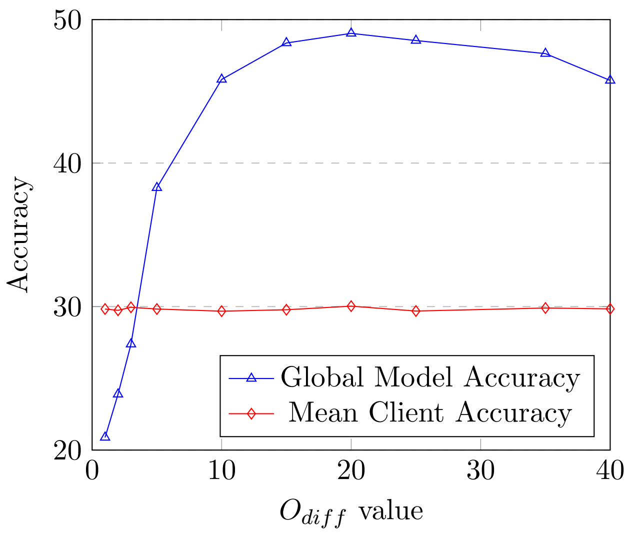


Figure 6.2: Impact of different values of the O_{diff} factor.

Impact of the O_{diff} factor

In this section, the impact of different values of the O_{diff} factor on the accuracy of the server's model and the clients' models is examined. According to Figure 6.2 the global model attains the highest accuracy when O_{diff} is set to 20. Thus, this value is employed in the subsequent experiments. It is observed that the client models are not significantly affected by the variation in the O_{diff} factor. This outcome is expected because the global model is directly influenced during training by O_{diff}, whereas the client models are indirectly affected due to the output modification of the global model.

Impact of the a_{cap} factor

This experiment addresses the effect of different values of a_{cap} on the local models' accuracy. The a_{cap} value determines the mean and standard deviation of the Gaussian distribution used to generate the initial a values. As it is illustrated in Figure 6.3, the best results are observed when a_{cap} is set to at least 0.5, with the optimal performance achieved when $a_{cap} = 0.8$. Therefore, this value has been selected for use in the subsequent experiments.

a_{cap}	Mean Client Accuracy
0.1	29.51
0.2	29.61
0.3	29.60
0.4	29.64
0.5	30.03
0.6	30.15
0.7	30.28
0.8	**30.41**
0.9	30.21

Table 6.3: Impact of different values of the a_{cap} factor.

Impact of non-IID data

Non-IID data often exhibits a wide range of patterns and variations among nodes, making it challenging to learn a generalized model. Therefore, to assess the impact of non-IID data on the proposed method, the mean client accuracy of the FedFMRL method (incorporating the multi-scale knowledge loss, as in Equation 6.3) is compared with the baseline federated distillation method for different values of beta, as shown in Fig. 6.4. The results demonstrate that the FedFMRL scheme outperforms the baseline federated distillation method across all beta values, highlighting the added value of the feature mixing and the multi-scale knowledge loss in handling non-IID data.

beta	0.25	0.5	0.75	1	IID
FLD	16.78	22.15	22.82	24.92	30.45
FedFMRL	**17.04**	**24.29**	**25.49**	**28.90**	**37.87**

Table 6.4: Experiments with Non-IID data, for different beta values

Comparison with baseline methods

In this section three baseline methods are compared with FedFMRL, as described in Section 6.3.2, using CIFAR-10 and CIFAR-100 datasets. In particular, on the CIFAR-100 dataset, FedFMRL achieves remarkable results with a Mean Client accuracy of 37.87%, surpassing the other methods. This is accompanied by an impressive Global Model accuracy of 49.04%, indicating the effectiveness of our approach. Similarly, on the CIFAR-10 dataset, FedFMRL outperforms the other methods with a Mean Client accuracy of 77.01% and a Global Model accuracy of 78.11%. It is worth noting that FedFMRL achieves superior results compared to FedAvg, even though only model outputs and intermediate representations are shared, and not model parameters. On the other hand, isolated clients who train separately exhibit the lowest accuracy, which was expected. However, this comparison underscores the value of the proposed approach.

Method	Mean Client Accuracy	Global Model Accuracy
CIFAR100		
FL(FedAvg)	N/A	34.17
Local Isolated Clients	23.29	N/A
FLD	30.45	N/A
FedFMRL	**37.87**	**49.04**
CIFAR10		
FL(FedAvg)	N/A	75.56
Local Isolated Clients	70.12	N/A
FLD	75.46	N/A
FedFMRL	**77.01**	**78.11**

Table 6.5: Comparative evaluation with baseline methods on CIFAR10 and CIFAR100 datasets.

6.4 Insights and Contributions

The study presented in this chapter represents a significant stride in this dissertation, particularly in addressing the challenges of representation learning and federated distillation. The introduction of a novel federated distillation weight aggregation method, tailored for distributed learning environments, marks a pivotal advancement. This approach effectively addresses the challenge of efficiently extracting and transferring knowledge across multiple nodes in a federated network, a critical issue in the realm of distributed data management. The core contribution of this study is the development of an algorithm that adeptly captures meaningful representations from various client nodes and models this information into actionable knowledge. The feature mixing technique used for knowledge consolidation at the central server is particularly innovative. It demonstrates an effective strategy for managing the complexity inherent in distributed learning systems. The comprehensive experiments conducted highlight the method's ability to maintain robust performance while significantly reducing communication costs, which is a notable achievement in the field.

Moreover, this study lays a foundational groundwork for exploring model architecture flexibility in FL, addressing the critical issue of data heterogeneity. By moving away from conventional model aggregation approaches and focusing on the exchange and aggregation of client model logits, this research not only advances more efficient and adaptable distributed learning systems but also sets the stage for an in-depth examination of model architecture flexibility in subsequent studies of this dissertation.

Chapter 7

Study 4: Incremental Learning and Knowledge Retention in FL

7.1 Overview of the Study

Following the establishment of an effective federated distillation weight aggregation method in Study 3, the fourth study in this series extends the exploration further into the realm of incremental learning within the field of computer vision. This study proposes a comprehensive approach to tackle the challenge of catastrophic forgetting—a critical issue in continual learning scenarios. Building upon the foundational concepts of meaningful representation extraction and knowledge modeling, this research introduces an innovative FL algorithm that is specifically designed to address the complexities of incremental learning. This algorithm stands out by eliminating the need for central model transfer, which is a common bottleneck in traditional FL systems.

Central to this study is the implementation of multi-scale representation learning, seamlessly integrated with Knowledge Distillation techniques. This integration is pivotal in facilitating the continual learning process, where the algorithm adeptly combines new knowledge with previously acquired information, thus preserving the continuity of learning across different states. The study also innovatively adapts a contrastive learning technique, further enriching the learning process by effectively amalgamating existing knowledge with new insights.

Extensive experimentation underpins this study, providing a thorough evaluation of the proposed methodology. The results of these experiments are compelling, demonstrating the method's significant potential in a FL context. Notably, the study achieves remarkable results in terms of reducing communication costs and ensuring robust performance across highly distributed incremental learning scenarios. This advancement, therefore, represents a significant stride in the evolution of FL, addressing one of the most pressing challenges in the field and paving the way for more dynamic and resilient learning models in distributed environments.

7.1.1 Challenges and Solutions in Incremental Learning

The ever-increasing number of smart devices leads to a rapid expansion in the amount of data being exchanged, presenting a substantial challenge, especially when handling distributed data. This becomes increasingly challenging when new labeled or unlabeled data are continuously introduced to the system. As a result, the demand for systems that can effectively exploit these data to adapt to new conditions, *i.e.* tasks, while minimizing costs becomes even more pronounced. Several researchers dedicated resources to find reliable solutions [127], but despite the vast amount of published research on the topic, the challenge of learning new knowledge without forgetting, particularly in the context of distributed datasets, still poses significant challenges.

In the realm of computer vision, to achieve truly robust recognition performance and maintain the capacity to recognize previously seen entities (*e.g.* objects or scenes) when encountering new classes or datasets, it is crucial to tackle several pivotal challenges. These include managing data heterogeneity and model bias effect, communication and synchronization, privacy and security concerns, as well as ensuring scalability and efficiency in processing and training distributed datasets to accommodate the increased complexity [80]. To tackle these, research efforts primarily concentrated on utilizing combinations of regularization techniques and data synthesis methods to mitigate or alleviate catastrophic forgetting, while network expansion techniques and rehearsal-based approaches were explored to expand network capacity and prioritize samples based on their importance for learning, respectively [128]. Recent advancements in lifelong learning approaches [129, 130, 131] have gained attention in incremental learning computer vision research, reshaping the way visual analysis is conducted. These techniques enable to learn generic learning patterns that can adapt to new tasks and data, allowing the model to continually improve its learning efficiency and adaptability. By leveraging knowledge from previous tasks to facilitate learning new ones, these methods enable more robust and coherent incremental learning.

While there has been extensive research on applying incremental learning to centralized data settings, the exploration of its application to federated datasets and tasks, is still relatively limited, due to data privacy concerns, system heterogeneity constraints, dynamic data distribution, *etc.* Under a realistic scenario, end-users, or the nodes of the system, constantly generate new data that neither necessarily belong to previously known categories nor follow a specific distribution. In an attempt to understand the problem and train a general model, older versions of models and data are maintained, which constantly push the limits of system requirements, inevitably leading to a sudden significant decrease in their performance in previous tasks. Research groups have devoted resources to approaches specifically designed for federated settings, leveraging the power of incremental learning while respecting the decentralized nature of data. These approaches often involve a range of learning techniques, such as knowledge distillation, life-long learning, and hybrid approaches to achieve better performance in preserving previous knowledge and learning new knowledge incrementally [132, 133, 134, 135].

Through the application of a combination of techniques such as representation learning, knowledge distillation, and contrastive learning (CLR) in our approach, it becomes feasible to address the challenges posed by dynamic, heterogeneous, and fragmented data. In a more specific context, the generation of representations that encompass knowledge from diverse levels of the model results in a robust feature extractor with enhanced capabilities. When combined with the Knowledge Distillation approach, this integration facilitates effective generalization at a federated level, allowing for broader applicability and improved performance. For incarnating the task incremental step, we leveraged CLR [136] to distinguish optimal representations by contrasting both global and local representation samples. The proposed methodology aimed to enhance the overall quality of representations by creating a latent space where global representations captured shared knowledge across different tasks, while task-specific representations were well-separated, enabling effective discrimination and understanding of individual tasks.

The main contributions of this work are summarized as follows:

(a) **A novel Federated Incremental Learning scheme is introduced** that utilizes multi-level representation learning coupled with rehearsal-based knowledge distillation approaches that support FL optimization algorithms and CRL techniques in an end-to-end learning manner. The holistic nature of our approach ensures a robust and scalable solution for incremental learning in federated environments.

(b) **A rehearsal-based knowledge distillation technique to transfer this enriched knowledge to neighboring nodes.** Through knowledge distillation, the learned insights and patterns from the well-mixed representations are effectively transferred and shared across the federated network.

(c) **A contrastive-based learning algorithm that utilizes mixed representations to retain the knowledge gained from previous tasks of the incremental scenario.** By combining features from different tasks, our algorithm encourages the model to learn and preserve valuable information across different task domains.

(d) **Validate the effectiveness of the proposed approach through extensive comparisons on one of the broadest publicly available benchmarks,** demonstrating its superiority and robustness across a broad spectrum of FL settings.

7.2 Methodology and Incremental Learning Techniques

7.2.1 Problem statement

In a federated system, the data are inherently localized to individual clients, and sharing them with other clients is strictly prohibited. The objective of this study is to

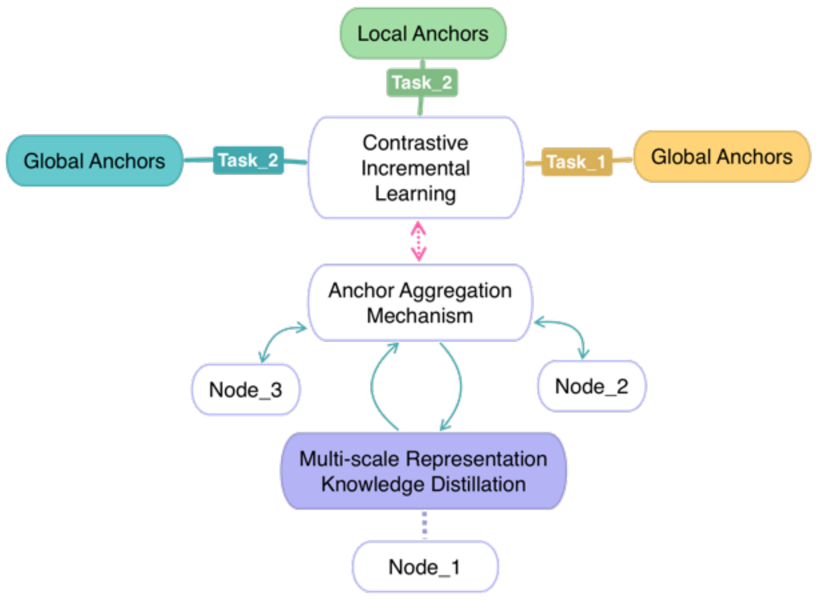

Local Anchors
Task_2
Global Anchors
Task_2
Contrastive Incremental Learning
Task_1
Global Anchors
Anchor Aggregation Mechanism
Node_3
Node_2
Multi-scale Representation Knowledge Distillation
Node_1

Figure 7.1: The proposed FL scheme utilizes a centralized server to create and share the aggregated global representations, while several local nodes participate asynchronously in the training process.

learn from a continuously evolving stream of data that introduces new classes in highly distributed environments. The data are divided into a sequence of non-overlapping training tasks, each representing a step of incremental learning. The final goal is to continually develop a classification model that incorporates knowledge from both current and previous tasks across various federated nodes. After each task, the model's performance is evaluated on all the classes encountered so far, which is the union of classes from all previous tasks.

The proposed methodology involves a federated incremental learning scheme that combines multi-level representation learning, knowledge distillation approaches coupled with FL optimization algorithms, and CRL techniques to address the challenges of evolving data domains, as depicted in Figure 7.1. For the cultivation of individual local clients, the CIFAR-100[123] dataset is deployed, and partitioned in both a balanced and unbalanced manner to facilitate experiments that cater to both IID and Non-IID conditions.

This work follows the common practice of using exemplar sets for incremental learning in computer vision. Exemplars are representative instances of known classes, selected from the training set. Instead of random sampling, the herding strategy is employed to choose the most representative exemplars for each class. Mean anchor images from the previous rounds are computed, and the exemplar set is formed by selecting class images that approximate these mean anchors [137].

7.2.2 Federated Knowledge Distillation

Local Supervision Representation Learning

The local clients undergo supervised training using the CIFAR-100 dataset, with each client being assigned a specific segment of the dataset. This assignment remains consistent throughout all federated rounds, and the clients receive training on their allocated segments. The training subset of the dataset is utilized for model training, while the validation subset is used to evaluate and authenticate the performance of each client's trained models. Apart from each client's training and validation set, a communal dataset exists that is accessible to all clients. To train the clients' networks, a classification head was added to map the feature vectors to the total number of classes in the dataset; 100 in the case of CIFAR-100. The Cross-Entropy was deployed as the loss function for the training process, and its formula is shown below:

$$L(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} y_{mk}\log p_{mk} \tag{7.1}$$

where M is the total number of images, K the total number of classes, y_{mk} the label of the m, k image and p_{mk} the probability of the class. The stochastic gradient descent (SGD) [138] algorithm was chosen as the optimization strategy.

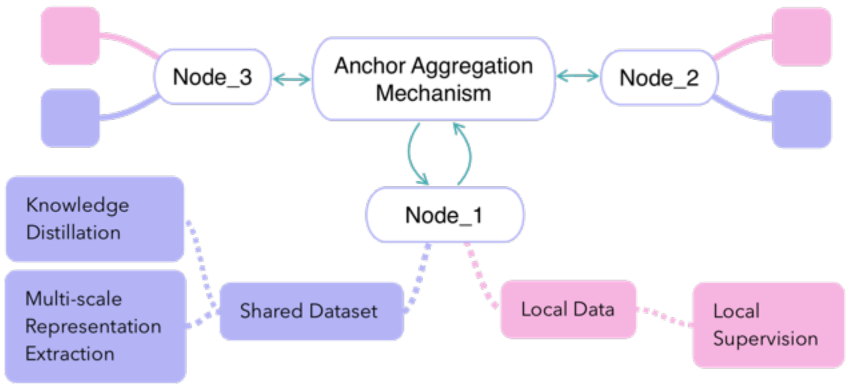


Figure 7.2: The proposed Federated Multi-scale Representation Knowledge Distillation scheme.

One drawback of the federated distillation scheme is the small amount of information transmitted from the clients to the data since only the model's output is transmitted to the central server (*i.e.* last fully-connected layer). In order to mitigate this drawback, a scheme of multi-scale knowledge distillation is adopted inspired by the work of

[139]. More specifically, apart from the model's output, each client transmits some intermediate outputs (*i.e.* hidden representation), where the number and the position of the intermediate outputs can vary depending on the problem.

Different-level features can be extracted, let $z_l^N = f_l^N(\cdot), l \in (1, ..., L)$, where l are the different model layers, z_l^N is the intermediate output at layer l for the client N, with size $H * W * C$ and $f_l^N(\cdot)$ is the subnetwork for feature extraction up to the layer l. Then, the final form of the feature map can be obtained by concatenating the $W * C$ height-pooled slices and the $H * C$ width-pooled slices for h_l^N:

$$\Phi(z_{l_{j-1}}^N) = \left[\frac{1}{W} \sum_{w=1}^{W} z_{l_{j-1}}^N[:, w, :] \middle|\middle| \frac{1}{W} \sum_{h=1}^{H} z_{l_{j-1}}^N[h, :, :] \right] \tag{7.2}$$

where $[\cdot||\cdot]$ denotes concatenation over the channel axis, N is the number of clients, j is the federated round and $\Phi(\cdot)$ is the network function. The intermediate outputs are calculated over the communal dataset, therefore all clients extract features from the same subset. After all intermediate outputs of all clients are extracted, they are averaged per client, $\Phi(z_l) = \frac{1}{N} \sum_{n=1}^{N} \Phi(z_{l_{r-1}}^N)$. At the beginning of the next federated round, for each client, the intermediate outputs are employed as soft labels to minimize the Euclidean distance between the mean averaged representation of the clients and each client's local representation. The multi-scale knowledge distillation term is calculated as:

$$\mathcal{L}_{multi-scale_{KL}} = \frac{1}{L} \sum_{l=1}^{L} \left| |\Phi(z_{l_j}^N) - \Phi(z_{l_{j-1}}) \right| | \tag{7.3}$$

Global Distilled Supervision

As previously mentioned, a communal dataset is available and accessible to all clients, following the common practice in the literature ([117], [140], [119]). This communal dataset serves as a reference point to address data heterogeneity among the clients in the FL scheme. At the end of each federated round, every client generates model outputs for each sample in the common dataset. These outputs are then transmitted to the server side and averaged across all clients (illustrated in Figure 7.2), similar to the approach described in 7.2.2. In the subsequent federated round, each client performs knowledge distillation using the global anchors derived from the common dataset, thus, ensuring knowledge alignment between the clients. The knowledge distillation term is calculated as follows:

$$\mathcal{L}_{KL} = \frac{1}{L} \sum_{l=1}^{L} ||\Phi(o_j^N) - \Phi(o_{j-1})|| \tag{7.4}$$

where o_r^c is the model output of client N at the federated round j.

7.2.3 Incremental Learning

The proposed federate incremental scheme draws inspiration from MOON [135], which is a simple and effective approach based on FedAvg with lightweight modifications in the local training phase. However, there are significant differences in our approach. We focus solely on representation learning at each federated round using knowledge distillation. Our approach involves extracting stronger representations through a robust aggregation algorithm. In contrast to MOON, we incorporate CRL, which is typically used for visual representations, on the aggregated global anchors and their respective local versions (as presented in Figure 7.1). This allows us to decrease the distance between the representation learned by the local model and the global aggregated representation of the previous task(s), while increasing the distance between the representation learned by the local model and the global aggregated representation of the previous federated round of the same task. By incorporating these elements, our approach enhances the learning procedure and leverages the power of CRL in the federated setting. Moreover, the global aggregated representation of all previous tasks is employed to further enhance the learning procedure.

In the federated incremental architecture, the primary server undergoes training via a contrastive approach on the dataset $X_m = (x_m^i, y_m^i), i = 1, ...N$, wherein the labels are disregarded. Let us assume that node N_i is performing the local training. Each image from the said dataset serves as the anchor image x_m^i. All of these images are subsequently passed through the network function $\Phi(\cdot)$, and are then projected into the latent space for the implementation of CLR as delineated in [136]. During the initialization phase, each node receives the global aggregated anchors a_t from the server, which are common for all clients at the current round. During this process, for each input image x_m^i, we extract the representation of x_m^i, following the federated knowledge distillation approach defined above, from the current global aggregated anchors from the previous federated round a_t^{j-1} as $z_t^{j-1} = \Phi a_t^{j-1}(x_m^i)$, the representation of x_m^i from the aggregated global anchors of the previous incremental round a_{t-1} as $z_{t-1} = \Phi a_{t-1}(x_m^i)$, and the representation of x_m^i from the current local anchor being updated a_t^j as $z_t^j = \Phi a_t(x_m^i)$. The feature maps are harnessed for the calculation of the contrastive loss. The formula for the loss is indicated below:

$$a_{con} = -\log \frac{\exp\left(sim(z_t^j, z_{t-1})/T\right)}{\exp\left(sim(z_t^j, z_{t-1})/T\right) + \exp\left(sim(z_t^j, z_t^{j-1})/T\right)} \tag{7.5}$$

where $\mathcal{T}$ is the temperature that is a scaling factor used to control the concentration of the output distribution, affecting the hardness of the positives in the CLR framework. Since the global model is expected to generate better representations, our objective is to minimize the distance between z_t^j and z_t^{j-1}, indicating that the local model aligns with the previous incremental round's global anchor's representations. Additionally, we aim to maximize the distance between z_t^j and z_{t-1}, indicating that the updated local anchors are diverging from the previous round's global anchor's representations and retaining their own learned representations. Through the implementation of contrastive loss, the network strives to learn representations that induce proximity between the local image and the positive anchor global image in the embedding space, whilst ensuring a separation between the local image and the previous round's global anchor image. The representations learned in this manner would have the capacity to segment the latent space in accordance with the context, without the true comprehension of the task.

7.2.4 FedRCIL Algorithm

In this section, the complete flow of the described procedure is presented. The entire process is divided into three main algorithms, the Algorithm 13 describes the required steps for training and updating the network in the main server, while the other two define the actions for training and updating the network in each local client (*i.e.* Algorithm 14 for local and Algorithm 15 for global supervision respectively).

7.3 Dealing with Knowledge Retention Challenges

7.3.1 Knowledge distillation

Knowledge distillation is a technique that aims to efficiently transfer information from a large model (known as the teacher model) to a smaller one (known as the student model). The teacher model guides the student model to achieve a better performance through an iterative learning process. Knowledge distillation is widely used for model deployment on resource-constrained devices. The concept of distillation learning is introduced in [141], using soft outputs of the teacher model in order to guide the training of the smaller model. Moreover, a distillation loss is introduced, combined with the cross entropy loss to strike a balance between data fitting and mimicking the teacher. In most recent works, Zhao *et al.* [142] divides the knowledge distillation into two parts, namely the target and the non-target. The target knowledge distillation part is a binary logit distillation for the target class and the non-target knowledge distillation part is a multi-category logit distillation for non-target classes. In [143], the deviation between the predictions of the teacher and the student model is addressed. A correlation-based loss is introduced to capture inter-class and intra-class relations from the teacher. The technique of knowledge distillation has also extended to the field of FL as described in 7.3.3

7.3.2 Incremental learning

The Incremental Learning problem has been the focus of various studies, encompassing different granular settings. Typically, it can be broadly categorized into three types, depending on the specific characteristics of the incremental scenario, namely task-incremental learning, class-incremental learning, and domain-incremental learning [144]. The first two share a similar setting in which new classes are introduced in new

Algorithm 13 FedRCIL Algorithm

```
Require: $T$ is the number of communication rounds, $\mathcal{N}$ is the total number of clients, $\theta_l^i$ represents the parameters of the local models, $\mathcal{D}_l^i$ are the separate datasets for the local clients, $\mathcal{D}_{ex}$ is the dataset of the exemplar set, $\mathcal{D}_{test}$ is the common testing dataset, and $\eta$ is the learning rate.
 1: for each $i$ from 1 to $N$ do
 2:     Initialize local models $\theta_l^i$
 3:     Prepare local datasets $D_l^i$
 4: end for
 5: Prepare exemplar set $D_{ex}$
 6: Prepare common global dataset for evaluation $D_{test}$
 7: Server executes:
 8: for each Incremental round $t = 0, 1, 2, \ldots$ do
 9:     for each Federated round $j = 0, 1, 2, \ldots$ do
10:         for each client in parallel do
11:             $\theta_{l_j} \leftarrow \text{ClientUpdate}(D_l, \theta_{l_j})$
12:             $z_{t_j}, .. \leftarrow$ Extract representations from $(\theta_{l_j})$
13:             Send representation to server
14:         end for
15:         $a_{t_j} \leftarrow$ Aggregate Anchor representations
16:         Distribute the updated global Anchors
17:         for each client in parallel do
18:             $\theta_{l_{j+1}} \leftarrow \text{KDUpdate}(D_{ex}, \theta_{l_j}, a, m_u = 0.5)$
19:         end for
20:     end for
21:     Validate the updated distillated models on $D_{test}$
22: end for
```

Algorithm 14 ClientUpdate Function

```
1: ClientUpdate($D_l$, $\theta$):                              ▷ Run on specific client
2: for each local epoch $i$ from 1 to $E$ do
3:     for each batch in $D_l$ do
4:         $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l, \text{labels})$      ▷ Update the client model with Eq. 6.1
5:     end for
6: end for
7: return $\theta_l$
```

Algorithm 15 DistillationUpdate Function

```
1: KDUpdate($D_{ex}$, $\theta_l$, $a$, $m_u$):                ▷ Run on specific client
2: for each batch in $D_{ex}$ do
3:     $l_{mul}$                      ▷ Compute Multi Scale loss with Eq. 6.3
4:     $l_{dis}$                      ▷ Compute Distillation loss with Eq. 6.2
5:     $l_{con}$                      ▷ Compute constrastive loss with Eq. 7.5
6:     $L = m_c * l_{mul} + l_{dis} + m_u * l_{con}$
7:     $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l)$     ▷ Update the client model with aggregated loss $L$
8: end for
9: return $\theta_l$
```

tasks. However, the key distinction between them lies in the inference stage. Unlike other incremental learning scenarios, where new tasks introduce new classes, domain-incremental learning addresses the challenge of adapting to shifts in data distribution or domains while preserving the existing label space. Each scenario presents unique challenges, such as avoiding catastrophic forgetting, handling domain shifts, or managing imbalanced data distributions, and requires tailored approaches to ensure effective continual learning [145]. Therefore, research on incremental learning covers a wide range of approaches, including regularization-based methods, rehearsal and replay techniques, knowledge distillation, network expansion, and hybrid approaches that combine multiple strategies [146, 147, 148, 149, 150, 151, 152].

CRL serves as a potent instrument within the realm of incremental learning. Its fundamental aim revolves around crafting representations that induce a tendency for similar instances to congregate in the embedding space (*i.e.* positive pairs), while simultaneously propelling dissimilar instances to be dispersed at a greater distance (*i.e.* negative pairs). Inspired by the typical CRL framework in [136], recent breakthroughs in incremental learning argue that contrastively learned representations are robust against the catastrophic forgetting [153, 129, 154, 130, 131], and could be transferred better to unseen tasks.

In recent studies, several approaches have been proposed to address the problem of catastrophic forgetting in different domains. Cha *et al.* in [129] propose a rehearsal-based continual learning algorithm that utilizes CRL and self-supervised distillation to learn and maintain transferable representations, leading to improved performance in image classification tasks. In a similar attempt, authors in [154], utilize instance-level and class-level contrastive losses, along with knowledge distillation and a spatial group-wise enhanced attention mechanism, to maintain the inner-class assignment information and alleviate catastrophic forgetting. A novel incremental learning framework is proposed in [153], which utilizes contrastive one-class classifiers to address catastrophic forgetting in class incremental learning. Additionally, [131] extends contrastive self-supervised learning to be primarily based on exemplars and applicable to both labeled and unlabeled data, enabling few-shot class incremental learning.

This theory paves the way for an incremental learning task, wherein the network endeavors to diminish the distance between the current and the previous task instances in the latent space, while concurrently increasing the space between all the other instances within the dataset, effectively achieving a balance between stability and plasticity. In line with this concept, the network aims to grasp the task-specific representation of the instances in a way that improves their distinguishability in the embedding space. This facilitates the identification and differentiation of individual instances, making the overall process simpler.

7.3.3 Federated Learning Distillation

In contrast to traditional centralized ML techniques, FL employs a training approach where an algorithm is trained through multiple independent sessions, with each session using its own distinct dataset. FL enables multiple actors to collaboratively train a shared and resilient ML model without the need to centralize their respective data. Through this approach, FL effectively addresses concerns related to data privacy, security, and authorization while allowing for the utilization of diverse and heterogeneous data sources.

Initially, the research on FL has primarily concentrated on enhancing communication efficiency and expediting model updates. The groundbreaking work by McMahan *et al.* [155] introduces a novel concept of averaging local stochastic gradient descent updates (known as FedAvg) to increase the overall amount of information used by each client during communication rounds. To overcome challenges like low device participation and non-independent and identically distributed (Non-IID) local data, several studies have explored the use of online knowledge distillation approaches. A novel approach for federated multi-task distillation is introduced in [156], while Wu *et al.* [157] presented a communication-efficient FL method, utilizing adaptive mutual knowledge distillation and dynamic gradient compression to reduce communication costs. Similarly, Li *et al.* [158] introduces a unified algorithmic framework for Federated Distillation (FD), employing active data sampling to reduce communication overhead.

Towards this direction, a recently introduced technique has emerged, named FLD (Federated Learning Distillation), which takes a different approach, by exchanging and aggregating client model outputs. This alternative methodology offers distinct advantages such as reduced communication costs, flexibility in model architecture, and improved handling of non-IID data distribution. Jeong *et al.* [116] propose an FLD scheme, where clients upload per-label averaged soft targets to train a conditional Generative Adversarial Network (GAN). Li *et al.* [117] introduce a common dataset accessible to all clients, training them on the public dataset before their private data. Gong *et al.* [120] introduce one-shot distillation where client models are fully trained and then distilled to the server using attention maps per class. These approaches combine distillation and aggregation mechanisms to facilitate FL, while also considering the integration of public or shared datasets.

Dealing with catastrophic forgetting in FL poses unique challenges that have not been extensively explored compared to other learning settings. To address this gap, researchers have proposed a set of models [132, 133, 134] specifically designed for federated scenarios. These models incorporate techniques such as class-aware gradient compensation, semantic distillation, adaptive class-balanced pseudo labeling, and forgetting-balanced semantic compensation. Additionally, a recent study introduced a local model contrastive loss [135] to enhance individual party training. These approaches effectively mitigate forgetting and provide solutions for catastrophic forgetting in the FL context.

While there has been notable progress in Federated Incremental Learning, there is still untapped potential in exploring strategies that leverage the best solutions from the literature on knowledge distillation and incremental learning and adapt them to the federated setting. The existing research presents opportunities to develop novel techniques that effectively address challenges in that field.

7.4 Experimental Results and Analysis

7.4.1 Dataset settings

In this section, experimental results from the application of the proposed FedRCIL scheme, using the CIFAR-100 image classification dataset as a benchmark, are presented. The CIFAR-100 dataset was adapted to represent both IID and Non-IID scenarios, capturing different FL settings. In the IID setup, the data was balanced, while the Non-IID setup represented an extreme case of data imbalance. Like previous studies [51], Dirichlet distribution is utilized to generate the Non-IID data partition among parties, using a concentration parameter β. Both datasets consisted of 100 categories, and a total of 10 participating nodes were involved, with images distributed among them.

7.4.2 Implementation Details

Architecture and Parameters

The architecture of the proposed method employs a distributed framework, where each local client utilizes ResNet56 [124]. The ResNet56 model is chosen for its fast training and satisfactory results. At each local client, the final fully connected layer of the network is replaced with a projection head to facilitate the Incremental CRL task, allowing the transition of ResNet feature maps to a new embedding space that supports the CLR scheme. Meanwhile, the local clients undergo supervised training using the proposed multi-loss scheme, involving three intermediate outputs corresponding to the outputs of three ResNet layers. The SGD method is employed with a learning rate of 0.1, the batch size of the system is 64, the m_u and $\mathcal{T}$ are set equal to 0.5 and m_c equals to 0.1. The local models are trained for 300 epochs to ensure comprehensive learning and convergence. Python 3.7 and PyTorch (version 1.7.0) environments are employed for the implementation of the DL models. The code is available at `https://github.com/chatzikon/FedRCIL`.

Federated setting

The training paradigm for the suggested FL system which comprises a centralized server and 10 local clients, involves 6 federated rounds without incremental learning. In each round, the local clients individually undergo training for 50 epochs without any inter-client communication. The final evaluation of the local models is performed using the test set of the CIFAR-100 dataset. The training set is divided into a common set accessible to all clients (20% of the train set), while the remaining 80% is divided among the clients. In the case of incremental learning, the epochs without any inter-client communication are 10 and the federated rounds are 30. The incremental learning process consists of 5 different tasks, each one with 20 unique classes. Each task has a training period of 6 federated rounds that constitute an incremental round. The common set mentioned above is employed as an exemplar set, with a steady size but with a varying number of samples per class (*i.e.* as new tasks arrive, fewer samples per class exist), employing the approach mentioned in Section 7.2.1.

Baselines

To validate the effectiveness of FedRCIL comprehensively, a comparative performance analysis was conducted on two distinct configurations. Initially, benchmark experiments were undertaken within a fully-supervised FedAvg framework, in both IID and Non-IID scenarios. Furthermore, the proposed approach was compared with a scheme of local isolated clients (without communication among them) and with a baseline federated distillation approach.

7.4.3 Performance Evaluation

This section provides a summary of the results obtained through the application of the proposed scheme in several distinct scenarios under various learning settings. In an attempt to showcase the distinctive characteristics of our architecture, we conducted direct comparisons with the proposed baseline methods, depicted in Table 7.2. However, it was not possible to compare with other methods from the literature, except for FedAvg, due to the difficulty of adapting state-of-the-art techniques to the specific problem we are addressing.

Representation learning setting

The accuracy results presented in Table 7.1 investigate various concepts associated with extracting multi-level and multi-scale representations from local clients. It is shown, that the FLD_{m_B}, where each extra loss is applied to the part of the model before it, back-propagating with layer 1 loss first and layer 3 loss last, achieved the highest accuracy of 38.06%. Similarly, FLD_{m_C}, with losses back-propagating with layer 3 loss first and layer 1 loss last, obtained a slightly lower accuracy of 37.82%. On the other hand, FLD_{m_A}, where all losses apply to the whole network, resulted in the lowest accuracy of 32.03%. These findings highlight the significance of selectively applying additional losses to specific parts of the model and the importance of the direction of back-propagation. The results clearly demonstrate that the proposed method outperforms the conventional cross-entropy loss approaches.

Multi-loss concept	Accuracy
FLD	28.41
FLD_{m_A}	32.03
FLD_{m_B}	38.06
FLD_{m_C}	37.82

Table 7.1: FLD_{m_A}: All losses apply to the whole network FLD_{m_B} : Each extra loss apply to the part of the model before it, backprop from layer 1 to layer 3 FLD_{m_C} : Each extra loss apply to the part of the model before it, backprop from layer 3 to layer 1

Federated distillation learning setting

In relation to a comparison with the baseline methods, the proposed approach, achieved the highest accuracy of 38.06%, suggesting that leveraging a shared dataset and optimizing multiple objectives simultaneously enhances the learning process. On the other hand, isolated clients where each client trains independently, had the lowest accuracy of 23.29%, of course, this was something we expected, but it highlights the added value of the proposed approach. FL without a common dataset also performed relatively poorly, with an accuracy of 20.61%. Concerning the investigated FLD schemes, the findings suggest choice of layer output, and the presence of a common dataset can influence the effectiveness of distillation. As discernible from Table 7.2, when juxtaposed with the FedAvg, the proposed approach provides superior results with relative improvement over the baseline of 11.39%. These findings highlight the importance of collaboration and the potential benefits of utilizing shared data and optimized loss functions in FL, ultimately leading to improved accuracy in image classification tasks.

Method	Common set	Accuracy
$FL(FedAvg)$		34.17
Local Isolated Clients		23.29
FLD_l		20.61
FLD_{l-1}	✓	19.71
FLD_l	✓	28.41
$FedRCIL$ (Proposed)	✓	38.06

Table 7.2: Comparative evaluation with baseline methods and various distillation schemes.

Incremental learning setting

Table 7.3 presents accuracy results for the proposed method at different m_u settings, which represent the weight of the contrastive loss in the learning process. Utilization of the contrastive learning mechanism, results in significantly higher accuracy, contrary to the case that m_u equals to 0, resulting in a significant drop in accuracy. In particular, the results indicate that the accuracy decreased by around 20% when moving from 1 task to 5 tasks. This significant drop highlights the superior added value of the proposed incremental mechanism. On the other hand, in the other three cases where m_u values are higher (*i.e.* $0.25, 0.5$, and 1), the results remain favourably comparable.

m_u	task=1	task=5
$FedRCIL^{m_u=0}$	29.54	10.96
$FedRCIL^{m_u=0.25}$		26.07
$FedRCIL^{m_u=0.5}$		26.79
$FedRCIL^{m_u=1}$		25.42

Table 7.3: Experiments with different values for the contrastive loss coefficient m_u.

In an attempt to investigate the impact of knowledge retention on performance, we intentionally increased the number of buffers per task progressively from 1 to 4, as depicted in Table 7.4. Keeping only the most recent task instance (buffer size $b = 1$) yields the lowest accuracy of 15.76%, indicating its inferior performance. However, as the buffer size increases, accuracy increases significantly, with buffer size $b = 4$ resulting in 26.79%. Buffer size values of 2 and 3 strike a balance between retaining knowledge and model performance, achieving accuracies of 19.52% and 21.12%, respectively. These findings emphasize the importance of managing buffer size in incremental learning; storing an adequate number of previous task instances can positively impact performance while storing only the most recent one results in the worst accuracy. It is evident that balancing the buffer size is crucial for enhancing knowledge transferability and optimizing the proposed method's effectiveness in incremental learning settings.

Buffer	Accuracy
$FedRCIL^{b=1}$	15.76
$FedRCIL^{b=2}$	19.52
$FedRCIL^{b=3}$	21.12
$FedRCIL^{b=4}$	26.79

Table 7.4: Experiments with different buffer size (number of previous task models employed as positives at the contrastive learning).

Non-IID data typically contain diverse patterns and variations across different nodes, which inherently lead to challenges in learning a generalisable model. In that context, Table 6.4 presents accuracy results for three different methods: FLD_c, FLD_{mc}, and the proposed $FedRCIL$, at various beta values ($0.25, 0.5, 0.75$, and 1), which control the non-iidness of the dataset. As beta increases, all methods generally show improved accuracy. The basic FLD_c method achieves the lowest accuracy, ranging from 16.45% ($beta = 0.25$) to 22.10% ($beta = 1$). Introducing multi-loss (FLD_{mc}) leads to higher accuracy across all beta values, ranging from 18.48% ($beta = 0.25$) to 28.79% ($beta = 1$). Notably, the proposed incremental $FedRCIL$ scheme demonstrates promising performance, achieving comparable results of 16.62% with an acceptable decrease of around 10% compared to the IID case. This outcome is particularly impressive considering the challenging evaluation setting involved in incremental learning tasks.

$beta$	0.25	0.5	0.75	1	IID
FLD_c	16.45	20.01	20.64	22.10	28.41
FLD_{mc}	18.48	24.07	25.86	28.79	38.06
$FedRCIL$	10.22	13.29	14.82	16.62	26.79

Table 7.5: Experiments with Non-IID data, for different *beta* values. Methods indicated with subscript 'm', and 'c' utilize multi-loss, and common dataset, respectively.

7.5 Insights and Contributions

This work presented a holistic approach to mitigate catastrophic forgetting and maximizing knowledge retention in computer vision during incremental learning, by integrating CRL, FL and rehearsal-based knowledge distillation techniques. Central to this study's contribution is the innovative FL algorithm tailored for incremental learning scenarios. This algorithm is adept at handling catastrophic forgetting, a common hurdle in continual learning, through the integration of multi-scale representation learning and Knowledge Distillation techniques. This integration not only facilitates continual learning but also enhances the quality and relevance of the knowledge transferred across the federated network. Additionally, the study's adaptation of CRL techniques enriches the learning process by combining existing knowledge with new insights, thereby preserving the continuity of learning across different states. The experimental results validate the method's potential in a FL context. The approach demonstrates significant potential in reducing communication costs and ensuring robust performance across highly distributed incremental learning scenarios. These results highlight the effectiveness of the proposed method in managing and leveraging decentralized data in federated environments. Furthermore, the study effectively addresses several key challenges in incremental learning, such as managing data heterogeneity, model bias effect, and ensuring scalability and efficiency in processing and training distributed datasets.

Chapter 8

Study 5: Representation learning with limited data in FL

8.1 Overview of the Study

Building on the insights from the previous studies, particularly the advancements in federated distillation and incremental learning, the fifth study embarks on a path to resolve the challenges posed by limited and unevenly distributed data in FL environments. This study introduces a pioneering approach that ingeniously combines self-supervised and supervised learning techniques within a federated framework. This hybrid methodology is meticulously tailored to accommodate the unique requirements of various federated scenarios, particularly addressing the complexities associated with sparse and imbalanced data distributions across different nodes.

At the heart of this study is a custom self-supervised learning strategy, deployed at the global level, to harness the untapped potential of unlabeled data efficiently. This approach is complemented by supervised learning techniques applied at the local level, enabling the effective utilization of available labeled data. The synergy of these methods results in a robust and versatile learning mechanism that excels in diverse federated settings.

The extensive experimentation conducted as part of this study provides a comprehensive evaluation of the proposed methodologies, shedding light on their specific characteristics and effectiveness in distributed scenarios. The findings of this research are significant, as the proposed approach not only achieves outstanding recognition performance on one of the broadest publicly available datasets but also surpasses all baseline models by a considerable margin. One of the key strengths of this solution is its capability to operate efficiently at the local level without necessitating prior knowledge about the data distribution or specific characteristics across nodes. This adaptability makes the approach highly applicable and versatile, addressing a crucial gap in FL research. Consequently, Study 5 represents a major leap forward in FL, showcasing the potential of integrating diverse learning techniques to optimize performance in scenarios characterized by limited and scattered data.

8.1.1 Challenges and Solutions in Representation Learning from limited data

The problem of learning visual models from very few available training data and scattered distribution poses a significant challenge in the field of ML. The majority of DL algorithms in general require a substantial amount of data to achieve the desired performance. However, when it comes to real-world applications, the available data sources are typically limited and often fragmented, making it challenging to apply traditional learning techniques. Many research groups have dedicated resources to find reliable solutions [159], but despite the abundance of published work, the problem of learning meaningful representation from sparse data still poses significant difficulties.

To achieve truly robust recognition performance in various visual analysis tasks where limited data are involved, it is essential to address several key issues. These include ensuring data quality and accurate annotation, mitigating the effects of overfitting and promoting generalization, handling imbalanced class distributions, *etc.* [160]. Initially, the research efforts focused on combinations of data pre-processing techniques, domain adaptation approaches and regularization methods [161, 162, 163, 164, 165, 166, 167, 168]. While these strategies were effective to some extent, recent advances in representation learning techniques [136, 169, 170] have significantly boosted the field, literally transforming the way visual analysis is approached. These techniques enable the extraction of meaningful and discriminative representations from limited data, enhancing the model's generalization power, especially on unseen samples. As a result, the extracted representation can better capture the inherent patterns within the data, enabling models to leverage limited data more effectively. This, in turn, leads to improved performance and robustness in achieving the desired task.

While there has been extensive research on applying representation learning to centralized data settings, the exploration of its application to federated datasets with incomplete annotations and scarce data samples is still relatively limited, due to data privacy concerns, data and system heterogeneity constraints, and limited annotations bottlenecks. Research groups have devoted resources to approaches specifically designed for federated settings, leveraging the power of representation learning while respecting the decentralized nature of data. These approaches often involve a range of learning techniques, such as self-supervised, semi-supervised, unsupervised learning, and transfer learning [171, 172, 173, 174, 175]. In the context of FL, the principal impediment resides in the sensitivity of data, rendering it non-transferable from local users to the central server. Moreover, an associated issue pertains to the quantity of annotated data. The central server may house semi or fully-annotated data, or data that is an amalgamation of different datasets.

In traditional FL a critical limitation faced by local nodes pertains to their constrained resources, both in terms of data processing and storage capabilities. As a result of these constraints, the deployment of both self-supervised and semi-supervised learning methodologies at the local level becomes infeasible. This challenge accentuates the pressing need for an innovative approach within FL. Such an approach should have the flexibility to alternate between and synergistically combine self-supervised and supervised learning techniques, specifically tailoring them to the distinct dynamics of both local and global FL rounds. The evolution and integration of these methods can potentially optimize the FL process, harnessing the strengths of both self-supervision and traditional supervision within the federated context.

Through the application of self-supervised learning methodologies in our approach, it becomes plausible to surmount the challenges posed by non-annotated data. In the proposed specific methodology, we harnessed Contrastive Learning (CLR) [136] with the objective of discerning the optimal representations by drawing contrasts between each instance of the dataset and the remainder thereof. This training modality does not necessitate labels for the data and yields exceptional outcomes, particularly given that the data requisitioned for training do not demand any annotation effort whatsoever.

In a paradigm involving fully-supervised training of the central server, it is incumbent that the data are fully-annotated with a high degree of label quality. However, when employing self-supervised training, there is no requirement for fully annotated data within the central server, hence enabling us to capitalize on all available data situated within the central server to construct a highly potent feature extractor that learns data representations independent of the need for classes.

Contrastingly, local clients acquire representation predicated on the specific annotated dataset that is accessible to them; hence each client learns the optimal representation dedicated to the specific dataset. By possessing a locally fine-tuned model for each distinct local dataset, which when amalgamated on the central server and combined with the powerful feature extractor located on the centralized server, the updated model will embody knowledge from each client and from the expansive dataset in the central server.

The main contributions of this work are summarized as follows:

(a) **A novel self-supervised learning approach that empowers the central server to exploit every piece of data within its possession**, by leveraging CLR techniques to explore meaningful representation and extract informative features at a global level even in scenarios where labeled data are limited.

(b) **A hybrid FL scheme that seamlessly blends self-supervised and supervised techniques**, adapting them to the unique dynamics of local and global learning rounds within the federated context.

(c) **Validation of the proposed approach by conducting extensive comparisons with a fully-supervised learning process within the same FL scheme**, thereby demonstrating the efficacy of our combination of contrastive and supervised learning on two standard benchmark databases. Extensive experimentation and comparative evaluation highlight the advantages of the proposed schemes under various federated scenarios.

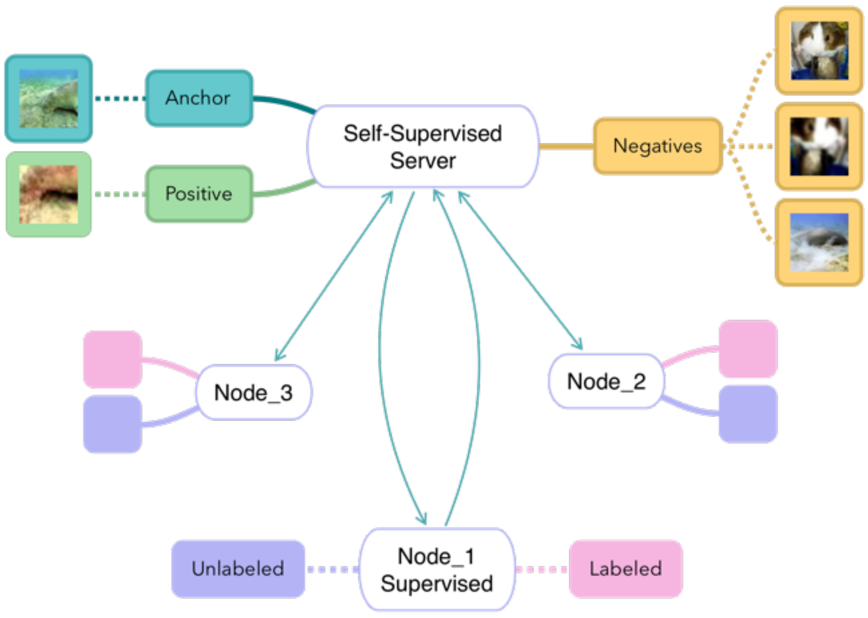


Figure 8.1: The proposed FL scheme utilizes a centralized server to create and share the global model, while several local nodes participate asynchronously in the training process.

8.2 Related Work

It is undeniable that there are many annotated datasets available nowadays. However, it is clear that we cannot constantly access full databases relevant to every imaginable work. The use of supervised learning approaches, which primarily rely on the availability of properly annotated datasets, is severely constrained by such obstacles. Self-supervised learning techniques, in contrast, have the advantage of not requiring annotations, demonstrating their applicability to a variety of issues. These approaches have shown tremendous promise when applied to semi-annotated or non-annotated data, providing outcomes that are, in fact, notable. They achieve the latter by creating a self-supervised job for learning.

Auto-encoders [176] is one such classic example. When compared to the lack of annotated data, they prove to be especially useful. In order to decode the latent representation and return to the original input, autoencoders transpose the input into a latent space representation first. Replicating the input as accurately as possible is the main goal. Despite being an unsupervised scheme, Generative Adversarial Networks (GANs) [177] can function in a self-supervised manner. By learning to perform tasks such as predicting the rotation of an image [178] or filling in missing parts of an image [179], the model discovers rich feature representations of the data without requiring explicit labels, which can then be used for solving demanding computer vision tasks under limited data settings.

Recently, similarity learning techniques have attained particular focus. More specifically, Bootstrap Your Own Latent (BYOL) [180] is a self-supervised learning algorithm that leverages two neural networks, namely a target network and an online network. Instead of relying solely on negative samples for learning, BYOL trains the online network to align its predictions with the output of the target network, which is a slow-moving average of the online network, thereby learning representations from different augmentations of the same image.

8.2.1 Contrastive Learning

CLR serves as a potent instrument within the realm of self-supervised learning [181, 182, 183]. Its fundamental aim revolves around crafting representations that induce a tendency for similar instances to congregate in the embedding space, while simultaneously propelling dissimilar instances to be dispersed at a greater distance. To fulfil this aspiration, every instance undergoes transformation, yielding a variant distinct from the original yet preserving the underlying principle intact. This is facilitated through the deployment of augmentations that possess the ability to modify the appearance of the instance without distorting its foundational principle [184].

This theory paves the way for a self-supervised learning task, wherein the network endeavors to diminish the distance between the original and the augmented instances in the latent space, while concurrently increasing the space between all the other instances within the dataset. Within this postulate, the network strives to comprehend the representation of the instances in a manner that enhances their separability in the embedding space, thereby simplifying the process of discerning distinct instances.

8.2.2 FL approaches

FL is an innovative ML approach that leverages decentralized data and computational resources to deliver more tailored and flexible applications while upholding the privacy of users and organizations. FL has demonstrated exceptional results in numerous visual analysis tasks, such as image classification, object detection and action recognition [185, 186, 187], indicating its robustness and effectiveness in these areas.

The research on FL has focused on increasing communication efficiency and accelerating model updates. McMahan *et al.*'s [155] pioneering work introduced the concept of averaging local stochastic gradient descent updates to increase the calculated quantity of each client between communication rounds. To address low device participation, non-independent and identically distributed (Non-IID) local data, other studies, employ online knowledge distillation approaches, also called codistillation, for communication-efficient FL. Unlike transferring model updates, codistillation focuses on transmitting the local model prediction on a public dataset that is accessible to multiple clients. This method proves beneficial in reducing communication costs, particularly when the size of the local model exceeds the size of the public data[158, 157, 188]. In a recent study [135], researchers introduced a novel approach, named MOON, where they propose a local model constrastive loss comparing representations of global and local models from successive FL rounds. This technique aims to improve the training of individual parties by conducting CLR at the model-level, specifically in the feature representation space, pushing the current local representation closer to the global representation and further away from the previous local one. Similarly, authors in [71] proposed a distillation-based regularization method, named FedAlign, that promotes local learning generality while maintaining excellent resource efficiency.

In an attempt to leverage the unlabeled data available across multiple nodes, while utilizing the limited labeled samples, several recent studies explored novel algorithms, ranging from self-training, and co-training to knowledge distillation. Authors in [189] proposed a novel semi-supervised method, termed FedMatch, which learns inter-client consistency between nodes, and decomposes model parameters to reduce interference between both supervised and unsupervised tasks. Zhang *et al*. [171] proposed an unsupervised representation learning algorithm, called FedCA, where each client performs unsupervised learning on its local data, leveraging techniques such as CLR to capture meaningful patterns and representations. The learned representations are then aggregated and refined at a central server, resulting in a powerful and comprehensive representation model that encapsulates the knowledge from all distributed sources. In a similar approach, Han *et al*. [175] introduced FedX, an unsupervised FL framework that learns unbiased representation from decentralized and heterogeneous local data, by employing a two-sided knowledge distillation with CLR as a core component. Its adaptable architecture can be used as an add-on module for existing unsupervised algorithms in federated settings. Moreover, two federated self-supervised learning frameworks for images with limited labels were proposed in [172], based on federated CLR with feature sharing (FedCLF). In contrast to previous approaches that assume labeled data are available at the client-side, Long *et al*. [173] introduced FedCon, a novel framework designed to address scenarios where local client data is unlabeled and only the server has access to labeled data. FedCon tackles this challenge by employing a contrastive network architecture, which consists of two subnetworks, enabling effective handling of the unlabeled data on the client-side. Recently, Khowaja *et al*. [174] proposed the SelfFed framework that operates in two distinct phases. The initial phase involves self-supervised pre-training, where a decentralized approach is employed to train a Swin Transformer-based encoder. In the subsequent phase, referred to as fine-tuning, the framework incorporates a contrastive network and introduces a novel aggregation strategy. This phase aims to refine the pre-trained encoder using limited labeled data specific to the target task.

While these approaches have made significant contributions to FL with limited labeled data, they have certain limitations. One limitation is the reliance on labeled data on either the client-side or the server-side. Another limitation is the lack of exploration of meaningful representations and informative features at a global level. In contrast, the proposed method introduces a novel self-supervised learning approach that enables the central server to harness the entirety of available data within its possession. By utilizing CLR techniques, this study explores strong representations and extracts informative features at a global level, even in scenarios where labeled data are limited or unavailable. This allows for more effective utilization of data and enhances the performance and generalization capabilities of the final model.

8.3 Methodology

Problem statement: In a federated system, data are inherently localized to individual clients and the dissemination of this information to other clients is strictly prohibited. On the other hand, the central server, effectively serving as a simulated environment for the local clients, has the ability to leverage a substantial volume of data for training purposes, thereby accommodating a broad array of data variations. The goal is to train each local model separately to its own subset of data, while the centralized server is trained in a self-supervised manner on the vast database, with high diversity, regardless of whether the data are annotated or not, whereas the complete system of the local clients maintains uniformity in the overall accuracy ascertained in the common test set.

In the methodology that we advocate, a hybrid learning scheme involving self-supervised and supervised training strategies is employed. Owing to the substantial magnitude of images contained within the Tiny-ImageNet[190] dataset, it serves as an ideal candidate for the training of the central server through the implementation of unsupervised CLR. On the other hand, for the cultivation of individual local clients, the CIFAR-100[123] dataset is deployed, partitioned in both a balanced and unbalanced manner to facilitate experiments that cater to both IID and Non-IID conditions. CIFAR-100 and Tiny-ImageNet are used since they are contextually similar as shown in figure 8.2.

8.3.1 FedLID Local Supervision

The local clients are trained in a supervised fashion utilizing the CIFAR-100 dataset. A distinct segment of the dataset is allocated for each client that remains consistent throughout all federated rounds, on which they receive training. The training portion of the dataset is utilized for the purpose of training, while the validation subset serves to authenticate each of the clients' models. Notably, the validation set is communal, hence accessible to all clients. For the purpose of training the clients' networks, a classification head was appended that maps the feature vectors to the total number of classes present in the dataset, amounting to 100. Cross Entropy was deployed as the loss function for the training process, the formula is displayed below:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{C} y_{ij}\log p_{ij} \tag{8.1}$$

where N is the total number of images, C the total number of classes, y_{ij} the label of the i, j image and p_{ij} the probability of the class. Adam[191] algorithm was chosen as the optimization strategy.

8.3.2 FedLID Global Supervision

In the federated architecture, the primary server undergoes training via a contrastive approach on the dataset $X_m = (x_m^i, y_m^i), i = 1, ...N$, wherein the labels are disregarded. Each image from the said dataset serves as the anchor image x_m^i. To generate a slightly variant, positive image denoted x_p^i, image augmentations are utilized. Both of these images are subsequently passed through the network function $\Phi(\cdot)$, and are then projected into the latent space for the implementation of CLR as delineated in[136].

The embedding dimensions of the feature map encompass 126 channels. For the purpose of projection, the final fully connected layer of the model is omitted, and a novel projection head is affixed that maps the 512-dimensional latent space to a 126-dimensional space employing a Rectified Linear Unit (ReLU)[192] activation function. The feature maps of both the anchor image $\Phi_m^i = z_i$ and the positive image $\Phi_p^i = z_j$ are harnessed for the calculation of the contrastive loss. The formula for the loss is indicated below:

$$\mathcal{L}_{i,j} = \log\frac{\exp\left(sim(z_i, z_j)/\mathcal{T}\right)}{\sum_{k=1,[k\neq i]}^{2N}\exp\left(sim(z_i, z_k/\mathcal{T}\right)} \tag{8.2}$$

where $\mathcal{T}$ is the temperature that is a scaling factor used to control the concentration of the output distribution, affecting the hardness of the positives in the CLR framework. Through the implementation of contrastive loss, the network strives to learn representations that induce proximity between the anchor image and the positive image in the embedding space, whilst ensuring a separation between the anchor image and all other images within the dataset. The representations learned in this manner would have the capacity to segment the latent space in accordance with the context, without the true comprehension of the class.

8.3.3 Federated aggregation

Optimization strategies and in particular aggregation algorithms play an important role in FL as they are responsible for combining the knowledge from all devices/nodes while respecting data's privacy. Although adapting FL to fully-supervised federated schemes seems to be straightforward, shifting to more complex schemes that involve further self-supervision steps, as the ones mentioned earlier, can prove to be more challenging than anticipated. Prior to the application of the proposed self-supervision step, federated optimization is realized for the locally fully-supervised trained models (Figure 8.1). In particular, the broadest aggregation mechanism, namely FedAvg, has been followed, examined in detail and adapted to the specific scenarios. Firstly, the server distributes the initial version of the model to each node for training on local data. This first version can either be pre-trained on a predefined dataset (*i.e.* ImageNet1k[193]) or be initialized randomly. In each round, the algorithm performs a set of local model updates (*i.e.* cross-entropy loss) on a subset of nodes, followed by a server-side aggregation task, trying to minimize the following objective function, which is actually the sum of the weighted average of the clients' local errors, where F_k is the local objective function for the kth device and p_k specifies the relative impact of each device:

$$\min_{w} = \sum_{k=1}^{N} p_k F_k(w) \tag{8.3}$$

In the final step, the updated global model is forwarded to the local nodes for another round of training. The process is continuing until the global aggregated model is fully trained and achieves the desired performance.

8.3.4 FedLID Algorithm

In this section, the complete algorithm of the described procedure is presented. The entire process is divided into two main algorithms, the first one describes the required steps for training and updating the network in the main server, while the second one defines the action for training and updating the network in each local client.

Algorithm 17 ClientUpdate Function

```
1: ClientUpdate($D_l$, $\theta$):                          ▷ Run on specific client
2: for each local epoch $i$ from 1 to $E$ do
3:     for each batch in $D_l$ do
4:         $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l, \text{labels})$     ▷ Update the client model with Eq. 6.1
5:     end for
6: end for
7: return $\theta_l$
```

8.4 Experiments

8.4.1 Data settings

Within this segment, experimental results from the application of the proposed Federated Self-Supervised learning strategies are presented. The evaluation process utilized the CIFAR-100 image classification dataset as a benchmark. This dataset was adapted into both IID and non-IID variations to reflect different FL scenarios. One is replicating the balanced data, *i.e.* IID dataset, while the other is an extreme example of

Algorithm 16 FedLID Algorithm

H

```
Require: $\mathcal{C}$ is the total number of clients, $\theta_g$ represents the parameters of the global model, $\theta_l^i$ represents the parameters of the local models, $\mathcal{D}_l^i$ are the separate datasets for the local clients, $\mathcal{D}_g$ is the dataset of the central server, $\mathcal{D}_val$ is the common validation dataset, and $\eta$ is the learning rate.
 1: Initialize global model $\theta_g$
 2: for each $i$ from 1 to $C$ do
 3:     Initialize local models $\theta_l^i$
 4: end for
 5: Prepare global dataset $D_g$
 6: for each $i$ from 1 to $C$ do
 7:     Prepare local datasets $D_l^i$
 8: end for
 9: Prepare common local dataset for validation $D_{val}$
10: Server executes:
11: for each Federated round $t = 0, 1, 2, \ldots$ do
12:     for each client in parallel do
13:         $\theta_{l_{t+1}} \leftarrow \text{ClientUpdate}(D_l, \theta_{l_t})$
14:     end for
15:     $\theta_g \leftarrow \text{FedAvg}(\theta_l) Eq.8.3$
16:     for each global epoch $i$ from 1 to $G$ do
17:         for each batch in $D_g$ do
18:             $\theta_g \leftarrow \theta_g - \eta \nabla L(\theta_g)$    ▷ Update the global model with Eq. 7.5 (Contrastive Loss)
19:         end for
20:     end for
21:     Distribute the updated global model to the clients
22:     Validate the updated models on $D_val$
23: end for
```

a Non-IID dataset. From both datasets, a set of 100 categories was used with the total number of participating nodes being 10, among which the images are to be distributed.

IID and Non-IID Settings

Like previous studies [51], the parameter $\alpha > 0$ controls the identicalness among participants. Different α values were tested, where with $\alpha \to \infty$, all participants have identical distributions and $\alpha \to 0$, each participant has examples from only one class. To support the IID setting, the dataset was divided with medium heterogeneity by setting $\alpha = 1$. Therefore, a node can have images from any number of classes. In contrast, for the case of the Non-IID set, the original CIFAR-100 dataset was divided, with a higher level of heterogeneity by setting $\alpha = 0.5$. Here, nodes tend to have a significant number of samples from some classes and few or no samples for the other classes. Each node randomly sampled $\frac{1}{10}$ of training and validation data respectively, while the test set data were left out for the final system evaluation, both at the global and local levels.

Image augmentations

For the CLR scheme, a data augmentation strategy as the one in SimCLR[136] is adopted. Initially, a stochastic crop of the image is procured, which is subsequently subjected to a random horizontal flip. This is then followed by arbitrary distortion of brightness, contrast, hue, and saturation parameters, complemented by an optional grayscale transformation. Subsequently, a Gaussian blur filter is administered as the terminal step of the augmentation process. The image is ultimately resized to dimensions of 224×224 and undergoes normalization.

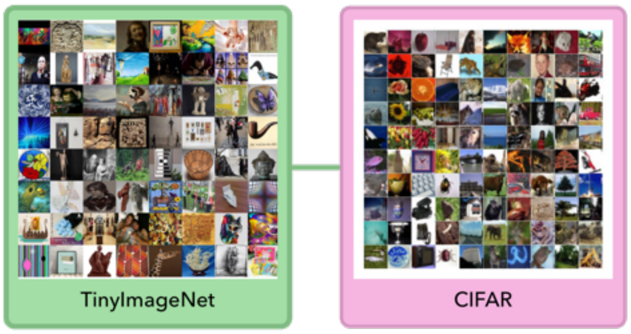


Figure 8.2: Images from Tiny-ImageNet (left) and the CIFAR-100 (right), showcasing the contextual similarities of the two datasets.

8.4.2 Implementation details

Architecture and Parameters

The architecture of the proposed method employs a distributed framework, composed of a Convolutional Neural Network (CNN)[194], specifically ResNet18[124] is used for experiments because it is shallow, fast in training and has satisfactory results, that is deployed both at the centralized server and the local clients, albeit with unique modifications for each entity. The step-by-step process of the training of the federated scheme is illustrated in Algorithm 16. For the central server, the final fully connected layer of the network is replaced with a projection head, enabling the transition of the ResNet feature map to a new embedding space, thereby facilitating the CLR scheme. Conversely, the local clients are trained via supervision, resulting in the substitution of the final fully connected layer with a classification head that maps the feature vectors to the corresponding classes, the full algorithm of the local supervision is depicted in 17. Regarding the optimizer, Adam is employed with a learning rate of 0.001 and the batch size of the system is 256. It is crucial to note that the ResNet model retains knowledge acquired from the ImageNet1k[193], as it operates on the basis of pre-trained weights. To facilitate an unbiased comparison between the proposed approach and leading-edge federated systems, additional experiments are conducted employing the ResNet50 architecture.

Federated setting

In the context of the training paradigm for the suggested FL system – a system comprising both a centralized server and 10 local clients – a comprehensive training period of 10 federated rounds is undertaken. Within each of these rounds, the local clients individually embark on a training process spanning 10 epochs, operating in isolation without any inter-client communication. Upon completion of their respective training, the local clients communicate with the central server, transferring their uniquely derived weights. This information is assimilated by the central server which proceeds to aggregate the weights and undergoes a training cycle for 60 epochs prior to disseminating the updated model to the local clients. Post distribution, the refreshed model undergoes an evaluation process leveraging the common test set accessible to all clients.

Baselines

To affirm the validity of FedLID as unequivocally as possible, a comparative performance analysis across two distinct configurations was executed. Initially, benchmark experiments were undertaken within a fully-supervised federated framework, in both IID and Non-IID scenarios, wherein the volume of data accessible on the central server progressively reduced in increments of 20%, ranging from 100% to 20%. Subsequently, a comparative evaluation was conducted against the following cutting-edge benchmarks: FedCA and FedSimCLR [171].

Training Setup

The proposed federated system's training was executed on a single computer, outfitted with a GeForce RTX 3090 (VRAM 24 Gbs) and furnished with 32 Gbs of RAM.

8.4.3 Performance Evaluation

IID Setting

The presentation of results, under circumstances of equitably distributed data amongst the local clients, is delineated below. As discernible from Table 8.1, when juxtaposed with the fully-supervised framework, FedLID provides superior results (47.52%) with relative improvement over the baseline of 8.05%, even when pitted against the optimum case scenario of supervision (43.98%), which implies the utilization of 100% of the data situated within the central server. As anticipated, the overall accuracy experiences a decline with a diminishing quantity of training data in the server. Consequently, in situations where the central server houses extremely limited annotated data, the proposed method surpasses the conventional supervised FL. Figure 8.3 shows the impact that the training on the server has on the overall accuracy of a client. The underlying cause for the observed outcome is that each local client independently assimilates the representation of its specific training subset. However, when these learned representations converge on the central server, this disparate information is consolidated. Coupled with the CLR setting deployed on the server, these combined representations converge closer to the authentic data distribution of the test set.

Percentage of Data in the Central Server	Average Accuracy over the Local Clients
20%	42.15%
40%	41.79%
60%	42.74%
80%	43.81%
100%	43.98%
FedLID(Ours)	**47.52%**

Table 8.1: Comparison of our method with different percentages of available data in the central server. The data in the local clients are divided equally.

It is subsequently discerned that FedLID surpasses the performance metrics of contemporary state-of-the-art algorithms as shown in Table 8.2. This superior performance of 47.52% is achieved even when utilizing shallower network architectures and a larger number of clients, both factors contributing to reduced training data for each individual local client.

Method	Architectute	Clients	CIFAR100
FedSimCLR	ResNet50	5	34.18%
FedCA	ResNet50	5	39.47%
FedLID(Ours)	ResNet18	10	**47.52%**

Table 8.2: Average Accuracy over the local clients on CIFAR-100 udner the IID setting with $\alpha = 1$.

Non-IID Setting

Non-IID data typically contain diverse patterns and variations across different nodes, which inherently lead to challenges in learning a generalisable model. In that context, FedLID achieves superior results (48.88%), as depicted in Table 8.3, by leveraging the inherent structure and relationships within the global data, with a relative improvement of 39.02% over the baseline system. Figure 8.4 demonstrates the effect that the main server has on the performance of the client. By pretraining the global aggregated model on a self-supervised task and subsequently fine-tuning it on the target local supervised task, the model can effectively exploit the knowledge gained at a global level while adapting to the specific characteristics of the Non-IID dataset, resulting in improved accuracy and generalization at the local level. It is entirely plausible in practical federated systems that the data distributed among local clients may not be evenly distributed, and the server data could be semi-annotated or non-annotated. Under such realistic conditions, the proposed methodology outperforms all baseline cases.

Percentage of Data in the Central Server	Average Accuracy over the Local Clients
20%	32.83%
40%	33.08%
60%	32.84%
80%	34.32%
100%	35.16%
FedLID(Ours)	**48.88%**

Table 8.3: Comparison of our method with different percentages of available data in the central server. The data in the local clients are divided in an imbalanced way.

In relation to a comparison with state-of-the-art methods, FedLID demonstrates exceptional performance, outpacing the next closest approach by a 10% margin, as portrayed in Table 8.4. This impressive outcome is achieved despite the fact that each client has less data, and the network used is smaller in terms of parameters.

It is evident from the results that the proposed method exhibits improved accuracy on Non-IID data compared to IID data distributions (48.88% and 47.52% respectively). This

Method	Architectute	Clients	CIFAR100
FedSimCLR	ResNet50	5	33.63%
FedCA	ResNet50	5	38.94%
FedLID (Ours)	ResNet18	10	**48.88%**

Table 8.4: Average Accuracy over the local clients on CIFAR-100 udner the Non-IID setting with $\alpha = 0.5$.

is due to the fact that in the IID setting, the training data across nodes is representative of the overall population, and the local model can learn common patterns more effectively, leading to fast convergence. In addition, the pretext task used for self-supervision is almost aligned with the target supervised task, thus the fine-tuning process on the balanced (*i.e.* uniformly distributed across nodes) dataset does not struggle to align the learned representation with the task-specific requirements; hence leading to less benefit from self-supervision step. On the contrary, self-supervised methods often excel in scenarios where there are variations, complex dependencies, or imbalances in the data distribution, as they can capture the underlying structure and extract meaningful representations. Consequently, in homogeneous scenarios, the impact of self-supervision may be relatively diminished, thus largely explaining the difference in performance.

Weakly annotated setting

In an effort to investigate the impact on performance, we intentionally reduced the available data to 10%, aiming to create conditions of partial annotation in the local nodes. This approach allows us to study the consequences both prior to and following the application of the proposed self-supervised method at the global level. The results demonstrate that despite the significant reduction in local data and a limited number of training iterations (10 local epochs and 10 federated rounds), the performance slightly increases compared to the baseline approach (*i.e.* FedCA) in the IID setting while the comparison is favourable in the context of Non-IID, as depicted in Table 8.5. It is evident that the limited number of local epochs hampers performance improvement, and the local model has the potential for further convergence, as illustrated by the accompanying diagrams. Furthermore, it can be observed that increasing the number of communication rounds consistently enhances performance, showcasing the cumulative strength of the self-supervised model in the federated aggregation process.

8.5 Insights and Contributions

The fifth study, delving into the field of representation learning with limited data in FL environments, marks a significant leap forward in addressing some of the most pressing challenges in FL. The study introduces a novel approach that innovatively combines self-supervised and supervised learning techniques within a federated framework,

Label Fraction	Setting	Method	CIFAR100
10%	IID	FedCA	32.09%
		FedLID(Ours)	**33.51%**
	Non-IID	FedCA	22.46%
		FedLID(Ours)	**24.12%**

Table 8.5: Accuracy of the state-of-the-art in weakly annotated scheme with different label ratios. FedCA's total number of local clients is *5*, while FedLID is utilizing a federated system with *10* clients in total.

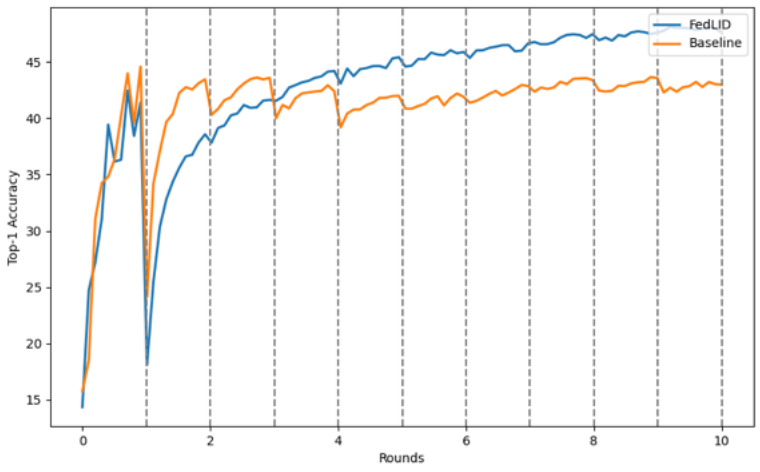


Figure 8.3: Comparison of the top-1 accuracy on CIFAR-100 in an IID setting, over the federated rounds of FedLID and the baseline. The dotted line is the aggregation of the local models and the training of the centralized server.

a methodology tailored to navigate the complexities associated with sparse and imbalanced data distributions across nodes. The heart of this study lies in its custom self-supervised learning strategy, deployed at the global level, which effectively capitalizes on the untapped potential of unlabeled data. This strategy is creatively complemented by supervised learning techniques at the local level, ensuring the optimal utilization of available labeled data. The synergy of these methodologies results in a robust and versatile learning mechanism, effective in adapting to various federated settings, especially those characterized by limited and uneven data distribution. The findings are significant: the study's approach not only demonstrates remarkable recognition performance on a broad publicly available dataset but also outstrips all baseline models by a considerable margin. One of the standout features of this solution is its ability to function efficiently at the local level without requiring prior knowledge of data distribution or specific characteristics across nodes, a capability that highlights its adaptability. The successful integration of

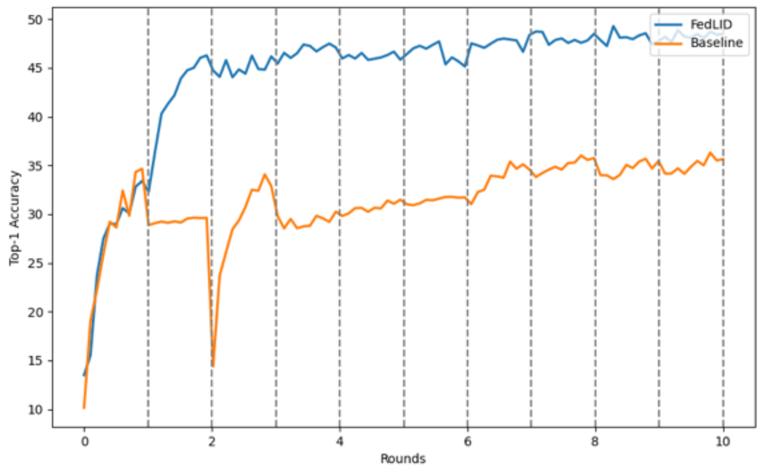


Figure 8.4: Comparison of the top-1 accuracy on CIFAR-100 in a Non-IID setting, over the federated rounds of FedLID and the baseline. The dotted line is the aggregation of the local models and the training of the centralized server.

self-supervised and supervised methods in this study provides a compelling direction for future research. It lays the groundwork for exploring the variability of model architectures across nodes in subsequent studies, potentially leading to even more adaptable and efficient FL systems.

Chapter 9

Study 6: Tackling Model Architecture Variability in FL

9.1 Overview of the Study - Rationale Behind Addressing Model Architecture Variability

In the realm of FL, addressing model architecture variability is not just a technical challenge but a pivotal aspect of the field's evolution, especially considering the ever-increasing number of model architectures emerging in the literature. This focus on architecture variability emerges from the unique nature of FL, where diverse devices or participants, each with their own data and computational constraints, collaboratively train a shared model. Modern ML has witnessed an expansion of model architectures, each tailored to specific types of data and computational tasks. For instance, ResNets (Residual Networks) [124] have gained popularity for their ability to enable the training of extremely deep networks by using skip connections. On the other hand, EfficientNets [195] offer a balanced approach, scaling different dimensions of the network in a compound manner to achieve remarkable efficiency and accuracy. Similarly, MobileNets [196] are designed to provide lightweight, yet effective, neural networks for mobile and edge devices, focusing on optimized performance in resource-constrained environments. Such architectures have exceled in a wide spectrum of computer vision tasks including classification[197, 198, 199], detection[200, 201], retrieval[202, 203].

In the context of FL, this diversity in model architectures presents both challenges and opportunities, which are elaborated in Chapter 2. Each architecture has unique characteristics in terms of how it processes data, its computational complexity, and its suitability for different types of tasks. For instance, while ResNets might be ideal for scenarios where high accuracy is paramount, MobileNets could be more appropriate for FL scenarios involving mobile devices due to their efficiency and reduced computational requirements. As detailed in previous studies, in FL, participants ranging from powerful servers to resource-constrained edge devices collaborate in training a shared model, while each retaining their unique datasets. This setup presents a complex picture of heterogeneity, not only in terms of computational capabilities but also in data characteristics. The data across these diverse devices are often non-IID, presenting significant challenges in model training and performance. Hence, the choice of model architecture in such a scenario becomes critical. A one-size-fits-all approach is impractical; instead, a more tailored strategy is required, where the model architecture aligns with the computational capabilities and data characteristics of each node.

One of the core challenges identified in previous studies is communication efficiency in FL [45] since the decentralized nature of FL adds another layer of complexity. The architecture of models directly influences the data that needs to be communicated between the central server and distributed nodes. Optimizing these model architectures for diverse environments can substantially reduce communication overhead. This requires innovative approaches in model aggregation and updating to accommodate different architectures while maintaining the overall efficiency and effectiveness of the learning process.

Moreover, the foundational promise of FL is its ability to preserve privacy while training models effectively without the need to centralize data [46]. This entails a distinct approach where models are tailored to specific data types and privacy requirements of different nodes. Variable model architectures allow for this flexibility, addressing the diverse privacy and security concerns across nodes. Scalability and robustness are other critical aspects that demand a focus on model architecture variability. As FL systems expand to include more nodes and diverse applications, robust models capable of adapting to varying architectures become increasingly important. This scalability encompasses not just the number of nodes but also the diversity in types of tasks and data modalities. Furthermore, the real-world applicability of FL spans a broad range of domains, from healthcare to smart cities. Each application domain might require different model architectures due to the nature of the data and specific task requirements. Therefore, developing strategies to manage architecture variability is crucial for the practical deployment and success of FL in these diverse fields.

As indicated in prior studies, FL models need to continuously learn and adapt to new data or tasks, a concept known as incremental learning. This requirement for models to evolve and incorporate new information without forgetting previous knowledge underscores the need for flexible architectures that can accommodate such dynamic learning processes. In addition, the increasing number of model architectures also opens up new directions for research in FL. It provides an opportunity to explore hybrid models or meta-learning approaches that can dynamically adapt to the most suitable architecture based on the task at hand and the nature of the data. Such exploration could lead to more versatile and powerful FL systems capable of handling a wide range of tasks and data types more efficiently.

Considering all the aforementioned aspects, the exploration of model architecture variability in FL stands as a strategic response to the multitude of challenges and opportunities inherent in this innovative learning paradigm. It embodies a dedicated effort to enhance the efficiency, scalability, privacy, and real-world applicability of FL. This attempt builds upon the foundational insights extracted from preceding studies, pushing the boundaries of what is achievable in collaborative, decentralized learning environments. Notably, this study is the first, to the authors' knowledge, to step on this path. It pioneers in addressing the complexity and diversity of model architectures within the FL framework, making it a groundbreaker in the field and setting a point of reference for future research in this area. The proposed FL system is illustrated in Figure 9.1

The main contributions in addressing model architecture variability in FL can be summarized as follows:

a) **Model-Agnostic FL Framework:** This study introduces a novel FL approach that is agnostic to the specific model architectures used across nodes. It successfully integrates diverse architectures such as ResNet, EfficientNet, and MobileNetV3 in FL settings, demonstrating the framework's adaptability to various computational capabilities and data requirements at each node.

b) **Hybrid Learning Approach with Focus on Representation:** A key innovation of this study is the blend of supervised and self-supervised learning paradigms, focusing on representation learning. This approach optimizes local models based on their specific data and then aligns them with global data patterns through self-supervised learning. It signifies a shift away from traditional federated averaging, emphasizing a representation-centric aggregation method.

c) **Efficient Handling of Model Architecture Variability:** The study addresses the challenge of combining outputs from various model architectures during the FL process. By focusing on the outputs from specific blocks or layers and adopting a feature excitation method, the study effectively manages the aggregation of learned representations from different architectures, ensuring coherence in the federated system.

9.1.1 Advancements and Challenges in Federated Representation Learning

The landscape of FL has rapidly evolved, as evidenced by the plethora of research[204, 205, 206, 207, 208, 209, 210] that have tackled its fundamental issues and put forth inventive ways to improve its effectiveness, confidentiality, and expandability. The promise of this emerging sector to facilitate collaborative learning without sacrificing data privacy has garnered a great deal of attention.

Representation learning serves as a cornerstone in the efficacy of FL, underpinning the ability to distill and generalize knowledge from distributed data. This foundational technique facilitates the extraction of informative features, enhancing the collaborative intelligence of FL models, and enabling them to perform robustly across all client datasets. The authors of [211] introduced an FL framework that aims to cultivate a common data representation among clients while maintaining distinct local heads for each. This approach capitalizes on the distributed computing resources and performs frequent local updates targeting low-dimensional parameters to achieve linear convergence and produce accurate representations. FedX[175] learns neutral representations from diverse local data by utilizing contrastive learning[212] to enable a bilateral knowledge distillation, allowing the system to function without the need for shared data features. Due to privacy constraints, FL schemes are required to excel on limited data on a per-client basis. Concurrently, as knowledge is centralized through model updates, it is essential for the aggregated model to assimilate and generalize information across the dataset from all clients while preventing the loss of client-specific characteristics. The work of Psaltis *et al*. [213] demonstrates that contrastive learning can significantly increase the performance of the local clients even when the distribution is imbalanced and scattered across them. Li *et al*. [214] employ contrastive learning at the model level, leveraging the congruence among model representations to refine the local training processes of individual clients. FedCA [215] used a shared split of the dataset to align the representations of the images. The framework is distinguished by its integration of centralized sample representations from each client to ensure a uniform representation space accessible by all, and a module that synchronizes each client's representation with that of a foundational model trained on publicly available data.

Addressing the aggregation of learned representations on a global scale presents significant challenges. Despite numerous attempts to synchronize heterogeneous models across diverse clients, current approaches struggle to achieve seamless architecture alignment without modifying model structures or directly aggregating model weights. This limitation points to the untapped potential for innovative methods capable of integrating diverse architectures without the need for altering client models. Although existing research has explored methods like distillation techniques or various ensemble strategies [122, 216, 217, 218, 219], these approaches often necessitate mapping features into a common latent space to a shared latent space for knowledge integration and weight aggregation[220, 221, 222, 223, 224]. Such processes typically involve additional training of either entire networks or specific network components (*e.g.*, network heads), leading to increased training durations and reduced efficiency and scalability, thereby diminishing the practicality of these solutions.

9.2 Strategies for Managing Diverse Model Architectures

Problem statement: The effective integration and management of a wide array of diverse model architectures across various nodes in a federated network. This problem arises due to the heterogeneous nature of FL environments, where nodes, ranging from high-powered servers to resource-constrained edge devices, each come with their own unique data characteristics and computational capabilities. The proposed methodology for managing diverse model architectures in an FL system is designed to be model-agnostic,

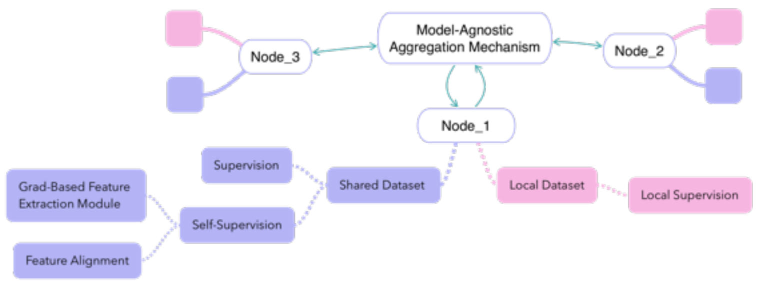


Figure 9.1: Schematic of the proposed FL System Architecture illustrating the feature extraction and alignment modules on the local dataset of each client that harmonizes the heterogenous architectures and the fully-supervised training on the local dataset. The Model-Agnostic Aggregation Mechanism is the process performed on the main server to create an enriched representation of the local dataset, derived from all the clients

leveraging the strengths of representation learning. This strategy is distinct from conventional federated averaging, taking advantage of the potency of both supervised and self-supervised learning paradigms, but lacking the advantage of aggregating the weights of the models due to the heterogeneity of the architectures. Traditional approaches, including FedAvg[22], are insufficient in confronting the challenges posed by the intrinsic diversity of ML models deployed throughout the network. Moreover, strategies such as model distillation and the integration of feature fusion at the upper layers, although aimed at ameliorating these difficulties, do not succeed in providing a substantive resolution. One of the key aspects of our methodology is the accommodation of various model architectures across local clients. The technique is particularly concentrated on CNN architectures, specifically: (i) ResNet, (ii) EfficientNet, and (iii) MobileNetV3. This flexibility ensures that each local node can select a model that best fits its computational capabilities and specific data requirements. Such diversity in model architecture is crucial for the adaptability and personalization of the FL system.

9.2.1 Local Supervision and Self-supervision Representation Learning

Within the methodology described in this research, there are two learning paradigms integrated, that are applied to distinct subsets of the datasets. Every client in the FL system has access to three distinct sets of data in terms of distribution: a test set, a private set, and a shared set. To foster a collaborative learning environment, all clients have access to the shared set, which is an image collection that is used for training. Conversely, the private set is a unique group of images for each client that is extracted from the original training dataset following the subtraction of the images assigned to the shared set. The training scheme applied to the private set employs a fully supervised learning procedure, enabling each client to learn and adapt to the distinct attributes and patterns that derive from the local data. A hybrid learning model is adopted for the shared set, encompassing both supervised and self-supervised learning mechanisms. It utilizes the labels to encourage the learning of specific patterns within the set, while simultaneously generating a descriptor for each image. The objective is to align the individual descriptors with a collectively aggregated embedding which is constructed by all the clients, thus boosting the cohesiveness and efficacy of the learning process across the federated system. These shared resources play a pivotal role in the latter stages of the training process, particularly during the self-supervised learning phase. It serves as a unifying element, bridging the diverse learning experiences of individual nodes and aligning them with the broader, global dataset context. Figure 9.2 illustrates this representation-centric learning approach.

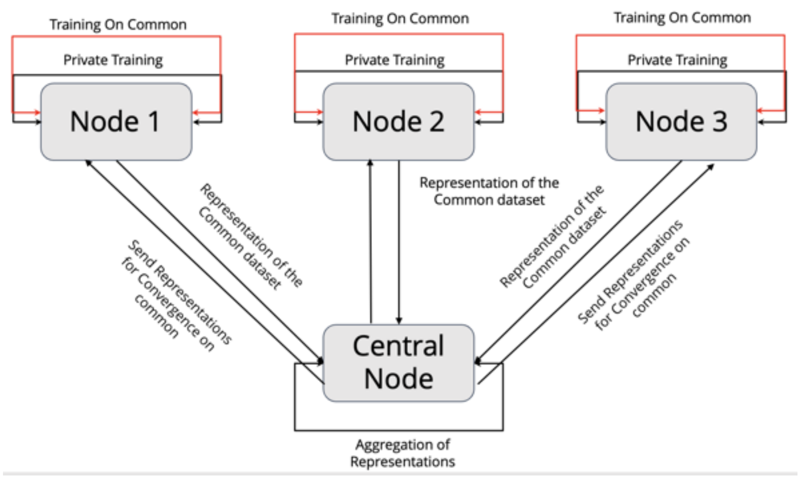

Training On Common
Private Training
Node 1
Training On Common
Private Training
Node 2
Training On Common
Private Training
Node 3
Representation of the Common dataset
Send Representations for Convergence on common
Representation of the Common dataset
Representation of the Common dataset
Send Representations for Convergence on common
Central Node
Aggregation of Representations

Figure 9.2: Representation-centric learning approach.

Grad-based Feature Extraction Module

The core of our excitation technique is grounded in the observation that the most influential features are highlighted by the gradients during the backpropagation process. Drawing inspiration from the insights provided in [225], where the authors underscore the significance of gradients in evaluating feature importance within their personalized FL system, the proposed methodology advances this understanding. Gradients within a model delineate the direction of optimization and effectively indicate the influence of each neural unit. The feature extraction method we propose builds upon this concept, harnessing the model's gradients relative to the ground truth to identify and extract pivotal features from each block's output. By selecting features that possess the largest absolute gradients, the algorithm ensures the inclusion of those with the most substantial impact on the model's predictive outcome, thereby enriching the feature descriptor with the most influential attributes for the task at hand. The formula 9.1 of the module is depicted below:

$$\text{grad}_j^b = \text{Top}k\left(\left|\frac{\partial p_j^c}{\partial F_j^b}\right|\right), k \in \{128, 256, 512, 1024\}, b \in \{1, 2, 3, 4\} \tag{9.1}$$

where b represents the block of the model, k number of features to be extracted, j the training sample, p_j^c the predictive output of the c-th category. This approach begins with initial training at each local node using its own private dataset, a critical phase for developing accurate data representations. These representations lay the groundwork for the method's subsequent phases. Following this initial training, every node processes the images from a shared subset. For each image, the algorithm selects the top features exhibiting the largest absolute gradients within each of the blocks of the network's feature map, as shown in the Algorithm 18 lines 6-12. It is these prominent features, considered pivotal for the image's representation, that are then forwarded to the central server. This process ensures that the most critical aspects of the data are emphasized and aggregated globally, enhancing the overall learning and representation capability of the system. In Figure 9.3, Grad-cam[226] was used to visually compare the gradient maps of each block across the network architectures

A comprehensive evaluation identified the most informative characteristics for constructing a meaningful feature descriptor, using targeted feature cropping from the feature maps and various adaptive pooling strategies, ultimately finding that the proposed Gradient-based Feature Extraction Module outperformed these techniques in feature selection efficacy.

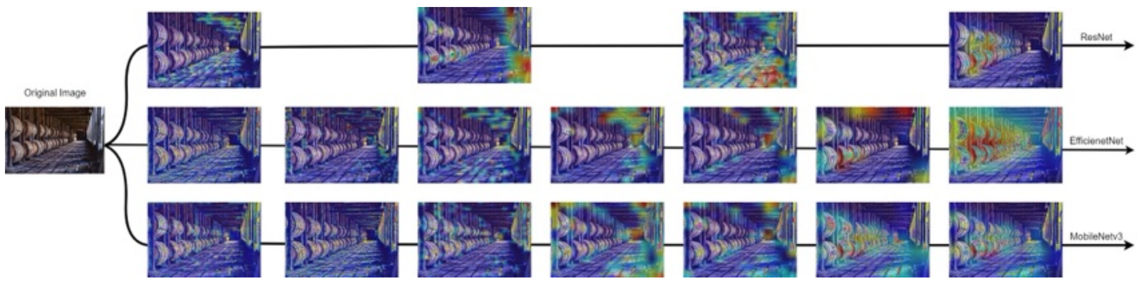


Figure 9.3: Visual representation of the gradient visualizations for three different neural network architectures applied to the same original image. On the left, the original image is depicted. Progressing to the right, the subsequent images represent gradient heatmaps as interpreted by ResNet, EfficientNet, and MobileNetV3, respectively. These heatmaps highlight areas of the image that contribute most significantly to the models' predictions, with warmer colors indicating higher gradient values and thus greater importance in the decision-making process of each network.

9.2.2 Model-Agnostic Global Aggregation

The central node in this architecture plays a vital role in aggregating the representations from each participating node. This aggregation focuses on the outputs from specific blocks or layers of the models, going beyond the individual differences in architectures. The process is designed to be model-agnostic, accommodating a wide variety of models without requiring uniformity. A significant departure from traditional approaches in our methodology is the avoidance of weight aggregation algorithms. Instead, we focus on a representation-centric aggregation approach. This shift ensures that the learning process is more coherent, communication efficient and effective, particularly suitable for environments with heterogeneous models. The ultimate goal is to align the local representations of the shared dataset with the global representation. The global representation is the aggregation of the local descriptors of each image in the shared subset, contributed by all participating clients in the FL system. This alignment enhances the model's ability to generalize and adapt to new data, improving its performance across the federated network. The hybrid training approach – combining supervised and self-supervised learning – ensures that local models are fine-tuned to both their specific data characteristics and the aggregated knowledge from the shared dataset. In conclusion, this proposed aggregation methodology offers a flexible and effective approach to managing diverse model architectures in FL. By focusing on representation learning and adopting a model-agnostic aggregation approach, it ensures efficient and robust learning across the federated network, paving the way for more adaptable and powerful FL systems.

Iterative Representation Refinement and Dual-Loss Aggregation

At the central server, an average representation for each image is computed by aggregating the feature information received from all clients. This global representation encapsulates the collective insights of all participating nodes, enriching the understanding of each image in the shared dataset. The training of the network then proceeds with a cosine similarity loss function. In each epoch of this training phase, two views of the data are considered: the global representation and the current local representation of the network being trained. The first epoch plays a pivotal role as it sets the baseline for the representations used in a knowledge distillation-inspired learning process. The representations for the subsequent epochs are then recalculated based on the top feature indices identified in the previous epoch. This iterative process is refined further by incorporating both cosine similarity and cross-entropy loss functions into the global aggregation step. This dual-loss approach allows the model to benefit from the strengths of both supervised and self-supervised learning paradigms, leading to more robust and well-rounded representations. By continuously refining the representations and the model through this iterative process, the FL system efficiently leverages the most relevant features of the data, enhancing the model's performance and adaptability to diverse datasets. Algorithm 18 provides a detailed illustration of the proposed aggregation technique in the suggested FL framework. The algorithm's goal is to reduce inconsistencies between the local and global features, promoting uniformity and coherence in the learned representations across the network. Upon successful alignment of the local models with the global representation, the updated models are aggregated at the central server, enhancing the global model with enriched insights from the network's distributed learning experience. This strategy ensures that local models not only perform well on their data but also contribute effectively to the collective intelligence of the federated system.

Image Descriptor - Feature Alignment

The core innovation of the proposed approach is based on the hypothesis that we can directly map blocks of features across different model architectures, owing to the similarity in the information they encode, and then aggregate them accordingly. This strategy suggests a more streamlined and potentially effective method for combining diverse models by capitalizing on the inherent parallels in their feature representations. However, in the journey of aggregating learned representations from different blocks of each node in our FL system, we encountered several significant challenges, particularly with the alignment metrics between node representations, especially with the implementation of cosine similarity as a measure of alignment. Unexpectedly, the use of cosine similarity not only failed to enhance the network's performance but rather led to a decrease in effectiveness. This issue was further complicated by the observation that for the majority of the layers, except the last one, the cosine similarity consistently equated to zero. The equation for cosine similarity is as follows:

$$\text{cosine similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{9.2}$$

where A and B are two vectors for which you are calculating the cosine similarity. The dot product of A and B is divided by the product of their magnitudes (or Euclidean norms). The magnitudes are calculated as the square root of the sum of the squared elements of each vector.

The construction of an image embedding derived from the entirety of a client's model, and the utilization of solely the last flattened embedding, did not yield a notable enhancement in accuracy. This outcome was observed despite each client's embedding being a high-dimensional 1920-vector, which was meticulously generated through the Gradient-based Feature Excitation Algorithm, a variance attributable to the disparate model architectures and the dimensions of the latent spaces involved.

The main descriptor's experimentation began with ResNet architectures—ResNet18, ResNet34, and ResNet50—due to their prevalence in FL literature and computational constraints. The technique encompasses the strategic extraction of features from all the blocks of the ResNet architectures to frame a thorough 1920-dimensional embedding, utilizing the Gradient-based Feature Extraction Module. This process initiates with the selection of 128 features from the initial main block, followed by the extraction of 256 features from the second block, 512 from the third, and culminating with 1024 from the fourth block. In particular, the process of feature elicitation serves as a pivotal step in harmonizing the local models with a central, aggregated global accumulation of the descriptors. It starts with an image passing through a neural network, which is segmented into distinct blocks, each responsible for extracting features at varying levels of complexity. The early blocks, such as *Block1*, typically capture elementary features like edges, while the deeper blocks, such as *Block4*, discern more intricate patterns. The outputs of these blocks undergo a selective process where a specific number of features are chosen based on their activation levels, which are indicative of their importance. These chosen features are then combined, with the deeper blocks contributing a larger share of features, reflecting their increased complexity and importance in characterizing the image. This process is illustrated in Figure 9.4. The reasoning behind this specific excitation approach is grounded in empirical findings. Experiments have revealed that similarity metrics for features in the initial layers of both trained and untrained models are remarkably high, indicating homogeneity in the primitive features extracted by these layers. Consequently, as the network delves deeper, the need for capturing a broader and more complex range of features grows, prompting the selection of larger feature maps from the deeper layers—(128, 256, 512, 1024) respectively—to ensure a comprehensive representation.

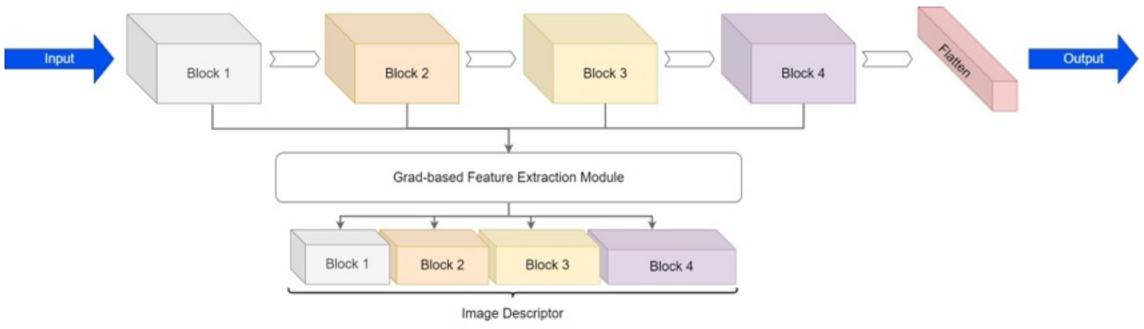


Figure 9.4: Diagram of the descriptor construction process from a ResNet architecture, illustrating the sequential extraction of features from the four distinct blocks of the model. The feature maps are processed through a Feature Extraction Module, resulting in a composite image descriptor that encapsulates multi-scale representations of the input.

Algorithm 18 Federated Learning with Grad-based Feature Extraction

```
Require: $D_i$ local datasets, $f^i(\theta)$ model of the local clients, $C$ common subset, $R$ total
    communication rounds, $E$ total epochs, $lr$ learning rate
1: for $r \leftarrow 1$ to $R$ do
2:     Local training
3:     for $i \leftarrow 1$ to $N$ do
4:         $f^i(\theta) \leftarrow \text{train}(f^i(\theta), D_i, L_{CE})$
5:     end for
6:     Grad-based Feature Extraction Module
7:     for $i \leftarrow 1$ to $N$ do
8:         for all $b$ in $f^i(\theta)_{blocks}$ do
9:             $b_{\text{grad}} \leftarrow \frac{\partial P_{D_i}}{\partial f^i(\theta)_{b,(wh)}}$
10:            $F^i_E \leftarrow \text{ExtractTopK}(|b_{\text{grad}}|), K \in \{128, 256, 512, 1024\}$
11:        end for
12:    end for
13:    Aggregation on Central server
14:    for $j \leftarrow 1$ to $C$ do
15:        $F_{Ej} \leftarrow \frac{1}{N} \sum_i^N F^i_{Ej}$
16:    end for
17:    for $i \leftarrow 1$ to $N$ do
18:        for $e \leftarrow 1$ to $E$ do
19:            for $j \leftarrow 1$ to $C$ do
20:                $F^i_{Ej} \leftarrow$ Grad-based Feature Extraction Module$(f^i(\theta), P_j)$
21:                $z^j_{\text{local}} \leftarrow f^i(\theta)(\mathrm{x}_j)$
22:                $L \leftarrow \alpha L_{CE}(z^j_{\text{local}}, P_j) + (1-\alpha) L_{COS}(F^j_{iE}, F^E_j)$
23:                $\theta \leftarrow \theta - lr \cdot \nabla L$
24:            end for
25:        end for
26:    end for
27: end for
```

In the Algorithm 18 presented before, L_{CE} and L_{COS} represent the Cross-Entropy and Cosine Similarity loss functions that are employed in the training process of the proposed method. b_{grad} is the gradient map of the block of the models, P_{D_i} represent the predictions of the i-th dataset of the i-th client's model. The variables w and h are the spatial dimensions of the feature maps, F^i_E represent the extracted features from the i-th client's model, while F_{E_j} is the aggregated descriptor manufactured in the main server and z_{local} is the prediction of the model of the x_j image and finally, the factor θ denote the weights of the model.

Harmonizing diverse architecture through block alignment strategies:

Expanding the scope of architectures to include EfficientNet and MobileNetV3, specifically EfficientNetB0, EfficientNetB1 and MobileNetV3-Small, MobileNetV3-Large, alongside the ResNet family, a nuanced approach to integration is necessitated by the variance in the total number of blocks within these models—EfficientNet and MobileNetV3 comprising seven blocks in contrast to ResNet's four. The incorporation of a diverse array of architectures, demanded a sophisticated alignment policy to accommodate the variations in block numbers—seven in EfficientNet and MobileNetV3 as opposed to four in ResNet. This alignment was meticulously conducted through an analysis of representational similarity and angular divergence among the models, utilizing cosine similarity metrics to achieve uniformity in representation spaces. The process entailed the synchronization of blocks that demonstrated the least angular divergence, taking into account the orientation and dimensionality of their feature maps. Despite challenges in direct comparison due to activation function and dimensionality differences, absolute feature grouping and KL-Divergence analysis enabled precise block alignment across architectures, fostering a unified FL framework. Consequently, the harmonization of block connections across ResNet, EfficientNet, and MobileNetV3 architectures is achieved, paving the way for a cohesive and aligned FL system that leverages the strengths of diverse model architectures while maintaining the integrity of their unique representational capacities.

9.3 Experimentation in Various Architectural Scenarios

The exploration of different model architectures in FL aims to understand how diverse neural network structures can affect the learning process when distributed across various nodes. In the series of experiments conducted, the focus was on integrating three different core CNN-based architectures ResNet, EfficientNet, and MobileNetV3 to investigate their performance in FL settings. The initial set of experiments deployed various versions of ResNet, specifically ResNet18, ResNet34, and ResNet50. Worth noting that, memory constraints limited the exploration to all these three models, highlighting the practical challenges of deploying larger and more complex networks in FL. More details on challenges related to architecture variability and further experimental evaluation can be found in Appendix F.

9.3.1 Experiment Setup

Datasets

The datasets that are used to validate the Fl system are CIFAR-10, CIFAR-100[123] and MNIST[194]. The distribution of the data to the clients is in an IID format, where each client contain the same number of images per class. Regarding the shared subset, experiments were conducted in a range 5000 - 10000 data instances, to discover the optimal number. It was revealed that despite the fact that the alignment process performed better the bigger the share subset was, the remaining data that were divided to the clients proved insufficient for the models to effectively capture their unique patterns, so the total number of the shared division of the dataset is set to 5000.

Data Augmentations

In the preprocessing phase of the study, a series of image augmentation techniques were implemented to enhance the diversity of the training dataset. The specific augmentations applied include horizontal and vertical flip, random crop, brightness adjustment of $+/-$ 20% and Gaussian blur.

Distribution of the Networks to Clients

The models are allocated to clients utilizing a standardized approach, where each variation of the networks is selected according to a uniform distribution method, with a requirement that the method contains at least one of each different network architectures.

Hyperparameters

The total communication rounds of our experiments are set to 10, while the epochs of training of each round are set to 25, the batch size is set to 128, the initial learning rate of the models was 0.01, using One-Cycle LR[227] with the minimum learning rate being 0.0001 and the optimized of the networks was the Adam[191]. The total number of variations to the number of clients is 5, 10, 20.

Compared Methodologies

To facilitate a meticulous comparative analysis of the proposed framework's performance, we have chosen benchmark methods grounded in the principles of representation learning. The FL schemes selected for this purpose include PerFCL[228], FedCon[173], MOON[135], FedSimCLR, FedCA and FedSimSiam[229]. However, it is important to highlight that the proposed approach diverges from the existing state-of-the-art methodologies by incorporating a varied architecture at each federated node. This distinction is a critical factor that should be considered in any comparative analysis below.

9.3.2 Evaluation and Results

The initial goal was to evaluate the efficacy of various ResNet models disseminated among the clients. The resulting performance is the mean accuracy of the corresponding method across all the clients on the test set of each database. For the evaluation metric, the accuracy is the mean accuracy of the corresponding method across all the clients on the validation set of each database. $FedHARM_{res}$ denotes the adaptation of this approach, incorporating ResNet architectures for client distribution, and it outperformed all competing methodologies on the CIFAR-10 dataset across every client configuration as shown in Table 9.1. A slight decline in performance was observed with an increase in the number of clients, attributable to the correspondingly reduced dataset available to each client. The integration of the alignment module significantly enhanced the framework's effectiveness, demonstrating its utility and success by achieving an accuracy of 83.87% in a scenario with 5 clients and 81.51% in a setting with 10 clients. The added value is further underscored by the fact that, unlike traditional FL strategies, the conventional solution avoids using weight averaging algorithms and instead opts for dominant feature utilization, showcasing its flexibility in extracting knowledge from a variety of model architectures.

Transitioning to the CIFAR-100 dataset, the proposed method exhibited superior performance in the 5-client configuration compared to the benchmark methods achieving 52.38% accuracy, but did not achieve the same level of success in the settings with 10 and 20 clients. This reduction in performance can be attributed to the insufficient amount of data per client in the local datasets, which hindered the creation of a robust embedding capable of accurately representing the data instances across the 100 classes of the dataset.

Table 9.1: Accuracy (%) comparison of FL Methods on CIFAR-10 and CIFAR-100 datasets. The validation was performed on the test set of each database, and the resulting number is the mean accuracy across the clients.

Method	CIFAR10			CIFAR100		
	5 Clients	10 Clients	20 Clients	5 Clients	10 Clients	20 Clients
FedCon	-	81.47	-	-	-	-
PerFCL	-	75.5	75.1	-	**61.2**	**58.6**
MOON	-	74.2	73.6	-	60	57.5
FedSimCLR	68.1	-	-	39.75	-	-
FedCA	71.25	-	-	43.30	-	-
FedSimSiam	76.27	-	-	49.79	-	-
$\text{FedHARM}_{\text{res}}$	82.88	81.51	78.01	52.38	52.54	38.89
$\text{FedHARM}_{\text{rem}}$	**83.87**	**82.03**	**79.76**	**53.9**	51.49	40.03

In the second set of experiments, the focus shifted to EfficientNet and MobileNetV3 models to accompany the ResNet models, where this variation is portrayed as $FedHARM_{rem}$. These architectures are known for their efficiency and performance on mobile devices, making them an interesting choice for FL scenarios. In the context of the CIFAR-10 dataset, the $FedHARM_{rem}$ variation outperforms all other methodologies evaluated, including $FedHARM_{res}$, demonstrating superior efficacy across various client settings. This enhanced performance is substantiated by considering that architectures such as EfficientNet and MobileNetV3 exhibit superior performance compared to the variations of ResNet networks. The augmentation in the quality of representations learned at the local dataset level indicates that the overall descriptors of the shared subset have facilitated the creation of a more significant embedding. Consequently, as it is depicted in Table 9.1 this has resulted in notable accuracy rates of 83.87%, 82.03%, and 79.76% in settings with 5, 10, and 20 clients, respectively. This outcome underscores the effectiveness of integrating advanced neural network architectures for improving the robustness and representational capacity of embeddings in distributed learning environments.

Table 9.2: Accuracy Results of the Proposed Method on the MNIST Dataset against the FedCon system

Method	Accuray (%)		
	5 Clients	10 Clients	20 Clients
$FedCon$	-	98.08	-
$FedHARM_{res}$	98.79	98.42	97.93
$FedHARM_{rem}$	98.94	98.65	98.54

Due to the noted lack of comparable outcomes for the MNIST database, an assessment was carried out, as shown in Table 9.2, in order to compare our framework's performance with FedCon's in a 10-client setting. In this analysis, the $FedHARM_{rem}$ variation demonstrated superior performance, achieving an accuracy of 98.62%. Furthermore, in configurations involving 5 and 20 clients, the $FedHARM_{res}$ variation recorded accuracies of 98.79% and 97.02%, respectively. In contrast, the $FedHARM_{rem}$ variation exhibited even greater heterogeneity in its performance, achieving remarkable accuracies of 98.97% in the 5-client setup and 98.01% in the 20-client configuration. This detailed comparison underscores the robustness and adaptability of the $FedHARM$ variations across different benchmark datasets.

9.4 Insights and Contributions

The study presents a comprehensive exploration of model architecture variability within the FL framework, introducing innovative strategies to incorporate diverse CNNs, specifically ResNet, EfficientNet, and MobileNet. It highlights the challenges and proposes solutions for efficient model aggregation and communication, emphasizing representation learning and model-agnostic frameworks. The proposed $FedHARM$ approach, especially $FedHARM_{rem}$, significantly outperforms existing methods in CIFAR-10, CIFAR-100 and MNIST IID datasets evaluations, demonstrating the effectiveness of representation-centric and model agnostic aggregation across different architectures. This research paves the way for more adaptable, efficient, and privacy-preserving FL systems, capable of leveraging the strengths of different architectures to improve learning outcomes across decentralized networks. The study sets the stage for extensive future research, with numerous potential experiments to further enhance FL, involving an in-depth analysis of data heterogeneity among clients, and integrating a wider array of network architectures.

Chapter 10

Study 7: Trustworthiness in Federated Learning

In this study, the research trajectory takes a pragmatic turn, focusing on a comprehensive evaluation of the practical challenges and obstacles inherent in the FL ecosystem. This horizontal study, which spans across the various themes and methodologies explored in the previous five studies, aims to bridge the gap between theoretical advancements and real-world applications of FL. The core of this study lies in designing and implementing a system adaptation pipeline, meticulously crafted to facilitate the integration of diverse AI-based tools into the FL framework.

This study embarks on a journey to test the FL system under conditions that mimic real-world scenarios. By employing a distributed hardware infrastructure, the study examines the FL system's resilience and adaptability against a range of challenges, including tool deployment intricacies, data heterogeneity complexities, and susceptibility to privacy breaches. These examinations are conducted across a variety of tasks and data types, providing a holistic understanding of the FL system's capabilities and limitations.

A carefully curated selection of AI-based tools and corresponding datasets forms the backbone of the experiments. These tools and datasets are distributed across edge devices, ensuring that the test conditions closely mirror the diversity and unpredictability of real-world environments. This approach enables a thorough validation of the FL system across multiple scenarios, offering a realistic appraisal of its performance.

The outcomes of the experiments conducted as part of this study are pivotal. They not only provide valuable insights into the performance of models under varied FL conditions but also shed light on potential issues and limitations encountered during the FL process. This comprehensive analysis contributes significantly to understanding the practical challenges of FL, identifying areas that require further research and development.

In essence, this study stands as a critical piece of research that connects the theoretical underpinnings of FL with its practical implications. By assessing the viability of FL in real-world settings, this study complements and extends the contributions of the preceding studies, collectively advancing the field of FL towards more robust, efficient, and practical solutions.

10.1 Overview of the Study

The effectiveness of AI models is closely tied to the quality and quantity of data utilized in their training. With the abundance of information available on various devices, including mobile and servers, there is an opportunity to glean valuable insights. FL is an innovative ML approach that leverages decentralized data and computational resources to deliver more tailored and flexible applications while upholding the privacy of users and organizations. By utilizing FL, it is possible to uphold data protection laws and regulations, thereby ensuring that privacy is not compromised. FL has demonstrated exceptional results in numerous analysis tasks, such as image classification, object detection and action recognition [185, 186, 187], indicating its robustness and effectiveness in these areas. To elaborate further, FL allows data to remain on individual devices (cross-device) or servers (cross-silo) rather than being centralized, thereby avoiding potential privacy breaches. This approach enables users to keep their data safe and secure while still contributing to the training of ML models. Moreover, FL can handle large and diverse datasets, which often lead to improved accuracy in analysis tasks. As a result, FL has emerged as a promising technique that can revolutionize the way AI models are trained and deployed, all while maintaining the privacy and security of users' data. FL has distinct characteristics when compared to distributed learning. Firstly, communication in FL is often slower and less stable. Secondly, FL involves participants with heterogeneous devices, which vary in terms of their computing capabilities. Lastly, privacy and security are emphasized more in FL. Although most studies assume that both participants and servers are trustworthy, this may not always be the case in reality.

Several studies have been conducted to offer a comprehensive understanding of the current state of FL, its potential applications, and the ongoing efforts to overcome its challenges and limitations [46, 80, 230, 20, 5, 231, 9, 232]. These studies explore various methods and techniques, such as optimization algorithms for efficient model aggregation, privacy-preserving mechanisms, and adaptive learning strategies. A detailed analysis of these topics is provided in Section 3. While these works acknowledge the potential benefits of FL, such as collaborative ML across decentralized data sources, privacy preservation, and empowering edge devices, they predominantly focus on the theoretical aspects. However, they lack practical and experimental analysis. Therefore, further research and development are needed to incorporate more practical implementations and experimental evaluations to validate the theoretical findings and provide real-world insights into the effectiveness and scalability of existing FL systems [233, 234, 235].

However, implementing and deploying FL systems can be even more challenging due to a variety of factors such as high communication costs, heterogeneity in data, regulations, and tasks across different participating organizations, the autonomy and redundancy of processing nodes, and the potential for data poisoning incidents. This study aims to address these challenges by analyzing and validating the FL system architecture against these key issues. The contributions of this work can be categorized as follows:

(a) Reporting and comprehensively describing the main challenges faced by FL systems: This involves identifying and discussing the challenges that FL systems are likely to encounter during deployment, including data-related challenges such as data heterogeneity and data privacy concerns, as well as challenges related to system architecture and design.

(b) Integrating different tools in the FL system: This involves incorporating various AI-based tools and technologies into the FL system architecture to facilitate real-world FL scenarios using distributed hardware infrastructure. This could include implementing data aggregation techniques, incorporating secure communication protocols, and integrating privacy-preserving methods for data sharing and training.

(c) Evaluating the FL system against reported challenges: This objective involves conducting dedicated experiments for different tasks and data types to evaluate the FL system's performance and accuracy against the challenges identified in objective (a). This will involve assessing the system's ability to handle data heterogeneity, maintain data privacy, and handle data poisoning incidents.

(d) Providing an overall assessment of the developed FL system: The final objective of this work is to provide an overall assessment of the FL system architecture developed through this study. This includes analyzing the system's scalability, efficiency, and robustness against the challenges identified in objectives (a) to (c).

Overall, this study seeks to contribute to the overall understanding of the challenges faced by FL systems and provide solutions to improve their efficiency, scalability, and security. The outcomes of this study could have significant implications for the development and deployment of FL systems in various domains, including healthcare, finance, robotics and education. Additional details on the best practices and guidelines for implementation can be found in Appendix B.

10.2 Practical Challenges and Solutions

10.2.1 FL Strategies

Chapter 10.1 presents a detailed description of the FL data-specific challenges related to local data distribution and dataset size heterogeneity. To tackle data heterogeneity in the FL process, multiple FL algorithms and strategies have been developed, including more sophisticated aggregation methods and strategies for more efficient local training. In our experiments, we evaluate the performance and the convergence of the global aggregated model by utilising the three more well-known and widely used FL strategies, the FedAvg, FedOpt and FedProx, representing both the classic baselines and current state-of-the-art. Each of these methods focuses on handling data heterogeneities considering label distribution skew, feature distribution skew and quantity skew that are highly related to datasets of the reported AI tools, exploring the effectiveness of these methods on different types of data sources such as images, audio, and text.

10.2.2 FL Security Mechanisms

To enhance privacy preservation in the FL system, a Differential Privacy (DP) method is applied as a security mechanism during the experiments. DP is one of the most well-known and widely used approaches for privacy preservation, which randomises part of the system's behaviour to provide privacy. Additionally, with DP the users can quantify the level of privacy of the system by selecting the appropriate parameters that achieve the best trade-off between the FL model performance and privacy level. Specifically, at tests the SVT (Sparse Vector Technique) Privacy protocol is utilised since it is characterised as a fundamental method to perform DP. The SVT Privacy is applied as a filter in the FL set-up of NVFlare, while the definition of it as well as its necessary parameters is done by the configuration file of the client. The chosen parameters of the SVT Privacy method are listed in 'Table 10.1 for each experiment.

Table 10.1: Parameters of SVT Privacy for each experiment.

Experiment ID	**Fraction**	**Epsilon**	**Noise_var**
Face-ReID	0.6	0.1	1
VSR	0.6	0.001	0.1, 1
NERC	0.6	0.001, 0.1	0.1, 0.5, 1
ASR	0.6	0.001, 0.1	0.1, 0.5, 1

10.2.3 Data Management

A typical lifecycle of ML models entails their training on example data. In the typical workflow, a model is trained on large amounts of local data and tested for performance on a smaller, disjoint, data set. The training itself comprises several rounds of model runs (*i.e.*, inference) and adjustments in order to converge to an acceptable performance. On the other hand, FL is a distributed ML paradigm where different sets of data, typically disjoint among them, are used at multiple self-sustained training locations, *i.e.*, training nodes. A schematic representation is drafted in Figure 10.1. Different versions of local models are fine-tuned at each federated party, while the final global model is obtained after their aggregation.

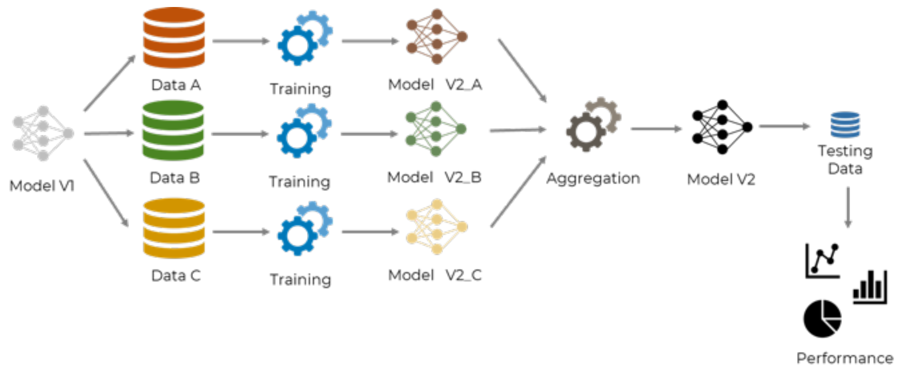


Figure 10.1: Schematic workflow of a basic FL principle.

In the FL setting, however, the aggregation of multiple models attenuates the differences among the local models and thus their adaptation to the local data. Multiple rounds of local training are necessary to reach convergence of the global model. The adaptation of a model can thus be tracked at each round. The ML training workflow uses a distinct set of data to train or assess the evolution of the training process. We distinguish between:

–**Training set:** A set of examples used for learning, *i.e.* fitting the best parameters of the ML model (classifier, detector, *etc.*). This is the set of positive and negative examples available at each federated party.

–**Validation set:** A set of examples, disjoint from the training set, used to assess the model's actual performance, at each epoch of the local training process. It is used by the researchers to examine the training process for possible irregularities and adjustment of the training optimization parameters. Similarly, it can be used by researchers to tune any hyper-parameters of an ML model, such as the number of hidden units in a neural network or any other specific internal settings.

–**Test set:** A set of examples used only to assess the performance of a fully specified classifier. During the FL setup, the different sets introduced aforehand must be defined at each federated party, as illustrated in Figure 10.2.

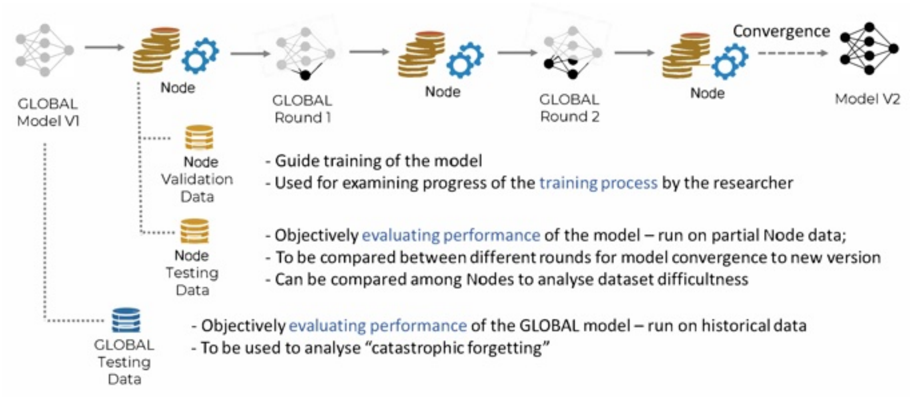

GLOBAL Model V1
Node
GLOBAL Round 1
Node
GLOBAL Round 2
Node
Convergence
Model V2
Node Validation Data
- Guide training of the model
- Used for examining progress of the training process by the researcher
Node Testing Data
- Objectively evaluating performance of the model – run on partial Node data;
- To be compared between different rounds for model convergence to new version
- Can be compared among Nodes to analyse dataset difficultness
GLOBAL Testing Data
- Objectively evaluating performance of the GLOBAL model – run on historical data
- To be used to analyse "catastrophic forgetting"

Figure 10.2: The use of different datasets during the FL training.

In terms of the FL testing of this study, we pursue the testing on local nodes data as well as on the GLOBAL dataset at each round. While the GLOBAL testing set evaluates a possible drift in the performance of the initial task, the local datasets give us a good estimation of how the models are adapting to new data. Moreover, two options for testing on local data are available: a) testing before the local training, *i.e.* evaluating the external knowledge on the local conditions; and b) testing after the local training, *i.e.* evaluating the adaptability of the global model to the local conditions. With respect to the testing of the global model, these two options refer to a) testing the global model after aggregation, and b) testing the global model before aggregation. The training performance is monitored using the Weights and Biases environment[1]. This platform enables research teams to simultaneously observe the training process, providing a unified perspective on it. Specifically, it allows for the tracking of training loss and accuracy at every round and across each federated node, as well as monitoring the overall performance of the global model, as illustrated in Figure 10.3.

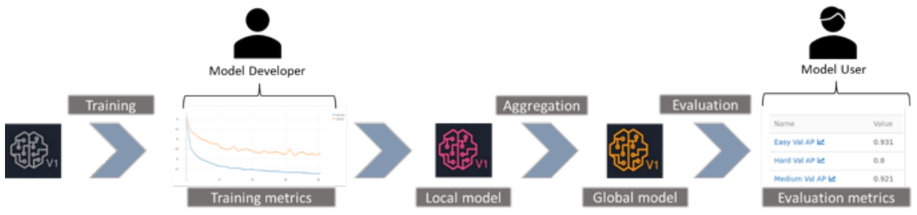


Figure 10.3: The use of Weight and Biases environment for tracking the training process.

10.3 Case Studies of FL Applications in Different Domains

10.3.1 Face Re-ID

Datasets: In the Federated setting, experiments on the Face Re-Identification tool involve using different datasets for both training and testing the model at each federated

[1]https://wandb.ai/site

node. These datasets are carefully selected to cater to various scenarios and final use cases. The training dataset is the *Ms1m-Retinanet*, a public dataset published in [236]. The dataset is derived from the Ms1m dataset, which comprises images of celebrities. To obtain a cleaned version, the images were cropped to a size of $112x112$ using Retinanet and 5 facial landmarks. A pre-trained model based on ArcFace [237] and ethnicity-specific annotators was then used for semi-automatic refinement. Ultimately, our training dataset consisted of around *50,000* identities, which were equally distributed among the federated nodes, with *10,000* identities per node. The experiments were performed using five nodes.

The goal of our experiment was to evaluate the adaptability of the face re-identification model to face occlusions within a federated framework, under multiple heterogeneity settings. To create an occluded version of the dataset, the *EyesOcclusionGenerator* library was used to add black rectangles over the eyes of the identities in the original dataset. The final training dataset comprised these two versions of the dataset, with and without eye occlusions. During training, the image of the identity was randomly selected from one of the datasets, with a certain probability.

To introduce heterogeneity in the experiments, one of the two datasets (Ms1m-Retinanet with and without occlusions) was exclusively used for training in some of the nodes. This was achieved by adjusting the probability of selecting the dataset for the next batch during training. The specific configuration per training node is given in Table 10.2. The training was conducted from scratch, with the initial random model being adapted to the previously unseen examples, and executed over *8* rounds with *10* epochs.

Table 10.2: Training data distributions among clients for different levels of heterogeneity.

	Site-1	Site-2	Site-3	Site-4	Site-5
Low	50%[1]	50%	50%	50%	50%
Medium	50%	50%	50%	100%	0%
High	50%	100%	100%	0%	0%

[1] Represents the percentage of occlusion in the training set.

During the testing phase, and similarly to the test setting of the ArcFace model [237], we employed multiple datasets, such as LFW, CFP and $Age{-}DB{-}30$ for checking the convergence on slightly different domain. The test datasets consisted of both, the original as well as eye occluded versions. For the sake of simplicity, LFW denoted as D^1, $CFP{-}FP$ denoted as D^2, $CFP{-}FF$ denoted as D^3 and $Age{-}DB{-}30$ denoted as D^4 (with their occluded versions denoted as D^1_{occl}, D^2_{occl}, D^3_{occl} and D^4_{occl} respectively). The occluded versions of the testing datasets were produced using the same library used for the training dataset. These testing datasets were distributed across the individual training nodes, with each node managing a distinct testing dataset as shown in Table 10.3.

Table 10.3: Testing data distributions among clients.

Site-1	Site-2	Site-3	Site-4	Site-5	Dummy
D^1 D^1_{occl}	D^2 D^2_{occl}	D^2 D^2_{occl}	D^1 D^1_{occl}	D^3 D^3_{occl}	D^4 D^4_{occl}

Quantitative: We test the face re-identification model performance using the validation protocols of well-known public face-recognition datasets (*i.e.* verification performance of a pair of images). The results of the experiments for the face re-identification tool are shown in the following Table 10.4.

Table 10.4: Verification performance evaluation of the centrally trained and the FL aggregated model on the global and the local datasets, where *low*, *medium* and *high*, denotes the level of heterogeneity.

Dataset	Centralised	$FedAvg_{low}$	$FedAvg_{medium}$	$FedAvg_{high}$	$FedAvg_{DP}$[1]
D^4	0.978	0.852	0.85	**0.858**	N/A
D^4_{occl}	0.834	**0.708**	0.686	0.683	N/A
D^3	0.998	**0.966**	**0.966**	0.964	N/A
D^3_{occl}	0.936	0.812	0.818	**0.826**	N/A
D^2	0.974	0.88	0.88	**0.885**	0.806
D^2_{occl}	0.781	0.703	0.701	**0.704**	0.642
D^1	0.997	0.968	0.967	**0.978**	0.946
D^1_{occl}	0.969	0.881	**0.891**	0.887	0.839

[1] Equally split dataset and svt_privacy.

It should be noted that the last configuration used for the distributed training encountered issues with the connectivity of some of the nodes, resulting in a lack of results for some testing datasets. The provided Figures 10.4 (a), (b), and (c) offer a comparison between the local training models obtained from each node and the global model for each heterogeneity experiment. The accuracy of each model was evaluated on the local test dataset, with identical results for nodes 1 and 4, as well as 3 and 5. The graphs suggest that there is a discernible variance in the accuracy of models trained on nodes that solely utilize occluded data, versus those that do not, as per the various heterogeneity settings outlined in Table 10.2. This difference is more evident in Figures 10.5 (a) and (b), which depict non-occluded and occluded images, respectively.

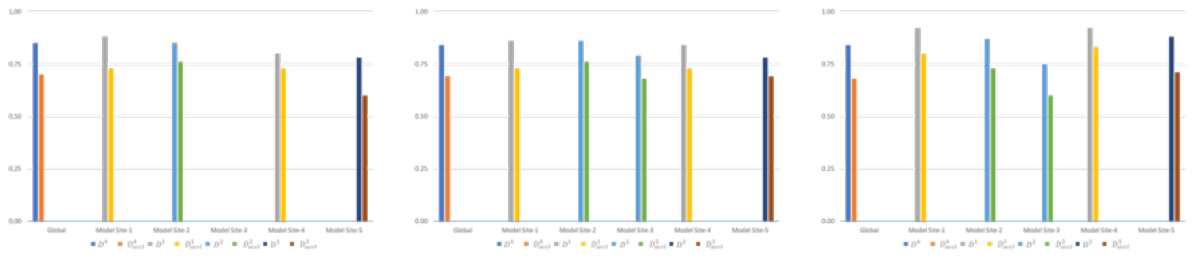
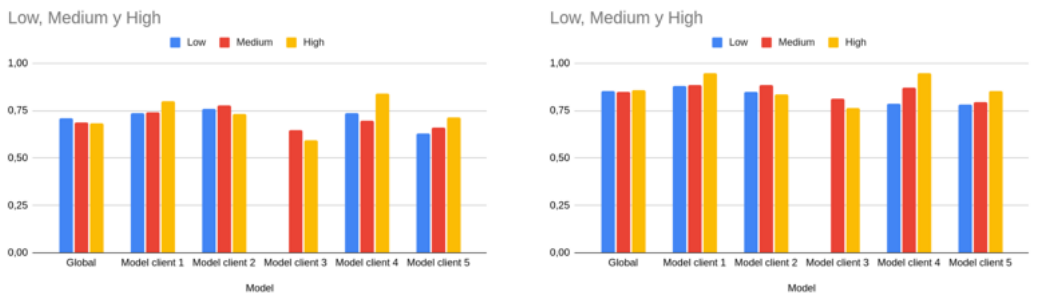

a) b) c)

Figure 10.4: (**a**) Low-, (**b**) Medium- and (**c**) High-heterogeneity results of global and local models at each node (ordered as dummy, and nodes 1 to 5). Note that the results for node 3 are missing within the low heterogeneity setting (a).



a) b)

Figure 10.5: Testing results comparing different heterogeneity settings, for (**a**) occluded and (**b**) non-occluded images. The results are grouped and ordered from dummy to nodes 1 to 5.

The graphs provided above enable us to compare the accuracy of the local models from each node within the local dataset across the different heterogeneity settings outlined in Table 10.2. Site 1's model was trained with 50% occluded data in each heterogeneity experiment, resulting in similar results. However, minor differences suggest the influence of other nodes on the training of the aggregated model utilized in each round. Site 2, in contrast, trains solely with occluded data in the high heterogeneity experiment, resulting in a slight performance decay. Site 3 shows a similar pattern to Site-2, although the low heterogeneity setting was incomplete due to technical connectivity problems. Site 4's model is trained solely with occluded data in the medium heterogeneity experiment and with only non-occluded data in the high heterogeneity experiment. This results in better performance on the non-occluded version of the dataset, but worse on the occluded version in the medium heterogeneity experiment. However, in the high heterogeneity case, the model performs better in both the non-occluded and occluded versions of the testing dataset. Finally, Site 5 trains exclusively with non-occluded data in the medium and high heterogeneity experiments. The model performs better on both the non-occluded and occluded versions of the testing dataset compared to training with 50% occluded data.

Qualitative: The experimental results indicate that the FL framework produces inferior results compared to local training. The variance in accuracy in some cases may be due to longer training times without the FL framework. Interestingly, the results for different data heterogeneities were similar, suggesting that heterogeneity had minimal impact on training. The LFW testing dataset achieved the highest accuracy for both the occluded and non-occluded versions. Regarding the security preserving mechanism experiment, due to node connectivity issues, the results cannot be accurately compared to previous experiments since the number of nodes was reduced. However, it was observed that the model converged, which is not the case with other tools such as NERC under similar circumstances. It is important to note that the results for this experiment were only obtained for some of the testing datasets.

Video Super Resolution

Datasets: For the Video Super Resolution (VSR) task, the dataset used was the Vimeo90k [238]. This dataset is of high quality and contains a large variety of scenes and actions, with video clips of a fixed resolution of 448x256. The upscaling factor used for the experiments was set to 4. The data heterogeneity type that was considered for the VSR task was quantity skew. This type of heterogeneity aims to simulate unbalanced data distribution by allocating a different number of video clips among the FL clients. To explore different levels of size heterogeneity, various splitting scenarios were designed.

Table 10.5 presents the data distributions among the five clients based on different levels of size heterogeneity. The clients are named Site-1, Site-2, and so on, with each corresponding to a specific server. The *Dummy* node is included only for evaluation purposes and does not contain any training data. In the low-level heterogeneity case, all clients have a similar number of LR-HR video clip pairs for training. However, in the high heterogeneity case, the number of samples for each client can be significantly different, resulting in an imbalanced distribution of data.

Table 10.5: Training data distributions for different levels of heterogeneity.

	Site-1	Site-2	Site-3	Site-4	Site-5	*Dummy*
Low	5470	5967	4560	4177	4122	N/A
High	11470	4072	824	1344	6490	N/A
Validation[1]	760	624	471	462	347	1934

[1] Validation data distribution is the same for both heterogeneity scenarios.

Two types of evaluation datasets were created for the VSR task: local and global evaluation datasets. Each FL client has its own local validation/test dataset, which is distinct from the local training and validation datasets of other clients. Additionally, a global validation dataset was created, located on the '*Dummy*' node, which contains video clips with varied content and actions and a more balanced distribution for evaluating the performance and generalization of both locally trained models and the globally aggregated model. The table displays the number of samples per local dataset and the size of the global dataset, which remain constant across different levels of heterogeneity.

Quantitative: The VSR task was evaluated using the FedAvg FL method, and the results are presented in Table 10.6. The evaluation was based on the PSNR metric (dB) on the global dataset for both the centrally trained model and the globally aggregated model for different levels of heterogeneity. The Vimeo90k dataset was used for the experiments, with an upscaling factor of 4 and the data heterogeneity type being quantity skew. The locally trained models were evaluated on their respective local validation/test datasets, while a global validation dataset was used to evaluate the performance and generalization of both the locally trained models and the globally aggregated one. The centrally trained model had the highest performance and served as the baseline. The FL global model, obtained by aggregating the local models' weights, achieved similar performance to the central training, with the performance gap being wider in the high heterogeneity case. These results indicate that the FL approach can successfully handle low to medium levels of data distribution heterogeneity utilizing a simple aggregation scheme (FedAvg), but for a higher level of heterogeneity where the gap between the different local optimal points is larger, more sophisticated learning strategies are needed.

Table 10.6: Performance comparison between the centrally trained model w/o FL (baseline) and the globally aggregated model on the global dataset for different FL strategies and high level of heterogeneity.

	Centralised	$FedAvg_{global}$	$FedProx_{global}$	$FedOpt_{global}$
Low	36.15	35.87	-	-
High	36.15	35.60	35.63	35.71

To address the issue of high heterogeneity in the size of local datasets, more advanced FL algorithms and strategies were employed in the experiments. Specifically, the *FedOpt* and *FedProx* methods were used. These two learning strategies have been extensively used to tackle heterogeneity in federated networks, requiring minor modifications in the aggregation scheme, and enabling an easy integration into the existing framework. FedProx adds a regularization term to the local objective minimization algorithm to reduce the shift between the local and global objectives, while FedOpt enables the use of adaptive optimizers on the server's side to improve model convergence. In our experiments, we want to investigate if these two long-established methods that follow different approaches can handle a realistic FL scenario in practice. Table 10.6 provides a comparison of the evaluation results for the different FL strategies on the global VSR dataset. As shown, the FedOpt strategy can more efficiently handle the quantity skewness than the FedProx which achieves a very small improvement. This result can be justified by assuming that the regularization term of the FedProx method is more suitable for objective shifts resulting from the label or feature distribution skew. Furthermore, we conducted an experiment to explore the impact of a privacy-preserving mechanism, specifically the SVT method, on the performance of the global model (see Table 10.7). In this experiment, a low-level heterogeneity approach was used to investigate the effects of the privacy mechanism only on the global model's performance. The results show that the SVT method with a noise_var of 0.1 reduces the model's performance by almost $3dB$. This indicates that it is crucial to strike the best balance between privacy and performance. A high level of privacy, as achieved with a noise_var of 0.1, may prevent any information leakage, but can result in a significant decrease in performance.

Table 10.7: Effect of SVT mechanism in global model's performance.

	$FedAvg_{global}$	$FedProx_{global+SVT}$[1]	$FedOpt_{global+SVT}$[2]
Low	35.87	34.45	32.50

[1] noise_var=1

[2] noise_var=0.1

The upcoming experiments focus on the impact of the FL processing on the performance of the local models. Table 10.8 displays the performance of the local models and the globally aggregated model on the local validation datasets throughout the FL training, comprising 12 rounds. The table showcases the difference in performance at each round.

Table 10.8: Performance comparison of global and local models on the local datasets. Experiment: High level of heterogeneity, FedOpt strategy, w/o privacy-preserving mechanism.

Client	Model	1	2	3	4	5	6	7	...	11	12
Site-3	*Local*	34.23	34.75	35.03	35.32	35.44	35.50	35.56		35.72	35.74
	Global	34.64	27.90	32.65	35.03	34.97	36.02	36.28		36.68	36.70

Through the analysis of Table 10.8, we can gain a deeper understanding of how the FL process operates. Initially, for at least the first 7 rounds of FL, the local models outperform the global aggregated model on their respective local datasets. This is because the local models prioritize minimizing the loss for their specific datasets, whereas the global model has a more general objective. However, as the FL training process continues, the global model can handle the different objectives of the local models through the aggregation algorithm, eventually achieving better performance than the local models on the local datasets. The benefits of FL training are more pronounced for clients with limited samples and limited performance capabilities, such as Site-3, where the aggregated model outperforms the local one by $0.12dB$.

Qualitative: This sub-section summarizes the key insights and conclusions derived from the experiments on FL VSR. Firstly, the FL global model can achieve performance comparable to the ideal centrally trained model, demonstrating the potential of using distributed training with data available on multiple devices. Secondly, the heterogeneity in data distribution, caused by variations in the number of samples per client, negatively impacts the performance of the global model. However, the use of FL strategies developed to handle heterogeneous data can limit this impact. Thirdly, certain privacy-preserving mechanisms, such as SVT, can lead to a significant reduction in the FL model's performance, highlighting the need to find an optimal balance between performance and privacy. Lastly, FL can be beneficial for clients lacking sufficient data to build high-performance models.

10.3.2 Named Entity Recognition and Classification

Datasets: The NERC (Named Entity Recognition and Classification) uses data from CoNLL datasets, which are widely recognized in the research community. These datasets contain sentences taken from news articles that are manually labeled with different types of entities, such as PER (person), LOC (location), ORG (organization), and MISC (miscellaneous). To introduce heterogeneity among the FL clients, two strategies have been followed: a) language heterogeneity and b)size heterogeneity. In terms of language heterogeneity, examples from English, Spanish, and Dutch have been combined and unevenly distributed across the subsets used by the clients, resulting in some clients having more examples from one language than others. As for size heterogeneity, different numbers of documents have been scattered across the FL clients, causing some clients to have significantly more training examples than others, simulating a possible scenario in which some of the participating clients have little training data in comparison to the others. The following Tables 10.9,10.10 illustrates the dataset distributions achieved by implementing the aforementioned heterogeneity approaches. The FL clients are denoted as Site-1, Site-2, and so on. Additionally, a *Dummy* node is included as a client with no training data, solely intended for evaluation purposes and containing data that does not belong to any client. It is presented how for language heterogeneity strategy the distribution of training instances per language varies from client to client, while for size heterogeneity strategy it is the distribution of the total amount of training instances (while keeping a balance language-wise) which varies among clients.

Table 10.9: NERC datasets distribution for the language unbalance heterogeneity strategy.

Sets	Language	Site-1	Site-2	Site-3	Site-4	Site-5	*Dummy*
train	es	4995	3442	0	1396	0	N/A
	en	0	2294	8424	2793	0	N/A
	nl	0	1147	0	4415	9483	N/A
	total	4995	6884	8424	8381	9483	N/A
validation	es	455	583	0	395	0	1355
	en	0	291	1381	395	0	1355
	nl	0	291	0	791	1558	1355
	total	455	1167	1381	1583	1558	4065

Table 10.10: NERC datasets distribution for the size unbalance heterogeneity strategy.

Sets	Language	Site-1	Site-2	Site-3	Site-4	Site-5	*Dummy*
train	es	5089	2544	2544	1272	1272	N/A
	en	5089	2544	2544	1272	1272	N/A
	nl	5089	2544	2544	1272	1272	N/A
	total	15269	7633	7633	3816	3816	N/A
validation	es	507	507	507	507	507	846
	en	507	507	507	507	507	846
	nl	507	507	507	507	507	846
	total	1523	1523	1523	1523	1523	2540

Quantitative: In the context of NERC, an experiment was conducted involving five clients and one validation *Dummy* node. The validation node contained more balanced data, while the five clients had different levels of data imbalance to simulate heterogeneity. The goal of the experiment was to evaluate NERC performance in the presence of data heterogeneity. The evaluation measures the F-score over the detected named entities, which is the harmonic mean of the precision (the fraction of relevant entities among the retrieved entities) and recall (the fraction of relevant instances that were retrieved).

Table 10.11: NERC FL experiments with the language unbalance heterogeneity strategy. The performance is measured using the F-score.

	FL round (only for FL)	1	5	10	15	20	25
	Training steps	600	3000	6000	9000	12000	15000
Test data	Evaluated model						
Dummy data	Local Training (baseline)	-	0.857	0.886	0.883	0.889	0.895
Site-5 data	Local Training (baseline)	-	0.863	0.881	0.886	0.898	0.895
Dummy data	Aggr. FL model	0.757	0.845	0.856	0.857	0.862	0.856
Site-5 data	Aggr. FL model	0.701	0.783	0.804	0.814	0.824	0.809

Table 10.12: NERC FL experiments with the size unbalance heterogeneity strategy. The performance is measured using the F-score.

	FL round (only for FL)	1	5	10	15	20	25
	Training steps	600	3000	6000	9000	12000	15000
Test data	Evaluated model						
Dummy data	Local Training (baseline)	-	0.854	0.885	0.881	0.887	0.892
Site-5 data	Local Training (baseline)	-	0.874	0.895	0.898	0.908	0.910
Dummy data	Aggr. FL model	0.751	0.833	0.842	0.841	0.840	0.844
Site-5 data	Aggr. FL model	0.766	0.812	0.834	0.842	0.855	0.841

The experimental results were positive, with high scores achieved by the different nodes despite the presence of data heterogeneity. In addition, privacy-preserving mechanisms were tested using SVT, and the obtained results for various SVT parameterizations are presented in Table 10.13 below. The experiments focused on heterogeneity based on differences in data size, and some cells in the table are empty due to errors encountered during certain parameter configurations.

Table 10.13: NERC FL experiments with SVT privacy (language unbalance data).

		FL round (only for FL)	1	5	10	15	20	25
		Training steps	600	3000	6000	9000	12000	15000
	Test data	Evaluated model						
$SVT_{fraction:0.6,epsilon:0.001,noise\_var:1}$	*Dummy* data	Aggr. FL model	0.021	0.021	0.021	-	-	-
	Site-5 data	Aggr. FL model	0.722	0.717	0.723	-	-	-
$SVT_{fraction:0.6,epsilon:0.001,noise\_var:0.5}$	*Dummy* data	Aggr. FL model	0.029	0.029	0.029	0.029	0.029	0.029
	Site-5 data	Aggr. FL model	0.706	0.740	0.710	0.728	0.686	0.679
$SVT_{fraction:0.6,epsilon:0.001,noise\_var:0.1}$	*Dummy* data	Aggr. FL model	0.019	0.019	0.019	0.019	0.019	0.019
	Site-5 data	Aggr. FL model	0.730	0.742	0.728	0.726	0.726	0.718
$SVT_{fraction:0.6,epsilon:0.1,noise\_var:1.0}$	*Dummy* data	Aggr. FL model	0.016	-	-	-	-	-
	Site-5 data	Aggr. FL model	0.723	-	-	-	-	-

Looking at Table 10.14, it appears that the federated model becomes excessively noisy and ineffective for the *Dummy* node, although the reason for this outcome is unclear. Additionally, a data-poisoning experiment was conducted in which one of the nodes was provided with 'poisoned' data. Specifically, the training data for this node was manipulated to switch all the labels for individuals (PER) and locations (LOC). The validation data for this node remained unaltered. This experiment was conducted using the size-unbalance heterogeneity strategy and the performance was measured using the F-score.

Table 10.14: Data-poisoning experiment, with one of the clients having 'poisoning' training data.

	FL round (only for FL)	1	5	10	15	20	25
	Training steps	600	3000	6000	9000	12000	15000
Dummy data	Aggr. FL model	0.756	0.817	0.788	0.831	0.837	0.811
Site-1 data (poisoned)	Aggr. FL model [1]	0.288	0.408	0.408	0.396	0.414	0.411
Site-1 data (poisoned)	Aggr. FL model [2]	0.76	0.844	0.796	0.838	0.849	0.831
Site-2 data	Aggr. FL model [1]	0.783	0.843	0.847	0.848	0.852	0.852
Site-5 data	Aggr. FL model [1]	0.721	0.833	0.827	0.822	0.83	0.843

[1] plus 1 round of local training

[2] without further local training

Qualitative: Based on the experimental results of the NERC tool using FL, several conclusions can be drawn. Firstly, data heterogeneity does not pose a significant problem, indicating that FL is viable in these situations. Both scenarios of heterogeneity devised are realistic and demonstrate the potential of FL to combine different data sources. Secondly, the privacy-preserving mechanism used in the experimentation (SVT) does not seem to be suitable for the NERC model, leading to a drop in performance to that of a randomly initialized model. Thirdly, the Federated training approach is robust against data poisoning, as it enables learning across datasets from different stakeholders and results in a more coherent and robust model. Overall, the findings suggest that FL can be a valuable approach for NERC models, allowing for joint learning across disparate data sources and robustness against data poisoning, while careful consideration should be given to the choice of privacy-preserving mechanism.

10.3.3 Audio Speech Recognition

Datasets: The ASR data consider a different dataset from the literature for each client. The considered corpora include the TEDLIUM [239] (node-1), debating technologies [240] (node-2), Librispeech-other (node-3), Librispeech-clean (node-4) [241], and the Spoken Wikipedia Corpus[2] (node-5). The data configuration is designed to evaluate the impact of heterogeneity levels on data distribution, which is more realistic for the application. To ensure equal data amounts in each node, the number of samples from each corpus is adjusted, and each node is given a subset of 1400 speech recordings for local training. For experiments with low heterogeneity levels, some samples from TEDLIUMv2 and the Spoken Wikipedia corpus are replaced by samples from debating technologies and librispeech-other datasets, respectively. The AI used for the pipeline is based on a Wav2Vec2.0 model [242], which is a self-supervised end-to-end architecture based on convolutional and Transformer layers. The training hyperparameters were the same for the five nodes, and included a batch size of 2, a learning rate of 5×10^{-5} warmed up in the first 10% of the training time, and a gradient accumulation of 16 steps. The local training is performed for 5 epochs. The central server is configured to run for 10 rounds of federated training.

Quantitative: The experiments conducted on ASR include (1) low heterogeneity, (2) high heterogeneity, (3) high heterogeneity with an SVT privacy-preserving strategy, and (4) high heterogeneity with a percentile privacy (PP) preserving strategy. The low heterogeneity experiment aims to evaluate the capabilities of the system to recognize speech under relatively controlled acoustic conditions, and with noise-controlled levels. On the contrary, the high heterogeneity experiment introduces datasets with high levels of noise and in non-controlled acoustic environments. For the SVT and PP we evaluate several hyperparameters (see Table 10.16) to test the impact of the privacy-preserving approaches. The main results of these experiments are presented in Table 10.15, where the performance is evaluated based on the Word Error Rate (WER) after 10 rounds of federated training in each node.

[2]https://nats.gitlab.io/swc/

Table 10.15: Summary of FL results for the different configurations of data heterogeneity and privacy-preserving approaches.

#	Description	WER					
		Site-1	Site-2	Site-3	Site-4	Site-5	Dummy
1	Low heterogeneity	17.7	15.3	8.1	2.7	25.8	3.9
2	High heterogeneity	13.5	12.4	7.9	2.8	25.7	3.8
3	SVT[1]	19.7	15.3	7.8	2.9	24.0	3.6
4	PP[2]	19.7	15.3	7.8	3.0	24.0	3.6

[1] fraction=0.6, eps=[0.001, 0.001, 0.001, 0.1], noise_var=[1, 0.5, 0.1, 1]

[2] perc=[10,40,70], gamma=0.01

Table 10.16 provide further details on the experimental results for each of the four ASR experiments. The tables present information on the behaviour of federated training over 10 rounds, as well as a comparison between the Word Error Rate (WER) before and after aggregation on local test sets.

Table 10.16: Federated Training in different heterogeneity and privacy settings for *Dummy* node and Site-1. Rows 1 and 2 correspond to low and high heterogeneity, respectively, and rows 3 and 4 correspond to the high heterogeneity setting for SVT and PP settings, respectively.

#	Node	Aggr.	1	2	3	4	5	6	7	8	9	10
1	Site-1	Before	19.7	18.3	17.9	18.6	19.0	19.0	19.5	19.5	19.4	19.5
		After	19.9	18.2	18.2	17.9	17.9	17.8	18.2	17.9	17.8	17.8
	Dummy	After	3.6	3.7	3.7	3.8	3.8	3.9	4.0	4.0	3.9	3.9
2	Site-1	Before	13.6	13.6	13.5	13.5	13.4	13.4	13.4	13.3	13.3	13.2
		After	13.5	13.6	13.5	13.5	13.5	13.5	13.3	13.2	13.2	13.2
	Dummy	After	3.9	3.8	3.8	3.7	3.7	3.7	3.6	3.6	3.6	3.6
3	Site-1	Before	19.8	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7
		After	19.6	19.7	19.6	19.7	19.6	19.7	19.6	19.7	19.6	19.7
	Dummy	After	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6
4	Site-1	Before	19.8	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7
		After	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7	19.7
	Dummy	After	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6

Qualitative: The experiments for Speech Recognition have provided several insights. Firstly, data heterogeneity did not have a significant impact on the performance of the speech recognizer, as observed in the *Dummy* node's results. This suggests that FL can be used to obtain a joint and potentially richer model by combining sources of data that cannot be otherwise combined. Secondly, the use of SVT and PP privacy-preserving approaches did not facilitate any improvement of the model during the federated setting. After the first round, the results for different nodes did not change (see Table 10.16, presenting only Site-1 states for simplicity), indicating that these approaches do not help in learning anything useful for the local model. Note that in the same, highly heterogeneous setting, without privacy preserving schemes the WER were changing throughout the FL process, finally reaching lower values.

10.3.4 Insights and Contributions

Summarizing the findings, it could be highlighted that the system adaptation pipeline is capable of enabling the integration of different (in terms of modality, domain and task) AI-based tools into the FL system, and the FL training under realistic conditions. However, the results showed that the performance of the FL model may be affected by certain challenges, such as data heterogeneity and the use of specific privacy-preserving mechanisms. This is highlighted in the low heterogeneity experiment of Table 10.16, where the WER decreases sequentially in Site-1 after 10 aggregation rounds, unlike the other experiments. This can be explained by the IID conditions in Experiment 1 (i.e. the data exhibits IID characteristics), where the central model benefits from the other nodes' contributions, leading to a reduction in WER specifically in Site-1. However, in experiments with higher heterogeneity, the lack of dataset representativeness prevents the central model from achieving similar improvements in Site-1's WER. Although there have been numerous studies on FL recently, there is a lack of research exploring the effectiveness of presented aggregated methods on different types of data sources such as images, audio, and text when used with DL models. Our experiments demonstrate that these existing studies offer minimal or no advantage over the traditional DL approaches. To mitigate these challenges, appropriate FL strategies need to be selected, such as aggregation algorithms that handle heterogeneous data and privacy mechanisms that strike a balance between performance and privacy. Moreover, no substantial problems were identified for the integration of the tools into the FL framework, except in some specific cases where the type of ML model hindered any development.

In terms of privacy, the integrated security mechanisms seem to provide a satisfactory level of protection against privacy attacks, although it is important to find the best trade-off between performance and privacy. In particular, even though in most cases the chosen security strategies had no particular impact on the training of the models, specific privacy-preserving mechanisms (*e.g.* SVT) could dramatically decrease the FL model's performance. In terms of fairness, the chosen aggregation mechanisms seem to mitigate the potential biases of the ML model, ensuring that the FL system does not yield systematic advantages to certain privileged nodes. For example, the results in Table 10.4 for the occluded datasets (D^1_{occl}, D^2_{occl}, D^3_{occl} and D^4_{occl}), the centrally trained model's accuracy drops compared to the global datasets. However, the FL aggregated models, particularly in the case of high-heterogeneity, still exhibit respectable performance, indicating their ability to handle occluded data under realistic conditions.

Regarding robustness, the results showed that the FL system appears to be resistant to data poisoning attacks. More specifically, FL appears to be resistant to label-based data poisoning attacks (intentional or not) and, according to the experiments, even nodes with poisoned data can benefit from the resulting federated model. In particular, results in Table 10.14 suggest that despite the poisoned data in Site-1 (the client has intentionally injected poisoned training data), the aggregated FL model manages to achieve comparable performance to the models trained on clean data from other sites (Site-2 and Site-5). This indicates the resilience of the FL system against label-based data poisoning attacks. However, it is important to test the system's robustness under different conditions, including naturally occurring conditions and those set up by malicious actors.

Finally, as a final observation, we could say that the global FL model can achieve similar performance to the ideal model trained centrally, taking into account the characteristics of each node, and each type of data and selecting the appropriate training tools (*i.e.* aggregation algorithms and privacy mechanisms). All of the above indicates that the FL platform has the potential to progress from proof of concept to a trustworthy application, as long as it is tested and assessed against appropriate performance indicators in a precise context.

Chapter 11

Conclusions and Future Work

11.1 Summary of Key Findings

Through a series of comprehensive studies on FL, this dissertation presented significant advancements and insights into the field, exploring various dimensions from methodologies to real-world applications and inherent challenges. At the core, FL emerges as a powerful paradigm for distributed learning, offering significant advantages in terms of training time efficiency, inference speed, privacy preservation, collaborative learning, and handling large distributed datasets. These strengths align FL closely with the current and future needs of handling data in a privacy-aware, efficient, and collaborative manner across various domains.

Efficiency and speed are critical findings, with FL demonstrating the ability to accelerate training time through parallel computations. This feature was especially notable in environments with homogeneous data distribution, where FL's parallel computation capabilities facilitated faster algorithm convergence. The inference speed, represented by quicker local model inference times in NER experiments, highlights FL's potential for real-time applications. In addition, privacy preservation and collaborative learning emerged as standout benefits of FL. The use of mechanisms like Secure Aggregation underscored FL's capability to maintain confidentiality, showcasing a significant advantage over centralized training methods in terms of privacy. Furthermore, FL's decentralized nature was found to streamline collaborative learning efforts, distributing tasks across federation nodes to enhance overall model performance.

However, the studies also shed light on challenges, particularly regarding FL's performance across data heterogeneity. While FL models maintained competitive accuracy against centralized approaches, simple algorithms like FedAvg struggled with highly heterogeneous data, suggesting a need for more sophisticated algorithms in such scenarios. The balance between model accuracy and device performance also posed a dilemma, highlighting the need for optimization to achieve efficient model performance without overloading device capacities.

In exploring data modalities and fusion in the context of 3D action recognition, the variability in FL strategies' effectiveness became clearly visible, especially in handling non-IID data distributions. The research underscored the complexity of communication overheads and data synchronization, while also highlighting the importance of multi-modal data fusion in enhancing FL performance, though with the challenges of integrating diverse datasets effectively.

Noteworthy advancements were made through the introduction of novel FL techniques, such as federated distillation for knowledge transfer and a hybrid approach that combines self-supervised and supervised learning. These methodologies aimed to address specific FL challenges, including catastrophic forgetting in incremental learning and the efficient handling of sparse and imbalanced data across nodes. Knowledge transfer within FL is significantly enhanced by advanced representation learning techniques, enabling the sharing of insights across models and nodes without the direct transfer of data. The efficacy of representation learning was found to be closely tied to the strategic application of losses at different model layers, emphasizing that tailored approaches to learning and model optimization can significantly enhance performance. The superior performance of FL strategies employing shared datasets and federated distillation pointed to the importance of leveraging collective knowledge, highlighting the value of collaborative and carefully tailored learning strategies in FL.

Insights from incremental learning settings underscored the importance of balance in contrastive learning and task memory management, illustrating FL's adaptability and the critical role of strategically devised learning strategies. Managing non-IID data through innovative schemes like FedRCIL showcased FL's robustness and adaptability in diverse and challenging data environments, emphasizing the necessity of innovative and flexible approaches to learning that can effectively handle the complexities of distributed data.

Architectural flexibility and robustness were also highlighted, with studies examining the variability of model architectures within FL frameworks and proposing solutions for efficient model aggregation and communication. These strategies ensure that despite the heterogeneity of models deployed across different nodes, the system can still harmonize learning and knowledge extraction. This exploration underscored FL's adaptability and robustness, emphasizing the significance of representation learning in improving learning outcomes across decentralized networks.

The interconnection of these studies illuminates the pivotal role of sophisticated representation learning as a fundamental catalyst in FL, particularly in navigating through the complexities of incremental learning, knowledge transfer, communication efficiency, and model architectural variability. Representation learning serves as the backbone of these FL advancements, enabling systems to distill and leverage complex data patterns across decentralized environments effectively. In conclusion, while FL shows promising potential in privacy preservation, efficiency, and handling distributed datasets, it faces significant challenges in data heterogeneity and balancing computational efficiency with accuracy. The insights from these studies underscore the importance of continued innovation in FL methodologies and algorithm development to overcome these challenges.

11.2 Contributions of the Dissertation

This dissertation has provided a comprehensive literature review in the field of FL, where challenges have been identified and analyzed. Building upon this foundation, a series of studies were conducted, each targeting distinct yet interrelated challenges within FL. From optimizing communication efficiency and model heterogeneity to enhancing representation learning, knowledge retention, and handling model architecture variability, these studies collectively contribute to a deeper understanding and advancement of FL. They not only address specific issues but also add layers of complexity and capability, broadening FL's applicability and effectiveness in various scenarios.

The first study addressed two fundamental challenges in FL: communication efficiency and model heterogeneity. It recognized that the traditional FL paradigm, while promising, faced significant hurdles in handling diverse model architectures across different nodes and ensuring efficient communication. The study's significant contribution was the development of novel algorithms that optimized data transmission between the central server and the nodes, thereby reducing bandwidth requirements and communication overheads. It also proposed adaptive learning algorithms tailored for heterogeneous models, enhancing the FL system's overall effectiveness. This foundational work set the stage for more advanced explorations in the FL domain, emphasizing the importance of efficient communication protocols and versatile model architectures.

Building on the groundwork laid by the first study, the second research focused on optimizing data partitioning and client selection in FL. This study addressed the challenge of ensuring fair and representative participation of nodes in the learning process, a crucial aspect for the scalability and effectiveness of FL systems. It introduced innovative strategies for data distribution and devised more intelligent criteria for selecting participant nodes. These advancements were pivotal in handling the data imbalance and ensuring that the learning process was more inclusive and representative of the diverse data sources in FL. The contributions from this study significantly improved the FL model's training process, making it more balanced and efficient.

The third study brought a novel perspective to FL by focusing on representation learning and federated distillation. It introduced an innovative federated distillation weight aggregation method, tailored for effective learning in distributed environments. The key contribution was an algorithm that facilitated knowledge distillation across nodes, effectively reducing communication costs and enhancing the efficiency of the FL system. This study marked a significant advancement in understanding knowledge transfer within FL networks, allowing for more efficient model training processes, especially in bandwidth-limited scenarios. It highlighted the importance of effective representation learning and knowledge distillation in improving FL systems' overall performance.

Addressing the challenge of catastrophic forgetting, the fourth study in the series proposed an innovative federated incremental learning algorithm. Integrating representation learning with knowledge distillation and contrastive learning, this study focused on retaining knowledge effectively in FL systems. It demonstrated how FL models could adapt to new conditions and tasks without forgetting previously acquired knowledge, a critical step towards creating dynamic and robust FL systems. This study's contributions were pivotal in enhancing the FL models' ability to continually learn and evolve, marking a significant progression in the field.

The fifth study tackled the challenge of training models with limited and scattered data, a common scenario in FL. It proposed a hybrid approach that combined self-supervised and supervised learning techniques, focusing on the efficient use of unlabeled data. This study's significant contribution was the development of a flexible learning approach that could adapt to various data availability scenarios in FL settings. It expanded the FL domain's capabilities, demonstrating how systems could operate efficiently with sparse data and without prior knowledge of data distribution across nodes.

The final study presents a novel approach to addressing model heterogeneity in FL, marking the first attempt to tackle this challenge within the field. By exploring strategies to manage diverse model architectures effectively, the study contributes significantly to enhancing FL's adaptability and efficiency in handling distributed learning scenarios across varied computational landscapes.

11.2.1 Overall Contributions and Impact to the Field

Collectively, these studies represent a comprehensive journey through the evolving landscape of FL. From addressing foundational challenges like communication efficiency and model heterogeneity to tackling advanced issues like knowledge retention and adaptation to sparse data scenarios, these studies collectively push the boundaries of FL. They highlight the progression from understanding basic FL principles to applying these principles in complex, real-world scenarios. Each study builds upon the insights and findings of its predecessors, creating a rich tapestry of research that significantly enhances our understanding and application of FL. The introduction of novel algorithms and methodologies in these studies has significantly expanded the toolkit available for FL, making it more adaptable to a range of scenarios and challenges. By addressing both foundational and advanced challenges in FL, these studies lay a robust groundwork for future research in the field. They open up new avenues for exploration, particularly in applying FL in dynamic, real-world environments. Together, they contribute to transforming FL from a theoretical concept to a practical, robust, and scalable learning paradigm ready for diverse applications. Moreover, by ensuring balanced data participation and fair client selection, these studies contribute to developing more ethical and equitable AI systems, which is crucial in the current landscape where AI's societal impact is increasingly examined.

11.3 Limitations and Future Research Directions

11.3.1 Limitations

The research conducted in this dissertation on FL has certainly pushed the boundaries of the field, yet there are limitations that pave the way for future research. In fact, while advancements have been made in communication efficiency and data partitioning, scaling these solutions to extremely large and heterogeneous networks remains challenging. The efficiency in environments with highly variable network connectivity and computational capacities needs further exploration. Additionally, this work, though comprehensive, might have limitations in its generalizability across vastly different domains, thus the applicability of the proposed solutions in domains with unique characteristics requires more investigation. Furthermore, techniques like federated distillation, representation learning and incremental learning add complexity to FL systems. This complexity could introduce computational and management overheads that might delay practical deployment, especially in resource-constrained environments. Lastly, while FL inherently aims to preserve data privacy, the variatons of data sharing and aggregation in the proposed FL systems raise new privacy and security concerns. As FL systems become more complex, ensuring robust privacy and security without compromising on efficiency is a growing challenge.

11.3.2 Future Research Directions

Future research in FL should focus on creating algorithms that can efficiently scale in extremely large and diverse networks, possibly through novel lightweight models that require fewer resources, alongside innovative data transmission techniques that reduce bandwidth usage and speed up communication. Another key area is the exploration of cross-domain generalizability and transfer learning, essential for adapting FL models across various domains. Simultaneously, advancing data privacy and security is crucial, potentially through innovative cryptographic techniques like homomorphic encryption and decentralized blockchain approaches. Addressing the robustness of FL systems against adversarial attacks and data poisoning is also pivotal, requiring sophisticated methods such as advanced anomaly detection and adversarial training. With the expansion of FL, its energy consumption and environmental impact become significant concerns, calling for research into energy-efficient FL algorithms and sustainable deployment strategies. With the increasing deployment of AI systems, ensuring that FL models are fair and unbiased is critical. Research into developing fairness-aware algorithms and addressing ethical considerations in FL will be increasingly important. Lastly, tackling real-world deployment challenges, including system interoperability and regulatory compliance, is vital for the broader adoption and practical application of FL technologies. By focusing on these diverse yet interconnected areas, future research can holistically advance FL, addressing current limitations and fostering innovation for various applications.

11.4 Final Thoughts and Perspectives on FL's Future

Reflecting on the progress and possibilities of FL, it's evident that this area is at a crucial point, offering significant opportunities for development and use in different sectors. As FL matures, we can anticipate wider industry adoption in sectors such as healthcare, finance, e-commerce, and smart cities, driven by the demand for privacy-preserving, collaborative learning solutions. The integration of FL with edge computing is set to transform IoT and mobile applications by enabling more efficient data processing and decision-making at the network's edge. This will be further accelerated by advancements in 5G networks, which promise to enhance the performance and scalability of FL systems. FL's impact will also expand into cross-disciplinary fields like bioinformatics, environmental science, and social sciences, opening up new collaborative research opportunities. One of the key challenges for FL will be tackling data heterogeneity and skewness, particularly in handling non-IID data and personalizing models to suit individual user needs. Sustainability and ethical considerations, including energy efficiency and model bias, will also gain attention. Ensuring interoperability between different FL systems and standardizing protocols will be vital for its broader adoption, requiring collaborative efforts across academia, industry, and regulatory bodies. Lastly, the evolution towards more human-centric AI solutions in FL will involve integrating user feedback, ethical AI principles, and societal impacts into the learning process, bringing a new era of responsible and inclusive AI. The future of FL is not just about technological advancements but also about creating a more collaborative, privacy-preserving, and user-centric AI ecosystem. As we move forward, FL will likely become a cornerstone technology in the AI landscape, reshaping how we think about data sharing, collaborative learning, and AI-driven solutions.

Bibliography

[1] A. Shahroudy, J. Liu, T. T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[3] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[4] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[5] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[6] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *ArXiv*, 2018.

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[8] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[9] J. Ding, E. Tramel, A. K. Sahu, S. Wu, S. Avestimehr, and T. Zhang, "Federated learning challenges and opportunities: An outlook," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8752–8756, IEEE, 2022.

[10] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[11] K. Das and R. N. Behera, "A survey on machine learning: concept, algorithms and applications," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 2, pp. 1301–1309, 2017.

[12] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, 2018.

[13] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, p. 2470, 2021.

[14] V. Sharma, M. Gupta, A. Kumar, and D. Mishra, "Video processing using deep learning techniques: A systematic literature review," *IEEE Access*, vol. 9, pp. 139489–139507, 2021.

[15] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu, "Human action recognition from various data modalities: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] S. B. Atitallah, M. Driss, W. Boulila, and H. B. Ghézala, "Leveraging deep learning and iot big data analytics to support the smart cities development: Review and future directions," *Computer Science Review*, vol. 38, p. 100303, 2020.

[17] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big data: Issues and challenges moving forward," in *2013 46th Hawaii International Conference on System Sciences*, pp. 995–1004, IEEE, 2013.

[18] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," 2017.

[19] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *Knowledge and Information Systems*, vol. 64, no. 4, pp. 885–917, 2022.

[20] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information Processing and Management*, vol. 59, no. 6, p. 103061, 2022.

[21] M. Ganapathy, *An Introduction to Federated Learning and Its Analysis*. PhD thesis, University of Nevada, Las Vegas, 2021.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[23] Y. Zhuang, G. Li, and J. Feng, “A survey on entity alignment of knowledge base,” *Journal of Computer Research and Development*, vol. 1, pp. 165–192, 2016.

[24] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, “Secureboost: A lossless federated learning framework,” 2019.

[25] Y. Liu, T. Chen, and Q. Yang, “Secure federated transfer learning,” 2018.

[26] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning,” in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 493–506, 2020.

[27] Y. Aono *et al.*, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[28] J. Zhang, H. Zhu, F. Wang, J. Zhao, Q. Xu, and H. Li, “Security and privacy threats to federated learning: Issues, methods, and challenges,” *Security and Communication Networks*, vol. 2022, 2022.

[29] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, “Protection against reconstruction and its applications in private federated learning,” 2018.

[30] L. Melis, C. Song, E. D. De Cristofaro, and V. Shmatikov, “Inference attacks against collaborative learning,” 2018.

[31] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, “Detection of adversarial training examples in poisoning attacks through anomaly detection,” 2018.

[32] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” 2018.

[33] J. Carnerero-Cano, L. Muñoz-González, P. Spencer, and E. C. Lupu, “Regularisation can mitigate poisoning attacks: A novel analysis based on multiobjective bilevel optimisation,” 2020.

[34] Y. Tian, W. Zhang, A. Simpson, Y. Liu, and Z. Jiang, “Defending against data poisoning attacks: From distributed learning to federated learning,” *The Computer Journal*, vol. 66, 12 2021.

[35] Z. Anastasakis, K. Psychogyios, T. Velivassaki, S. Bourou, A. Voulkidis, D. Skias, A. Gonos, and T. Zahariadis, “Enhancing cyber security in iot systems using

fl-based ids with differential privacy," in *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 30–34, IEEE, 2022.

[36] A. Segal, A. Marcedone, B. Kreuter, D. Ramage, H. B. McMahan, K. Seth, K. A. Bonawitz, S. Patel, and V. Ivanov, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017.

[37] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[38] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[39] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, "V eri fl: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1736–1751, 2020.

[40] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

[41] B. Choi, J.-y. Sohn, D.-J. Han, and J. Moon, "Communication-computation efficient secure aggregation for federated learning," *arXiv preprint arXiv:2012.05433*, 2020.

[42] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 479–489, 2021.

[43] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2009.11248*, 2020.

[44] N. Bouacida and P. Mohapatra, "Vulnerabilities in federated learning," *IEEE Access*, vol. 9, pp. 63229–63249, 2021.

[45] O. Shahid, S. Pouriyeh, R. M. Parizi, Q. Z. Sheng, G. Srivastava, and L. Zhao, "Communication efficiency in federated learning: Achievements and challenges," 2021.

[46] P. M. Mammen, "Federated learning: Opportunities and challenges," 2021.

[47] S. P. Ramu, P. Boopalan, Q. V. Pham, P. K. R. Maddikunta, T. Huynh-The, M. Alazab, *et al.*, "Federated learning enabled digital twins for smart cities:

Concepts, recent advances, and future directions," *Sustainable Cities and Society*, vol. 79, p. 103663, 2022.

[48] Q. Yang, "Toward responsible ai: An overview of federated learning for user-centered privacy-preserving computing," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–22, 2021.

[49] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[50] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.

[51] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International Conference on Machine Learning*, pp. 7252–7261, PMLR, 2019.

[52] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016.

[53] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1310–1322, 2019.

[54] P. Jiang and L. Ying, "An optimal stopping approach for iterative training in federated learning," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2020.

[55] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[56] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018.

[57] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[58] V. Smith, C. K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[59] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, *et al.*, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[60] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, “Efficient and privacy-enhanced federated learning for industrial artificial intelligence,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[61] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, “Abnormal client behavior detection in federated learning,” 2019.

[62] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, “Safa: a semi-asynchronous protocol for fast federated learning with low overhead,” *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2020.

[63] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y. C. Liang, and D. I. Kim, “Incentive design for efficient federated learning in mobile networks: A contract theory approach,” in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, pp. 1–5, IEEE, 2019.

[64] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, “Federated learning over wireless networks: Optimization model design and analysis,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1387–1395, IEEE, 2019.

[65] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, *et al.*, “Towards taming the resource and data heterogeneity in federated learning,” in *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, pp. 19–21, 2019.

[66] T. Li, M. Sanjabi, A. Beirami, and V. Smith, “Fair resource allocation in federated learning,” 2019.

[67] M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” in *International Conference on Machine Learning*, pp. 4615–4625, PMLR, 2019.

[68] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, “In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning,” *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[69] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” 2019.

[70] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, “Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records,” *Journal of Biomedical Informatics*, vol. 99, p. 103291, 2019.

[71] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, “Local learning matters: Rethinking data heterogeneity in federated learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8397–8406, 2022.

[72] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data," *Plos One*, vol. 15, no. 4, p. e0230706, 2020.

[73] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2019.

[74] A. Suri, P. Kanani, V. Marathe, and D. Peterson, "Subject membership inference attacks in federated learning," 06 2022.

[75] N. Tomashenko, S. Mdhaffar, M. Tommasi, Y. Estève, and J.-F. Bonastre, "Privacy attacks for automatic speech recognition acoustic models in a federated learning framework," in *ICASSP*, pp. 6972–6976, IEEE, 2022.

[76] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023.

[77] Y. Zhang, D. Zeng, J. Luo, Z. Xu, and I. King, "A survey of trustworthy federated learning with perspectives on security, robustness, and privacy," *arXiv preprint arXiv:2302.10637*, 2023.

[78] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[79] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and S. Y. Philip, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE transactions on neural networks and learning systems*, 2022.

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

[81] T. Li, A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[82] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020.

[83] M. Asad, A. Moustafa, and T. Ito, "Fedopt: Towards communication efficiency and privacy preservation in federated learning," *Applied Sciences*, vol. 10, no. 8, p. 2864, 2020.

[84] P. Jain, S. Goenka, S. Bagchi, B. Banerjee, and S. Chaterji, "Federated action recognition on heterogeneous embedded devices," 2021.

[85] K. Doshi and Y. Yilmaz, "Federated learning-based driver activity recognition for edge devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3338–3346, 2022.

[86] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, "Federated learning for vision-and-language grounding problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 11572–11579, 2020.

[87] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L. P. Morency, "Think locally, act globally: Federated learning with local and global representations," 2020.

[88] P. Guo, P. Wang, J. Zhou, S. Jiang, and V. M. Patel, "Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2423–2432, 2021.

[89] Y. Zhao, P. Barnaghi, and H. Haddadi, "Multimodal federated learning on iot data," 2022.

[90] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5457–5466, 2018.

[91] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European Conference on Computer Vision*, pp. 816–833, Springer, 2016.

[92] M. Liu, H. Liu, and C. Chen, "Robust 3d action recognition through sampling local appearances and global distributions," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 1932–1947, 2018.

[93] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high-performance human action recognition from skeleton data," 2017.

[94] T. S. Kim and A. Reiter, "Interpretable 3d human action analysis with temporal convolutional networks," 2017.

[95] H. Liu, J. Tu, and M. Liu, "Two-stream 3d convolutional neural network for skeleton-based action recognition," 2017.

[96] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Adaptive spectral graph convolutional networks for skeleton-based action recognition," 2018.

[97] P. Wang, W. Li, Z. Gao, C. Tang, and P. O. Ogunbona, "Depth pooling based large-scale 3-d action recognition with convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1051–1061, 2018.

[98] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu, "Online human action detection using joint classification-regression recurrent neural networks," in *European Conference on Computer Vision*, pp. 203–220, Springer, 2016.

[99] H. Rahmani and A. Mian, "3d action recognition from novel viewpoints," in *CVPR*, June 2016.

[100] Z. Tu, W. Xie, Q. Qin, R. Poppe, R. C. Veltkamp, B. Li, and J. Yuan, "Multi-stream cnn: Learning representations based on human-related regions for action recognition," *Pattern Recognition*, vol. 79, pp. 32–43, 2018.

[101] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense rgb-d scene flow," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 98–104, IEEE, 2015.

[102] Z. Luo, B. Peng, D. A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *CVPR*, 2017.

[103] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona, "Scene flow to action map: A new representation for rgb-d based action recognition with convolutional neural networks," in *CVPR*, 2017.

[104] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," 2015.

[105] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," 2014.

[106] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," 2016.

[107] A. Shahroudy, T. T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in rgb+ d videos," 2016.

[108] Z. Luo, J. T. Hsieh, L. Jiang, J. C. Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged information," in *European Conference on Computer Vision (ECCV)*, 2018.

[109] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[110] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3d: generic feature for video analysis," 2014.

[111] B. van Berlo, A. Saeed, and T. Ozcelebi, "Towards federated unsupervised representation learning," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pp. 31–36, 2020.

[112] Y. Zhao, H. Liu, H. Li, P. Barnaghi, and H. Haddadi, "Semi-supervised federated learning for activity recognition," 2020.

[113] J. Zhang, Y. Yu, S. Tang, J. Wu, and W. Li, "Variational autoencoder with cca for audio-visual cross-modal retrieval," 2021.

[114] W. Wang, R. Arora, K. Livescu, and J. Bilmes, "On deep multi-view representation learning," in *International Conference on Machine Learning*, pp. 1083–1092, PMLR, 2015.

[115] J. Liu, A. Shahroudy, M. Perez, G. Wang, L. Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, 2019.

[116] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[117] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[118] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 191–205, 2021.

[119] S. Cheng, J. Wu, Y. Xiao, and Y. Liu, "Fedgems: Federated learning of larger server models via selective knowledge fusion," *arXiv preprint arXiv:2110.11027*, 2021.

[120] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje, "Ensemble attention distillation for privacy-preserving federated learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15076–15086, 2021.

[121] A. Parvaneh, E. Abbasnejad, D. Teney, G. R. Haffari, A. Van Den Hengel, and J. Q. Shi, "Active learning by feature mixing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12237–12246, 2022.

[122] A. Psaltis, C. Chatzikonstantinou, C. Z. Patrikakis, and P. Daras, "Fedrcil: Federated knowledge distillation for representation based contrastive incremental

learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3463–3472, 2023.

[123] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

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

[125] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[126] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[127] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[128] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2023.

[129] H. Cha, J. Lee, and J. Shin, "Co2l: Contrastive continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9516–9525, October 2021.

[130] G. Yang, E. Fini, D. Xu, P. Rota, M. Ding, M. Nabi, X. Alameda-Pineda, and E. Ricci, "Uncertainty-aware contrastive distillation for incremental semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 2567–2581, 2022.

[131] D. T. Chang, "Exemplar-based contrastive self-supervised learning with few-shot class incremental learning," *arXiv preprint arXiv:2202.02601*, 2022.

[132] J. Dong, L. Wang, Z. Fang, G. Sun, S. Xu, X. Wang, and Q. Zhu, "Federated class-incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10164–10173, 2022.

[133] J. Dong, Y. Cong, G. Sun, Y. Zhang, B. Schiele, and D. Dai, "No one left behind: Real-world federated class-incremental learning," *arXiv preprint arXiv:2302.00903*, 2023.

[134] J. Dong, D. Zhang, Y. Cong, W. Cong, H. Ding, and D. Dai, "Federated incremental semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3934–3943, June 2023.

[135] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

[136] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[137] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

[138] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, pp. 462–466, 1952.

[139] Y.-n. Han and J.-w. Liu, "Online continual learning via the meta-learning update with multi-scale knowledge distillation and data augmentation," *Engineering Applications of Artificial Intelligence*, vol. 113, p. 104966, 2022.

[140] L. Hu, H. Yan, L. Li, Z. Pan, X. Liu, and Z. Zhang, "Mhat: An efficient model-heterogenous aggregation training scheme for federated learning," *Information Sciences*, vol. 560, pp. 493–503, 2021.

[141] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[142] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11953–11962, 2022.

[143] T. Huang, S. You, F. Wang, C. Qian, and C. Xu, "Knowledge distillation from a stronger teacher," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 33716–33727, Curran Associates, Inc., 2022.

[144] E. Belouadah, A. Popescu, and I. Kanellos, "A comprehensive study of class incremental learning algorithms for visual tasks," *Neural Networks*, vol. 135, pp. 38–54, 2021.

[145] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-incremental learning: survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2022.

[146] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[147] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[148] Y. Luo, L. Yin, W. Bai, and K. Mao, "An appraisal of incremental learning methods," *Entropy*, vol. 22, no. 11, p. 1190, 2020.

[149] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 583–592, 2019.

[150] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[151] S. Yan, J. Xie, and X. He, "Der: Dynamically expandable representation for class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3014–3023, June 2021.

[152] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3589–3599, 2021.

[153] W. Sun, J. Zhang, D. Wang, Y.-a. Geng, and Q. Li, "Ilcoc: An incremental learning framework based on contrastive one-class classifiers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3580–3588, June 2021.

[154] J.-y. Han and J.-w. Liu, "Instance-level and class-level contrastive incremental learning for image classification," in *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2022.

[155] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and*

Statistics (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR, 20–22 Apr 2017.

[156] Z. Wu, S. Sun, Y. Wang, M. Liu, Q. Pan, X. Jiang, and B. Gao, “Fedict: Federated multi-task distillation for multi-access edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[157] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, “Communication-efficient federated learning via knowledge distillation,” *Nature communications*, vol. 13, no. 1, p. 2032, 2022.

[158] L. Liu, J. Zhang, S. Song, and K. B. Letaief, “Communication-efficient federated distillation with active data sampling,” in *ICC 2022-IEEE International Conference on Communications*, pp. 201–206, IEEE, 2022.

[159] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, “Self-supervised representation learning: Introduction, advances, and challenges,” *IEEE Signal Processing Magazine*, vol. 39, no. 3, pp. 42–62, 2022.

[160] Y. Chen, M. Mancini, X. Zhu, and Z. Akata, “Semi-supervised and unsupervised deep visual learning: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[161] X. Wang, D. Kihara, J. Luo, and G.-J. Qi, “Enaet: A self-trained framework for semi-supervised and supervised learning with ensemble transformations,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1639–1647, 2020.

[162] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, “Virtual adversarial training: a regularization method for supervised and semi-supervised learning,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[163] V. Verma, K. Kawaguchi, A. Lamb, J. Kannala, A. Solin, Y. Bengio, and D. Lopez-Paz, “Interpolation consistency training for semi-supervised learning,” *Neural Networks*, vol. 145, pp. 90–106, 2022.

[164] L. Zhang and G.-J. Qi, “Wcp: Worst-case perturbations for semi-supervised deep learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3912–3921, 2020.

[165] Y. Chen, X. Zhu, and S. Gong, “Semi-supervised deep learning with memory,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 268–283, 2018.

[166] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, “Deep co-training for semi-supervised image recognition,” in *Proceedings of the european conference on computer vision (eccv)*, pp. 135–152, 2018.

[167] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.

[168] G.-J. Qi, L. Zhang, H. Hu, M. Edraki, J. Wang, and X.-S. Hua, "Global versus localized generative adversarial nets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1517–1525, 2018.

[169] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1476–1485, 2019.

[170] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22243–22255, 2020.

[171] F. Zhang, K. Kuang, Z. You, T. Shen, J. Xiao, Y. Zhang, C. Wu, Y. Zhuang, and X. Li, "Federated unsupervised representation learning," *arXiv preprint arXiv:2010.08982*, 2020.

[172] Y. Wu, D. Zeng, Z. Wang, Y. Sheng, L. Yang, A. J. James, Y. Shi, and J. Hu, "Federated self-supervised contrastive learning and masked autoencoder for dermatological disease diagnosis," *arXiv preprint arXiv:2208.11278*, 2022.

[173] Z. Long, J. Wang, Y. Wang, H. Xiao, and F. Ma, "Fedcon: A contrastive framework for federated semi-supervised learning," *arXiv preprint arXiv:2109.04533*, 2021.

[174] S. A. Khowaja, K. Dev, S. M. Anwar, and M. G. Linguraru, "Selffed: Self-supervised federated learning for data heterogeneity and label scarcity in iomt," *arXiv preprint arXiv:2307.01514*, 2023.

[175] S. Han, S. Park, F. Wu, S. Kim, C. Wu, X. Xie, and M. Cha, "Fedx: Unsupervised federated learning with cross knowledge distillation," in *European Conference on Computer Vision*, pp. 691–707, Springer, 2022.

[176] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[177] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[178] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby, "Self-supervised gans via auxiliary rotation loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12154–12163, 2019.

[179] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[180] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[181] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

[182] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12299–12310, 2021.

[183] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

[184] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18661–18673, 2020.

[185] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 13172–13179, 2020.

[186] C. He, A. D. Shah, Z. Tang, D. F. N. Sivashunmugam, K. Bhogaraju, M. Shimpi, L. Shen, X. Chu, M. Soltanolkotabi, and S. Avestimehr, "Fedcv: a federated learning framework for diverse computer vision tasks," *arXiv preprint arXiv:2111.11066*, 2021.

[187] A. Psaltis, C. Z. Patrikakis, and P. Daras, "Deep multi-modal representation schemes for federated 3d human action recognition," in *Computer Vision – ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*, (Berlin, Heidelberg), p. 334–352, Springer-Verlag, 2023.

[188] Z. Mo, Z. Gao, C. Zhao, and Y. Lin, "Feddq: A communication-efficient federated learning approach for internet of vehicles," *Journal of Systems Architecture*, vol. 131, p. 102690, 2022.

[189] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," *arXiv preprint arXiv:2006.12097*, 2020.

[190] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[191] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[192] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[193] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[194] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[195] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 6105–6114, 2019.

[196] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[197] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

[198] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[199] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12116–12128, 2021.

[200] A. Hammam, F. Bonarens, S. E. Ghobadi, and C. Stiller, "Identifying out-of-domain objects with dirichlet deep neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4560–4569, 2023.

[201] L. Stäcker, J. Fei, P. Heidenreich, F. Bonarens, J. Rambach, D. Stricker, and C. Stiller, "Deployment of deep neural networks for object detection on edge ai devices with runtime optimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 1015–1022, October 2021.

[202] S. Lee, H. Seong, S. Lee, and E. Kim, "Correlation verification for image retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5374–5384, June 2022.

[203] S. Gkelios, A. Kastellos, Y. Boutalis, and S. A. Chatzichristofis, "Universal image embedding: Retaining and expanding knowledge with multi-domain fine-tuning," *IEEE Access*, 2023.

[204] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proceedings of the ACM Web Conference 2022*, pp. 622–632, 2022.

[205] C.-H. Yao, B. Gong, H. Qi, Y. Cui, Y. Zhu, and M.-H. Yang, "Federated multi-target domain adaptation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1424–1433, 2022.

[206] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

[207] I. R. Dave, C. Chen, and M. Shah, "Spact: Self-supervised privacy preservation for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20164–20173, 2022.

[208] K. Doshi and Y. Yilmaz, "Federated learning-based driver activity recognition for edge devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3338–3346, June 2022.

[209] Q. Meng, F. Zhou, H. Ren, T. Feng, G. Liu, and Y. Lin, "Improving federated learning face recognition via privacy-agnostic clusters," *arXiv preprint arXiv:2201.12467*, 2022.

[210] D. Chowdhury, S. Banerjee, M. Sannigrahi, A. Chakraborty, A. Das, A. Dey, and A. D. Dwivedi, "Federated learning based covid-19 detection," *Expert Systems*, vol. 40, no. 5, p. e13173, 2023.

[211] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International conference on machine learning*, pp. 2089–2099, PMLR, 2021.

[212] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[213] A. Psaltis, A. Kastellos, C. Z. Patrikakis, and P. Daras, "Fedlid: Self-supervised federated learning for leveraging limited image data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1039–1048, 2023.

[214] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10713–10722, June 2021.

[215] F. Zhang, K. Kuang, L. Chen, Z. You, T. Shen, J. Xiao, Y. Zhang, C. Wu, F. Wu, Y. Zhuang, *et al.*, "Federated unsupervised representation learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, no. 8, pp. 1181–1193, 2023.

[216] Y. Shen, Y. Zhou, and L. Yu, "Cd2-pfed: Cyclic distillation-guided channel decoupling for model personalization in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10041–10050, 2022.

[217] N. Tastan and K. Nandakumar, "Capride learning: Confidential and private decentralized learning based on encryption-friendly distillation loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8084–8092, 2023.

[218] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[219] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10174–10183, June 2022.

[220] J. Jang, H. Ha, D. Jung, and S. Yoon, "Fedclassavg: Local representation learning for personalized federated learning on heterogeneous neural networks," in *Proceedings of the 51st International Conference on Parallel Processing*, pp. 1–10, 2022.

[221] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning across heterogeneous clients," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8432–8440, 2022.

[222] Y. Tan, G. Long, J. Ma, L. Liu, T. Zhou, and J. Jiang, "Federated learning from pre-trained models: A contrastive learning approach," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19332–19344, 2022.

[223] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10143–10153, 2022.

[224] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.

[225] H. Xia, K. Li, and Z. Ding, "Personalized semantics excitation for federated image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19301–19310, 2023.

[226] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[227] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[228] Y. Zhang, Y. Xu, S. Wei, Y. Wang, Y. Li, and X. Shang, "Doubly contrastive representation learning for federated image recognition," *Pattern Recognition*, vol. 139, p. 109507, 2023.

[229] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

[230] P. Singh, M. K. Singh, R. Singh, and N. Singh, "Federated learning: Challenges, methods, and future directions," in *Federated Learning for IoT Applications*, pp. 199–214, Springer, 2022.

[231] T. R. Gadekallu, Q.-V. Pham, T. Huynh-The, S. Bhattacharya, P. K. R. Maddikunta, and M. Liyanage, "Federated learning for big data: A survey on opportunities, applications, and future directions," *arXiv preprint arXiv:2110.04160*, 2021.

[232] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[233] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 965–978, IEEE, 2022.

[234] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[235] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," in *International Conference on Machine Learning*, pp. 11814–11827, PMLR, 2022.

[236] J. Deng, J. Guo, D. Zhang, Y. Deng, X. Lu, and S. Shi, "Lightweight face recognition challenge," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[237] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 5962–5979, oct 2022.

[238] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, pp. 1106–1125, 2019.

[239] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Esteve, "Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *Speech and Computer: 20th International Conference, SPECOM 2018, Leipzig, Germany, September 18–22, 2018, Proceedings 20*, pp. 198–208, Springer, 2018.

[240] S. Mirkin, M. Jacovi, T. Lavee, H.-K. Kuo, S. Thomas, L. Sager, L. Kotlerman, E. Venezian, and N. Slonim, "A recorded debating dataset," *arXiv preprint arXiv:1709.06438*, 2017.

[241] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[242] A. Baevski *et al.*, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Sysrecognitiontems*, vol. 33, pp. 12449–12460, 2020.

[243] A. Shafahi *et al.*, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[244] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," 2017.

[245] E. Bagdasaryan *et al.*, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020.

[246] A. N. Bhagoji *et al.*, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*, PMLR, 2019.

[247] A. Adler, M. Geierhos, and E. Hobley, "Influence of training data on the invertability of neural networks for handwritten digit recognition," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2021.

[248] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2013.

[249] K. Psychogyios, T.-H. Velivassaki, S. Bourou, A. Voulkidis, D. Skias, and T. Zahariadis, "Gan-driven data poisoning attacks and their mitigation in federated learning systems," *Electronics*, vol. 12, no. 8, p. 1805, 2023.

[250] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J. yong Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," 2020.

[251] L. Xu *et al.*, "Information security in big data: privacy and data mining," *IEEE Access*, vol. 2, pp. 1149–1176, 2014.

[252] E.-M. Schomakers, C. Lidynia, and M. Ziefle, "All of me? users' preferences for privacy-preserving data markets and the importance of anonymity," *Electronic Markets*, vol. 30, no. 3, pp. 649–665, 2020.

[253] N. Papernot and I. Goodfellow, "Privacy and machine learning: Two unexpected allies," 2021.

[254] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[255] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[256] L. J. Aslett, P. M. Esperança, and C. C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," 2015.

[257] F. Bourse *et al.*, "Fast homomorphic evaluation of deep discretized neural networks," in *Annual International Cryptology Conference*, Springer, 2018.

[258] C. Zhao *et al.*, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[259] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019.

[260] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pp. 9–14, IEEE, 2010.

[261] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Human activity detection from rgbd images," *plan, activity, and intent recognition*, vol. 64, 2011.

[262] B. Ni, G. Wang, and P. Moulin, "Rgbd-hudaact: A color-depth video database for human daily activity recognition," in *Consumer Depth Cameras for Computer Vision*, pp. 193–208, Springer, 2013.

[263] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1290–1297, IEEE, 2012.

[264] Z. Cheng, L. Qin, Y. Ye, Q. Huang, and Q. Tian, "Human daily action analysis with multi-view and color-depth data," in *European Conference on Computer Vision*, pp. 52–61, Springer, 2012.

[265] S. Essid, X. Lin, M. Gowing, G. Kordelas, A. Aksay, P. Kelly, T. Fillon, Q. Zhang, A. Dielmann, V. Kitanovski, R. Tournemenne, A. Masurelle, E. Izquierdo, N. E. O'Connor, P. Daras, and G. Richard, "A multi-modal dance corpus for research into interaction between humans in virtual environments," *Journal on Multimodal User Interfaces*, vol. 7, pp. 157–170, Mar 2013.

[266] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[267] O. Oreifej and Z. Liu, "HON4D: Histogram of oriented 4d normals for activity recognition from depth sequences," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 716–723, June 2013.

[268] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu, "Modeling 4d human-object interactions for event and object recognition," in *2013 IEEE International Conference on Computer Vision*, pp. 3272–3279, IEEE, 2013.

[269] G. Yu, Z. Liu, and J. Yuan, "Discriminative orderlet mining for real-time recognition of human-object interaction," in *Asian Conference on Computer Vision*, pp. 50–65, Springer, 2014.

[270] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, "Cross-view action modeling, learning and recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2649–2656, 2014.

[271] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian, "HOPC: Histogram of oriented principal components of 3d pointclouds for action recognition," in *European Conference on Computer Vision*, pp. 742–757, Springer, 2014.

[272] K. Wang, X. Wang, L. Lin, M. Wang, and W. Zuo, "3d human activity recognition with reconfigurable convolutional neural networks," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 97–106, ACM, 2014.

[273] C. Chen, R. Jafari, and N. Kehtarnavaz, "UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 168–172, IEEE, 2015.

[274] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian, "Histogram of oriented principal components for cross-view action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[275] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

Appendix A

Implementation Tools and Practical Aspects

A.1 Review of Tools and Libraries for FL

In our effort to answer the question: 'Is there a framework that can overcome the aforementioned limitations?', we proceeded to evaluate various open-source FL frameworks based on the following criteria: a) Node type: Where does the training take place (e.g., edge devices, smartphones, personal computers, or cloud servers)? b) Topology: Should training be conducted on remote devices, or is local execution sufficient for an experimental setup? c) Data Privacy: Is protection required from specific security mechanisms to ensure data privacy? d) GPU: Is GPU execution necessary to accelerate training with a large number of samples or large models? e) Community: Is it supported by a broad community and well-documented tools that allow for quick adaptation to the new framework and assist with bug detection and development? Within this framework, there's a demand for a tool that effectively manages and secures the connections between various data owners and the central entity. This tool is also responsible for coordinating the myriad tasks vital for successful federated training. It plays a key role in setting up communication protocols and interfaces with each component of the system. Initially, the tool acquires the necessary code files for the training process. These files, which might include training and evaluation scripts or configuration documents, are usually stored in a code versioning system like GitLab, which is capable of tracking different versions of the same code. Once secure communication is established and the new data training is complete, the model validation outcomes are fed into an experiment tracking system, like Weights and Biases or MLFLow (further details can be found in Section A.2 below). This setup enables the tracking of performance metrics and hyperparameters for various training prototypes via a user-friendly interface included in the tool. Additionally, the tool serves as a model registry, allowing for the storage of each experimental iteration's model as annotated artifacts in a centralized repository. This arrangement facilitates comprehensive oversight of the entire model development and prototyping process, with all the different model versions and their respective evaluation metrics systematically organized within a single platform.

A.1.1 FL Platform

Operational Model

Having explained the FL platform macroscopically, we are ready to dig deeper into the interrelated processes that take place between the data-owning party (i.e. "FL node") and the data scientist (i.e. "FL server"). The initial step involves configuring and deploying a central orchestration tool that establishes connections between multiple nodes, ensuring secure data transmission within the system. In preparing for the FL lifecycle, annotated data is placed in a specific directory designated for this purpose. Concurrently, the necessary code for data preparation, model training, and validation is organized and readied for execution. This setup phase is critical for the smooth functioning of the process. Once the data and code are in place, the system automatically detects the availability of new data versions, signaling readiness for training. This notification marks the beginning of the iterations in the FL process, indicating that all preliminary configurations are complete and the system is primed for the next stages of the learning cycle as detailed in the dissertation. When there's a need to fine-tune the model with a new version of data, the relevant code for training is prepared. This involves adjusting the training and validation scripts located in the code versioning component and configuring the training hyperparameters. Triggering the FL orchestration tool then initiates the process, where the model is sent to the participating nodes holding the data. This procedure is replicated across multiple data parties, creating a cycle that includes continuous notifications about new data versions and subsequent training requests. This loop facilitates an ongoing exchange, ensuring that the model is consistently updated and improved with the latest data inputs. Scaling down to a single data party scenario, we engage in a secondary loop within the primary one, focusing on the specific steps and metadata exchanges within this context. Here's how the process unfolds: Once the model reaches the data party, it undergoes training. After training, the metadata or training logs, along with the saved artifacts, are fed into the ML lifecycle management tool, making them accessible for analysis. Assuming the training meets expectations, the resulting model is then combined with models from other nodes, with the data scientist choosing the appropriate weight aggregation method. Next, the merged model and the validation script are sent back to the data party for performance evaluation. The results of this evaluation, documented in the validation logs, are then integrated into the experiment management tool. In the final phase, the data scientist reviews these outcomes to determine if the performance metrics meet the required standards. If not, the original model is reinitiated for another training cycle. Conversely, if the model's performance is satisfactory, it's stored in the model registry, ready for retrieval and deployment for future inference needs.

FL Orchestrator

The orchestration process in FL involves managing and coordinating the various aspects of applying learning technologies. The orchestrator, a key player in this setup, is tasked with managing device connections via secure communication protocols. This role includes considering specific criteria for device participation, such as the optimal number of devices, reliable internet connections, and other factors. The orchestrator's responsibilities extend to handling potential failures in the FL process, like device crashes, to ensure uninterrupted training progress. While many orchestration strategies presuppose complete device participation in every training round, the reality is that full participation does not always lead to successful training convergence. Therefore, an effective orchestrator must account for the diverse characteristics of each device, including data variability, computational capacity, and internet connectivity, to develop the most effective device selection strategy. This comprehensive analysis is crucial in formulating the most effective device selection strategy. Thus, the orchestrator's role is pivotal in answering our initial query about navigating the challenges in FL, ensuring that each device selected for participation is optimal based on the collective analysis of these varied parameters. There are four notable open-source technologies that can fulfil the role of an FL orchestrator, each having been validated against the aforementioned criteria. These technologies include the Federated AI Technology Enabler (FATE) [1], Google TensorFlow Federated (TFF) [2], OpenMined PySyft [3], and NVIDIA Federated Learning Application Runtime Environment (NVFLARE) [4]. Each of these frameworks offers unique features and capabilities that align with the key requirements for effective orchestration in FL environments. Their validation against criteria such as device participation, secure communication, and efficient management of heterogeneous device properties ensures they are well-equipped to handle the complexities and challenges of orchestrating FL processes.

Federated AI Technology Enabler (FATE): FATE is an open-source project initiated by Webank's AI Department to provide a secure computing framework to support the federated AI ecosystem. It implements secure computation protocols based on homomorphic encryption and multi-party computation (MPC). It supports FL architectures and secure computation of various ML algorithms, including logistic regression, tree-based algorithms, DL and transfer learning. Google TensorFlow Federated (TFF): TensorFlow Federated (TFF) is an open-source framework for ML and other computations on decentralized data. TFF has been developed to facilitate open research and experimentation with FL, an approach to ML where a shared global model is trained across many participating clients that keep their training data locally. For example, FL has been used to train prediction models for mobile keyboards without uploading sensitive typing data to servers. OpenMined PySyft: Pysyft is an open-source library that enables

[1]https://github.com/FederatedAI/FATE
[2]https://github.com/tensorflow/federated
[3]https://github.com/OpenMined/PySyft
[4]https://github.com/NVIDIA/NVFlare

FL by extending DL frameworks such as PyTorch in a transparent, lightweight, and user-friendly manner. PyGrid is a peer-to-peer network in order for multiple devices to collectively train AI models using the PySyft library. Its device in order to establish a secure connection with the PyGrid network has to create credentials and provide a token for authentication. NVFLARE: NVFLARE is a domain-agnostic, open-source FL platform, developed by NVIDIA, used by researchers and data scientists to adapt existing ML/DL workflow (Pytorch, Tensorflow) to a federated paradigm and enables developers to build a secure, privacy-preserving offering for a distributed multi-party collaboration. Flower [5]: Flower is an open-source FL framework. It is designed to be flexible and agnostic to client hardware and ML frameworks, making it adaptable for a wide range of use cases and environments. Flower facilitates the development and implementation of FL systems by allowing researchers and developers to run experiments across diverse computational settings and with various ML algorithms.

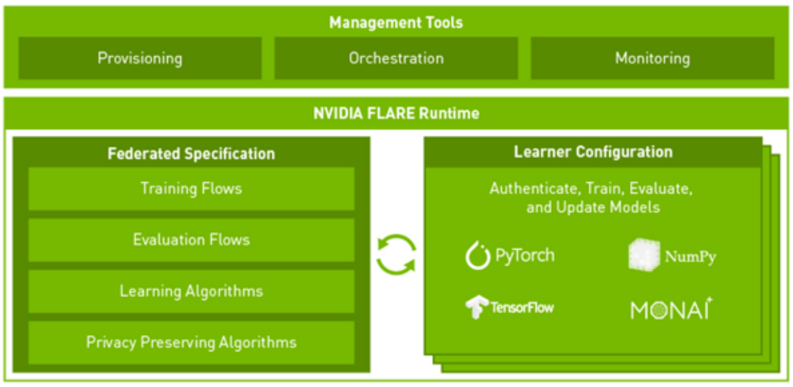


Figure A.1: NVIDIA FL Platform.

Figure A.1 illustrates that NVFLARE offers a comprehensive suite of tools and functionalities for creating a tailored FL environment. It includes extensible management tools that support secure provisioning through SSL certifications, efficient orchestration via an administrative console, and detailed monitoring of the FL process using various visualization techniques. NVFLARE also features pre-built workflow strategies for both training and evaluation, which can be integrated with learning algorithms for effective model aggregation. Additionally, it encompasses privacy-preserving algorithms designed to safeguard sensitive data and thwart attempts to reverse-engineer ML models. In NVFLARE, provisioning serves to create a trusted configuration that is system-wide and applicable to all participants, enabling them to engage in the FL process from various locations. This involves generating configuration files that contain essential network information like domain names, IP addresses, and ports, as well as authentication credentials such as participant certificates. Data scientists or tool developers, who act as

[5]https://flower.dev

administrators in NVFLARE, are responsible for customizing these files according to their specific needs. They then distribute these configurations to the participants, including the Orchestrator and various devices, to set up a secure and reliable connection. Once this connection is in place, the administrator takes on the role of orchestrating the FL process, which includes initiating, halting, or restarting the distributed training, among other tasks.

A.1.2 Privacy Preserving Capabilities of FL Frameworks

In this section, we will evaluate the FL frameworks described previously, with a particular emphasis on their capabilities for preserving privacy. FATE: It is designed to cater to industrial needs, offering secure computation capabilities for various ML tasks. FATE achieves this by incorporating technologies like Multiparty Computation and Homomorphic Encryption, making it a robust choice for FL applications in business contexts. PySyft: For implementing privacy-preserving mechanisms with PySyft, integrating the PyGrid framework is essential. PyGrid, designed as a peer-to-peer network, facilitates ML and data privacy. It supports DP and SMC, utilizing HE to achieve these privacy goals. TensorFlow Federated: Included within this framework is TensorFlow Privacy, a Python library that facilitates the application of privacy techniques during ML model training. TensorFlow Privacy particularly focuses on implementing Differential Privacy by introducing Gaussian or Laplacian noise into the data, depending on specific values, to enhance privacy protections. NVIDIA Flare: Equipped with the necessary features to establish a secure FL environment and supports a variety of privacy-preserving methods, including DP and HE, to facilitate efficient and secure multi-party distributed collaboration. PaddleFL [6] is a recognized FL framework notable for its versatility in handling various FL tasks and its suitability for large-scale deployments. It also provides support for secure aggregation and incorporates privacy-preserving measures like Differential Privacy and secure multiparty computations, enhancing its utility in diverse FL scenarios. Sherpa.ai [7], is an open-source framework designed for FL. It facilitates the training of ML models in a federated setup and integrates Differential Privacy into the process. Additionally, several popular FL frameworks exist that do not include privacy-preserving features. For instance, Flower is an open-source, client-agnostic FL framework known for its expandability and scalability. It allows researchers to experiment with and develop tailor-made solutions, though it lacks built-in privacy-preserving mechanisms. The above-described frameworks are summarized in Table A.1, by presenting the main pros and cons of different FL Frameworks regarding the training of ML models with privacy-preserving capabilities.

Based on the previous analysis, it's evident that each framework offers distinct privacy-preserving methods. To determine the most suitable framework for our experimental needs, a comparison table has been developed. This table, referred to as

[6] https://github.com/PaddlePaddle/PaddleFL

[7] https://sherpa.ai

Table A.1: Pros and Cons of different Federated Learning frameworks focusing on privacy-preserving aspects.

FL Framework	**Pros**	**Cons**
PySyft & PyGrid	Easy to use Provides privacy preserving techniques, like Differential Privacy, Multi-Party Computation and Homomorphic encryption	PySyft is only for 1 server and 1 client (Duet) Run only in simulation mode For real FL scenarios, the PyGrid is needed
TensorFlow Federated	Seamless integration with existing TensorFlow ML models Supports secure aggregation and differential privacy	It can be used only in simulation mode Neither Homomorphic Encryption nor Multi-party computation
NVIDIA Flare	Provides privacy-preserving methods such as DP, Homomorphic Encryption, and MPC Training of ML model between 1 server and many remote clients GPU training support It is customizable, supporting both TensorFlow and PyTorch Good documentation	It is a new framework, and the support community is not that big
FATE	Production Ready Provides many FL algorithms Secure computation protocols, based on Homomorphic Encryption and Multi-Party Computation	It does not establish any differential privacy algorithms Doesn't use GPUs for training
PaddleFL	Provides enough privacy-preserving methods such as DP, MPC and secure aggregation	Has poor documentation and has a small community
Sherpa.ai	Provides differential privacy It can be easily customizable	Can run only in simulation mode Limited application scenarios
Flower	Supports a great number of clients It is customizable	It does not support privacy preserving techniques.

Table A.2, outlines the specific privacy-preserving mechanisms supported by each of the analyzed frameworks.

Table A.2: Privacy preserving methods of different frameworks.

FL Framework	**SA**	**DP**	**HE**	**SMC**
PySyft & PyGrid	x	x	x	x
TensorFlow Federated	x	x		
NVIDIA Flare	***x***	***x***	***x***	***x***
FATE	x		x	x
PaddleFL	x	x		x
Sherpa.ai		x		
Flower	x			

Based on these insights, NVIDIA Flare has been chosen as the FL framework for this dissertation, as it stands out among other open-source options in terms of its comprehensive features. NVIDIA Flare enables seamless integration of existing ML and DL models from TensorFlow or PyTorch into a federated system, facilitating model training across multiple remote clients with GPU support. Furthermore, it addresses security concerns related to model exchanges by ensuring the confidentiality and integrity of data, and safeguarding against threats such as data and model poisoning attacks. This level of security is attributed to the robust privacy-preserving mechanisms incorporated within the NVIDIA Flare framework.

A.2 Tools and Libraries for FL Implementation

A.2.1 FL Topologies and Design Principles

FL involves distributing the training process across a network of nodes, and the arrangement of this network can significantly impact the model training. The various topologies used in FL differ in factors such as the aggregation algorithm and distribution of the model, the number of communication links needed for training, and the associated costs of setting up the training system. It is possible to implement FL with or without a central server for orchestration. Three common network topologies used in FL are the *Star* topology, *Ring* topology, and *Hybrid* topology, as documented in Ganapathy's work [21]. In a *star-like topology* for FL, each node communicates with a central server and only shares small updates to the model with the server. This allows for asynchronous learning, where nodes can train independently without waiting for other nodes. Adding new nodes is straightforward, but the speed of training can vary due to differences in device configurations and the number of training samples. Each node requires two communication links for each round of communication: one to upload local updates and another to download the global model from the server. In the *Ring-like topology* for FL, devices are connected in a circular data path where each node is connected to two others. The training process starts with the first node, and a global model is initialized across all clients. Each node updates the model weights after a fixed number of local iterations and transfers them to the next adjacent node. This cyclical weight transfer is repeated until convergence, and the output of the last node is the final model that needs to be redistributed among all clients. This approach follows synchronous learning, as nodes depend on their peer nodes, and computational capabilities should be comparable across all nodes. As this topology does not require a central server, the cost of infrastructure development is lower, and fewer communication links are needed. The *hybrid topology* combines the star-like and ring-like topologies by involving both nodes and a server. Nodes are grouped together and arranged sequentially, and the first node in each group begins training asynchronously using their local data. Each node sends its model updates to the next adjacent node in the group after training for a pre-defined number of local epochs. This process is repeated for all nodes in each group, following the ring-like topology. The hybrid topology uses a combination of synchronous and asynchronous learning, and the computational power of the server can be compromised compared to the star-like topology. If the number of nodes is equal to the number of groups, it is equivalent to FL in star-like topology.

A.2.2 FL Platform Processes and Workflow

Before starting FL, the data needs to be annotated and split into training and validation sets. A tool is needed to establish communication protocols and orchestrate the various tasks necessary for federated training. This tool retrieves code files, executes the training process, and stores the results in an experiment tracking component. The specific tool also acts as a model registry, storing the resulting model version in a central repository. The best-performing model is chosen and fused with its previous version using a state-of-the-art weight aggregation method, defined during the configuration of the orchestration component. Lastly, the fused model is then served back to the FL platform, ready for deployment. The aforementioned process is illustrated in Figure A.2.

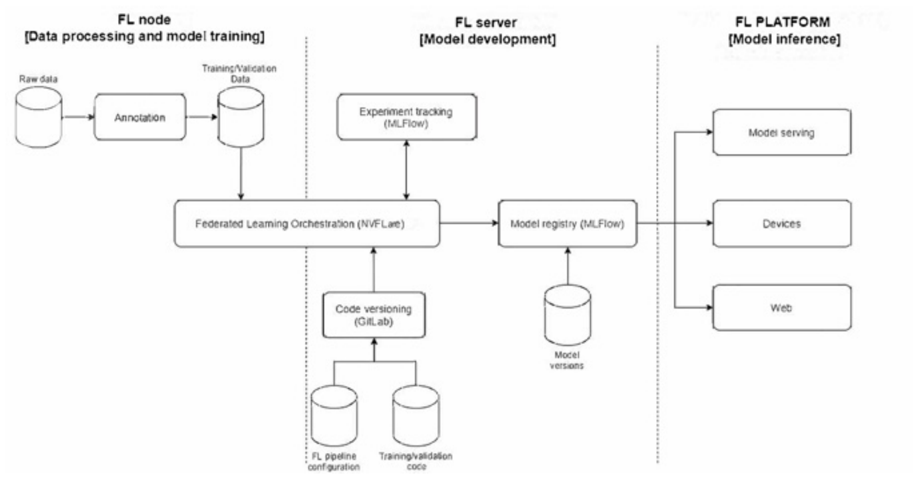


Figure A.2: Building Block View of FL Platform.

Figure A.3 shows the workflow for FL, which involves configuring and deploying the central orchestration tool to establish connections between nodes and ensure information security. Annotated data is placed in a specific directory, and the data scientist organizes code for tasks such as data preparation and model training. Once new training data is in the data owner's directory, the FL orchestration tool is triggered, and the model is sent to the data owner party. After training, metadata and artifacts are ingested into the ML lifecycle management tool. The FL server merges the locally trained models of each node utilizing a weighted aggregation method. The aggregated global model and validation script are sent back to the data owner party for evaluation, and results are assessed by the data scientist. If necessary, the loop starts over with new training. If the model performs well, it is stored in the model registry for deployment in inference.

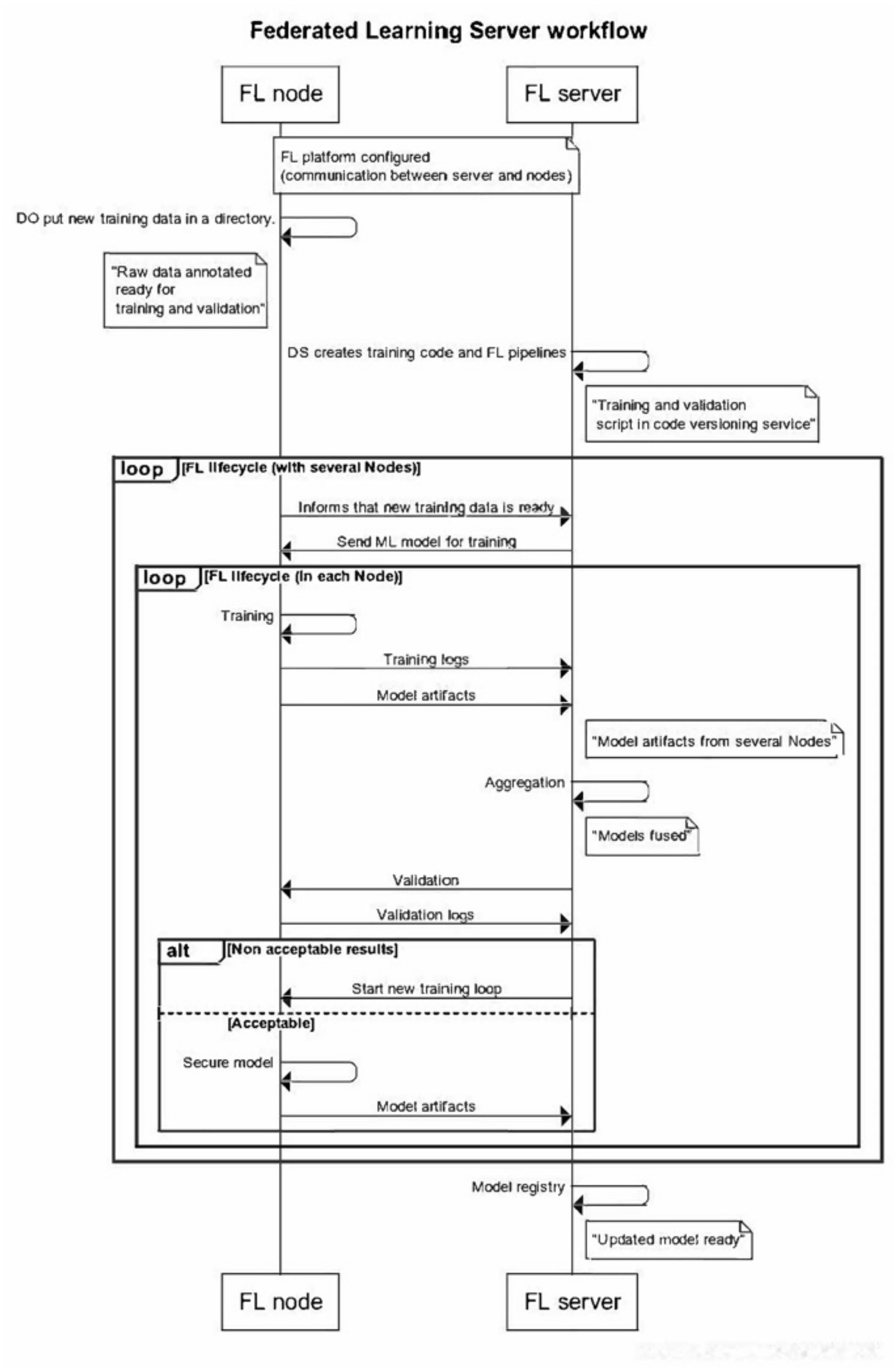

Federated Learning Server workflow
FL node
FL server
FL platform configured (communication between server and nodes)
DO put new training data in a directory.
"Raw data annotated ready for training and validation"
DS creates training code and FL pipelines
"Training and validation script in code versioning service"
loop [FL lifecycle (with several Nodes)]
Informs that new training data is ready
Send ML model for training
loop [FL lifecycle (in each Node)]
Training
Training logs
Model artifacts
"Model artifacts from several Nodes"
Aggregation
"Models fused"
Validation
Validation logs
alt [Non acceptable results]
Start new training loop
[Acceptable]
Secure model
Model artifacts
Model registry
"Updated model ready"
FL node
FL server

Figure A.3: Workflow view of FL.

A.2.3 FL Framework

The FL orchestration process refers to the coordination activities performed while applying learning technologies. The orchestrator is the actor in FL who is responsible for accepting and forwarding device connections through a secure communication protocol while taking into consideration certain participation criteria such as the optimal number of participating devices, fast internet connection speed and *etc.* It is responsible for handling failures in the FL process, including connected devices crashing to ensure the training will continue to make progress. Most of the orchestration strategies assume full device participation, that is, all devices participate in every training round. In practice, the participation of all devices does not guarantee the convergence of the FL training process. An efficient orchestrator must take into consideration insights regarding each device's heterogeneous properties, considering data heterogeneity and bias, computational resources, and Internet connection, in order to implement the most efficient device selection strategy.

The NVFLARE framework [8] has been selected for the development of the FL platform, as it supports both DL and traditional ML algorithms, as well as horizontal and vertical FL. It includes built-in FL aggregation algorithms (*e.g.*, FedAvg, FedProx, FedOpt, Scaffold, Ditto) and supports multiple training and validation workflows (*i.e.* global model evaluation, cross-site validation), data analytics, and ML lifecycle management. NVFLARE also offers privacy preservation with differential privacy and homomorphic encryption, as well as security enforcement through federated authorization and privacy policy. It is easily customizable and extensible and can be deployed on the cloud and on-premise. Additionally, NVFLARE includes a simulator for rapid development and prototyping, a dashboard UI for simplified project management and deployment, and built-in support for system resiliency and fault tolerance.

ML lifecycle management

The ML lifecycle involves six processes, including Experimentation and Prototyping, which is essential for achieving the best model. Model management is necessary to handle different model-variants and experiments, track ML metadata, and govern model deployment. MLflow [9] is an open-source tool for ML lifecycle management that consists of four components, including MLflow Tracking and MLflow registry. MLflow enables remote monitoring of training logs, tracking of hyperparameters, evaluation of models against quality measures and fairness indicators, central storage and retrieval of models, continuous evaluation of deployed models, and traceability, debugging, and reproducibility of potential issues. For FL, MLflow Tracking and MLflow registry were deployed, providing a UI that facilitates the interaction with the MLflow server.

A.2.4 Infrustructure and Implementation Details

Regarding the hardware infrastructure, for the FL experiments, we used 5 physical nodes located at different locations across EU (Spain, Greece, Portugal and Cyprus). Following the star-like topology (see Figure A.4), every local node connects to the central server and establishes a one-to-one communication. In our case, a central server has been

[8]https://nvidia.github.io/NVFlare/

[9]https://mlflow.org/

selected among the 5 nodes that acts as the aggregator server for all the FL experiments and the other devices are the clients that hold the local datasets. Except for the physical nodes, we created an extra virtual client (*Dummy* node) that contains a global dataset and is used only for evaluation purposes. Overall, we have 1 aggregator server, 5 nodes for both training and evaluation and 1 virtual node only for validation. Secure communication between clients and the server is established through a VPN connection to ensure that sensitive data (parameters) is safely transmitted and prevent unauthorised access. All the devices that participate in the FL process are almost identical and have the same resources. The aggregator server has 5 Nvidia RTX 3090 GPUs with 24GB VRAM, 2 nodes (Spain) have 4 Nvidia RTX 3090 GPUs with 24GB VRAM, 1 node (Greece) has 4 Nvidia RTX A5000 GPUs with 24GB VRAM, last two nodes (Cyprus and Portugal) are equipped with 4 Nvidia RTX 3090 GPUs and 24GB VRAM.

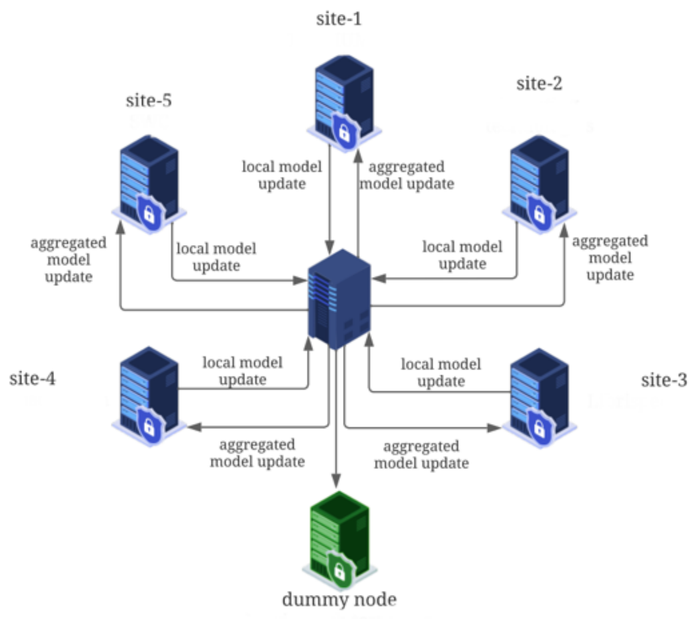


Figure A.4: The adopted Star-like topology that utilizes a centralized server to create and share the global model, while five peripheral nodes participate asynchronously in the training process.

A.2.5 Workflow Adaptations

Unlike previous studies that conduct mostly simplified Proof of Concept (PoC) experiments in federated settings, usually involving virtual nodes, in this work we report the results and the analysis of large-scale FL experiments for various AI tools, using the specific hardware infrastructure described, under realistic and demanding scenarios. Since we conducted the experiments under realistic conditions in physical nodes, we followed a more sophisticated FL approach than starting the process, training the models and evaluating the performance. Firstly, the network configuration file is created to provide information related to participants including domain names, port numbers or ID addresses that will be used for the connection with the server. For each different tool's experiments, we use a separate set of ports to avoid conflicts during models training in parallel. Each developer creates for each of his/her tools a Dockerfile that contains the required information and packages for the communication establishment with the aggregator and the deployment of the tools to clients' sites. The Dockerfile is sent to each client device through the VPN connection to be built. Once all the clients have the required information, the FL process starts and follows the FL server's and clients' configurations.

The NVFLARE framework offers built-in methods and algorithms but also gives the capability to build custom methods dedicated to specific needs. In our case, our tools cover a large range of different tasks and data types including *Face Re-identification (Face ReID)*, *Video Super-Resolution (VSR)*, *Named Entity Recognition (NERC)* and *Audio Speech Recognition (ASR)* and is necessary to develop some custom methods to handle the local training and evaluation, and global aggregation process.

FL Server (Aggregator) Stack: As we mentioned before, we created a virtual (*Dummy*) node only for evaluation purposes. The node holds a global dataset that is used to evaluate the performance and generalisation of both the locally trained and the global aggregated models. Since the '*Dummy*' node does not participate in the training process we built a custom aggregation method that can identify and exclude the weights of the *Dummy* node from the aggregation process. Moreover, the built-in JSON generator function, which is used to report the results of the FL process in a JSON format, was modified to enable the usage of MLFlow for the tracking of the training and evaluation phases by logging parameters, code versions, metrics, and output files. Last, besides the evaluation of the local models on their local test sets and the global test set, we also conducted the cross-site evaluation, when all the clients had finished training. During the cross-site model evaluation, every client validates other clients' models and the global model on their local datasets.

FL Client (Data Owner) Stack: The client configuration files contain mainly all the parameters that are required for the local training process, including model and data loader arguments, number of local epochs, etc. These parameters are different for each tool and the developer is responsible to choose the more suitable ones and create the client configuration files. The only extra arguments that have been added are a few parameters related to the MLFlow framework for results reporting.

Appendix B

Best Practices and Guidelines for Implementation

The guidelines for using NVIDIA FLARE with PyTorch offer a comprehensive and detailed roadmap for researchers and developers looking to implement FL solutions. This document covers essential requirements, outlines the necessary folder structure, and provides step-by-step instructions for custom file modifications. Furthermore, it delves into the specifics of integrating code with NVIDIA FLARE, ensuring a seamless setup process. With a focus on practical implementation, these guidelines are designed to facilitate the development of FL applications, leveraging the advanced capabilities of NVIDIA FLARE in a PyTorch environment.

B.1 Installing NVFLARE for FL training on virtual nodes

B.1.1 Requirements

NVFLARE requires python >= 3.8.10

Install *pip3 install nvflare*, in the virtual environment builder to run your tool

B.1.2 Folder Structure

FL application: (e.g. Image classification) The FL application folders can have the below structure: FL_application_name (e.g. image_classification) config: contains client and server configurations config_fed_client.json config_fed_server.json custom: contains the custom components of your application network.py dataset.py trainer.py pt_model_locator.py pt_constants.py ... any other file related to your application

B.1.3 Modifications on Your Custom Files

Network & Dataset

The class that creates your model (e.g. Network class) is not related to NVIDIA FLARE, so no changes are required. Important note: Model class should be nn.Module. Errors may occur for other types of inherited classes.

```
class SimpleNetwork(nn.Module):
    def __init__(self, kernel):
        super(SimpleNetwork, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, kernel)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, kernel)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Dataset/DataLoader classes for dataset preprocessing are not related to NVIDIA FLARE and no changes are required.

Setup

To integrate your code into the NVIDIA FLARE a custom class OurTrainer should be implemented as an NVIDIA FLARE Executor. Class OurTrainer works similarly to the main function that you probably use to handle the arguments and set up the optimizer, loss function and local training. All the parameters required for training should be passed as arguments to the init() method OurTrainer can take the arguments' values from the client configuration file

```
class OurTrainer(Executor):
    def __init__(self, lr, epochs, kernel,
        train_task_name = AppConstants.TASK_TRAIN,
        submit_model_task_name=AppConstants.TASK_SUBMIT_MOD
        super(OurTrainer, self).__init__()
        self.lr = lr
        self.epochs = epochs
        self.kernel = kernel
        self.train_task_name = train_task_name
        self.submit_model_task_name = submit_model_task_name
        self.exclude_vars = exclude_vars
        # Training setup
        self.model = SimpleNetwork(self.kernel)
        self.device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu") self.model.to(self.device)
        self.loss = nn.CrossEntropyLoss()
        self.optimizer = SGD(self.model.parameters(), lr=lr, momentum=0.9)
        # Set dataset
        self.train_dataset = datasets.OurDataset_train(...)
        self.train_loader = torch.utils.data.DataLoader(self.train_dataset,
        batch_size=64, shuffle=True, num_w
        # Setup the persistence manager to save PT model.
        # The default training configuration is used by persistence manager
        # in case no initial model is found.
        self._default_train_conf = {"train": {"model": type(self.model).__name__}}
        self.persistence_manager = PTModelPersistenceFormatManager(
            data=self.model.state_dict(), default_train_conf = self._default_train_conf)

    def local_train(self, fl_ctx, weights, abort_signal):
    # Set the model weights self.model.load_state_dict(state_dict=weights)
        # Basic training
        for epoch in range(self._epochs):
            running_loss = 0.0
            for i, batch in enumerate(self._train_loader):
```

Everything up to this point is completely independent of NVIDIA FLARE. To integrate your local train code into the NVIDIA FLARE API, implement the method execute which is called every time the client receives an updated model from the server with the task "train". This is a typical execute function that you can use exactly as it is to enable FL. Indeed, you can make any modification you want to cover your needs

```
def execute(self, task_name: str, shareable: Shareable, fl_ctx: FLContext, abort_signal: Signal) -> Shareable:
    try:
        if task_name == self._train_task_name:
            # Get model weights
            try:
                dxo = from_shareable(shareable)
            except:
                self.log_error(fl_ctx, "Unable to extract dxo from shareable.")
                return make_reply(ReturnCode.BAD_TASK_DATA)

            # Ensure data kind is weights.
            if not dxo.data_kind == DataKind.WEIGHTS:
                self.log_error(fl_ctx, f"data_kind expected WEIGHTS but got {dxo.data_kind} instead.")
                return make_reply(ReturnCode.BAD_TASK_DATA)

            # Convert weights to tensor. Run training
            torch_weights = {k: torch.as_tensor(v) for k, v in dxo.data.items()}
            self.local_train(fl_ctx, torch_weights, abort_signal)

            # Check the abort_signal after training.
            # local_train returns early if abort_signal is triggered.
            if abort_signal.triggered:
                return make_reply(ReturnCode.TASK_ABORTED)

            # Save the local model after training.
            self.save_local_model(fl_ctx)

            # Get the new state dict and send as weights
            new_weights = self.model.state_dict()
            new_weights = {k: v.cpu().numpy() for k, v in new_weights.items()}

            outgoing_dxo = DXO(data_kind=DataKind.WEIGHTS, data=new_weights,
                               meta={MetaKey.NUM_STEPS_CURRENT_ROUND: self._n_iterations})
            return outgoing_dxo.to_shareable()
        elif task_name == self._submit_model_task_name:
            # Load local model
            ml = self.load_local_model(fl_ctx)

            # Get the model parameters and create dxo from it
            dxo = model_learnable_to_dxo(ml)
            return dxo.to_shareable()
        else:
            return make_reply(ReturnCode.TASK_UNKNOWN)
    except:
        self.log_exception(fl_ctx, f"Exception in simple trainer.")
        return make_reply(ReturnCode.EXECUTION_EXCEPTION)


def save_local_model(self, fl_ctx: FLContext):
    run_dir = fl_ctx.get_engine().get_workspace().get_run_dir(fl_ctx.get_prop(ReservedKey.RUN_NUM))
    models_dir = os.path.join(run_dir, PTConstants.PTModelsDir)
    if not os.path.exists(models_dir):
        os.makedirs(models_dir)
    model_path = os.path.join(models_dir, PTConstants.PTLocalModelName)

    ml = make_model_learnable(self.model.state_dict(), {})
    self.persistence_manager.update(ml)
    torch.save(self.persistence_manager.to_persistence_dict(), model_path)


def load_local_model(self, fl_ctx: FLContext):
    run_dir = fl_ctx.get_engine().get_workspace().get_run_dir(fl_ctx.get_prop(ReservedKey.RUN_NUM))
    models_dir = os.path.join(run_dir, PTConstants.PTModelsDir)
    if not os.path.exists(models_dir):
        return None
    model_path = os.path.join(models_dir, PTConstants.PTLocalModelName)
    self.persistence_manager = PTModelPersistenceFormatManager(data=torch.load(model_path),
                                                               default_train_conf=self._default_train_conf)
    ml = self.persistence_manager.to_model_learnable(exclude_vars=self._exclude_vars)
    return ml
```

Server & Client Configuration files

These .json files contain all the components that the server and the clients need for the FL training process. Most of them are NVIDIA FLARE built-in components. In the files below, the components that are used are the minimum ones required for the process.
config_fed_server.json

```
{
  "format_version": 2,

  "server": {
    "heart_beat_timeout": 600
  },
  "task_data_filters": [],
  "task_result_filters": [],
  "components": [
    {
        "id": "model",
        "path": "models.SimpleNetwork",
        "args": {
            "kernel": 5,
        }
    },
    {
        "id": "persistor",
        "name": "PTFileModelPersistor",
        "args": {
            "model": "model"
        }
      },
    {
        "id": "shareable_generator",
        "path": "nvflare.app_common.shareablegenerators.full_model_shareable_generator.FullModelShareableGenerator",
        "args": {}
    },
    {
        "id": "aggregator",
        "path": "nvflare.app_common.aggregators.intime_accumulate_model_aggregator.InTimeAccumulateWeightedAggregator",
        "args": {
            "expected_data_kind": "WEIGHTS"
      }
    },
    {
        "id": "model_locator",
        "path": "pt_model_locator.PTModelLocator",
        "args": {
      }
    }
  ],
  "workflows": [
      {
          "id": "scatter_and_gather",
          "name": "ScatterAndGather",
          "args": {
              "min_clients" : 1,
              "num_rounds" : 2,
              "start_round": 0,
              "wait_time_after_min_received": 10,
              "aggregator_id": "aggregator",
              "persistor_id": "persistor",
              "shareable_generator_id": "shareable_generator",
              "train_task_name": "train",
              "train_timeout": 0
        }
      }
  ]
}
```

config_fed_server.json

```
{
  "format_version": 2,

  "executors": [
    {
      "tasks": ["train", "submit_model"],
      "executor": {
        "path": "trainer.OurTrainer",
        "args": {
            "lr": 0.001,
            "epochs": 3,
            "kernel": 5
            }
        }
      }
  ],
  "task_result_filters": [
  ],
  "task_data_filters": [
  ],
  "components": [
  ]
}
```

Extra files

The custom folder contains two extra files, the pt_model_locator.py and the pt_constants.py, as provided below.

```python
import os
from typing import List
import torch.cuda
from nvflare.apis.dxo import DXO
from nvflare.apis.fl_context import FLContext
from nvflare.app_common.abstract.model import model_learnable_to_dxo
from nvflare.app_common.abstract.model_locator import ModelLocator
from nvflare.app_common.pt.pt_fed_utils import PTModelPersistenceFormatManager
from pt_constants import PTConstants

class PTModelLocator(ModelLocator):

    def __init__(self, exclude_vars=None, model=None):
        super(PTModelLocator, self).__init__()

        self.model = model
        self.exclude_vars = exclude_vars

    def get_model_names(self, fl_ctx: FLContext) -> List[str]:
        return [PTConstants.PTServerName]

    def locate_model(self, model_name, fl_ctx: FLContext) -> DXO:
        if model_name == PTConstants.PTServerName:
            server_run_dir = fl_ctx.get_engine().get_workspace().get_app_dir(fl_ctx.get_run_number())
            model_path = os.path.join(server_run_dir, PTConstants.PTFileModelName)
            if not os.path.exists(model_path):
                return None
            try:
                # Load the torch model
                device = "cuda" if torch.cuda.is_available() else "cpu"
                data = torch.load(model_path, map_location=device)
                self.log_info(fl_ctx, f"Loaded {model_name} model from {model_path}.")
            except Exception as e:
                self.log_error(fl_ctx, f"Unable to load model: {e}.")

            # Setup the persistence manager.
            if self.model:
                default_train_conf = {"train": {"model": type(self.model).__name__}}
            else:
                default_train_conf = None

            # Use persistence manager to get learnable
            persistence_manager = PTModelPersistenceFormatManager(data, default_train_conf=default_train_conf)
            ml = persistence_manager.to_model_learnable(exclude_vars=None)
            # Create dxo and return
            return model_learnable_to_dxo(ml)
        else:
            self.log_exception(fl_ctx, f"PTModelLocator doesn't recognize name: {model_name}")
            return None

class PTConstants:
    PTServerName = "server"
    PTFileModelName = "FL_global_model.pt"
    PTLocalModelName = "local_model.pt"
```

B.1.4 Setting up the Application Environment and Training the model with virtual clients

To get started, run this command to generate a poc folder with a server, two clients, and one admin:

```
poc −n 2
```

Copy your application folder (e.g. image_classification) to the admin's working folder:

```
mkdir −p poc/admin/transfer
cp −rf .../application_folder/* poc/admin/transfer
```

Once you are ready to start the FL system, you can run the following commands to start all the different parties. You have to start the server first:

```
./poc/server/startup/start.sh
```

Once the server is running you can start the clients in different terminals (make sure your terminals are using the environment with NVIDIA FLARE installed).

```
./poc/site−1/startup/start.sh
./poc/site−2/startup/start.sh
```

In one last terminal, start the admin (default username, password -¿ admin):

```
./poc/admin/startup/fl_admin.sh
```

With the admin client command prompt successfully connected and logged in, enter the commands below in order.

```
upload_app application_name
```

Uploads the application from the admin client to the server's staging area.

```
set_run_number 1
```

Creates a run directory in the workspace for the run_number on the server and all clients. The run directory allows for the isolation of different runs so the information in one particular run does not interfere with other runs.

```
deploy_app application_name all
```

This will make your application the active one in the run_number workspace. After the above two commands, the server and all the clients know your application will reside in the run_1 workspace.

```
start_app all
```

This start_app command instructs the NVIDIA FLARE server and clients to start training. Once the fl run is complete and the server has successfully aggregated the clients' results after all the rounds, run the following commands in the fl_admin to shutdown the system.

```
shutdown all
```

B.2 NVIDIA FLARE remote training on distributed infrastructure

B.2.1 Procedure overview

Establish a secure connection between the server (central node) and multiple remote clients (node1, node2). Start the FL process. Train remotely the DL models for each tool.

B.2.2 Provisioning - Networking details

The purpose of provisioning in NVIDIA FLARE is to generate mutually trusted system-wide configurations for all participants so all of them can join the NVIDIA FLARE system across different locations. The configurations usually include, but are not limited to, the following information: network discovery, such as domain names, port numbers or IP addresses credentials for authentication, such as certificates of participants and root authority authorization policy, such as roles, rights and rules tamper-proof mechanism, such as signatures convenient commands, such as shell scripts with default command line options to easily start an individual participant.

B.2.3 Project yaml file

This is an example yaml file, describing participants and builders, dedicated to a sample distributed infrastructure. This is the key file that describes the information which the provisioning tool will be using to generate startup kits for server, clients and admins. IMPORTANT NOTE: To avoid any conflict between FL applications that will be running in parallel, each tool will use a predefined set of ports to establish a connection. The file with the predefined set of ports for each tool will be sent as soon as possible.

```
    api_version: 2
name: example_project
description: NVFlare sample project yaml file

participants:
  # change example.com to the FQDN of the server
  - name: central_server_name
    type: server
    org: nameOfOrg
    fed_learn_port: 8534
    admin_port: 8535
    # enable_byoc loads Python codes in the app.  Default is false.
    enable_byoc: true
  - name: node1
    type: client
    org: nameOfOrg
    enable_byoc: true
  - name: node2
    type: client
    org: nameOfOrg
    enable_byoc: true
  - name: admin@mail.com
    type: admin
    org: nameOfOrg
    roles:
      - super

# The same methods in all builders are called in their order defined in builders section
builders:
  - path: nvflare.lighter.impl.workspace.WorkspaceBuilder
    args:
      template_file: master_template.yml
  - path: nvflare.lighter.impl.template.TemplateBuilder
  - path: nvflare.lighter.impl.static_file.StaticFileBuilder
    args:
      # config_folder can be set to inform NVFlare where to get configuration
      config_folder: config
      # when docker_image is set to a docker image name, docker.sh will be generated on server/client/admin
      # docker_image:
  - path: nvflare.lighter.impl.auth_policy.AuthPolicyBuilder
    args:
      orgs:
        nameOfOrg:
          - relaxed
      roles:
        super: super user of system
      groups:
        relaxed:
          desc: org group with relaxed policies
          rules:
            allow_byoc: true
            allow_custom_datalist: true
      disabled: false
  - path: nvflare.lighter.impl.cert.CertBuilder
  - path: nvflare.lighter.impl.he.HEBuilder
    args:
      poly_modulus_degree: 8192
      coeff_mod_bit_sizes: [60, 40, 40]
      scale_bits: 40
      scheme: CKKS
  - path: nvflare.lighter.impl.signature.SignatureBuilder
  - path: nvflare.lighter.impl.workspace.DistributionBuilder
    args:
      zip_password: false
```

NOTES

- Please make sure that the FL server port number is accessible by all participating sides.
- This file is independent of your application and it should not be added to the application's folder.
- Please create a separate folder to put the file (e.g., workspaces).

B.2.4 Create your FL workspace

To create the secure workspace, please use the project.yml file to build a package and copy it to secure_workspace (in the admin site).

```
cd ./workspaces
provision -p ./project.yml
cp -r ./workspace/example_project/prod_00 ./secure_workspace
```

Workspace structure

```
    workspace
    example project
        prod\_00
        admin\_name
        startup
        server\_name
        startup
        client1\_name
        startup
        client2\_name
        startup
        ...
```

The prod_NN folders contain the provisioning results. The tool developer, who holds the role of the admin in the FL process, should send to each participant (central-1, node1, node2) the .zip file that contains the information required to establish a secured connection.

B.2.5 Set-up the FL environment of central server

Developers do not have permission to create a virtual environment (venv command) on the central server. To handle this limitation, developers should setup the environment using Docker. Below is a basic Dockerfile example that creates a Python env and installs NVFLARE package (required). Each tool requires different packages that should be included and installed.

```
FROM ubuntu:focal-20220113

    RUN apt-get update && apt-get upgrade -y

    RUN apt-get install -y python3 python3-pip

    # TODO: provide version numbers for every Python package
    RUN python3 -m pip install nvflare==2.0.16

CMD [python3]
```

Each participant(central-1, node1, node2) should build the Docker image by running the command:

```
docker build --network=host -f Dockerfile -t federated:latest .
```

Establish connection & start FL process

IMPORTANT: Please before starting the FL process, remove from each participant's (central-1, node1, node2) start.sh function

(/secure_workspace/participant_name/startup/start.sh) the symbol &, as highlighted below:

```
#!/usr/bin/env bash
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"
$DIR/sub_start.sh          #& removed
```

To get started, copy your application folder to the admin's working folder (on the distributed infrastructure, the admin is running on the central_server_name node)

```
mkdir -p secure_workspace/admin@nameOfOrg.com/transfer
cp -rf .../application_folder/* secure_workspace/admin@nameOfOrg.com/transfer
```

Once ready to start the FL system, you can run the following commands to start all the different parties. You have to start the server first. On the server's site run the command below:

```
docker run -it --network="host" --dns 10.41.41.1 -v
        `pwd`:/fed federated:latest
/fed/secure_workspace/central_server_name/startup/start.sh
```

Once the server is running you can start the clients:

```
docker run -it --network="host" --dns 10.41.41.1 -v
        --gpus all `pwd`:/fed federated:latest
/fed/secure_workspace/node1/startup/start.sh

docker run -it --network="host" --dns 10.41.41.1 -v
        --gpus all `pwd`:/fed federated:latest
/fed/secure_worckspace/node2/startup/start.sh
```

Once the connection between the server and the clients has been established, start the admin (username: admin_name)

```
docker run -it --network host --dns 10.41.41.1 -v
        `pwd`:/fed federated:latest
/fed/secure_workspace/admin@nameOfOrg.com/startup/fl_admin.sh
```

With the admin client command prompt successfully connected and logged in, enter the commands below in order.

```
upload_app application_name
```

Uploads the application from the admin client to the server's staging area.

```
set_run_number 1
```

Creates a run directory in the workspace for the run_number on the server and all clients. The run directory allows for the isolation of different runs so the information in one particular run does not interfere with other runs.

```
deploy_app application_name all
```

This will make your application the active one in the run_number workspace. After the above two commands, the server and all the clients know your application will reside in the run_1 workspace.

```
start_app all
```

This start_app command instructs the NVIDIA FLARE server and clients to start training.

Once the fl run is complete and the server has successfully aggregated the clients' results after all the rounds, run the following commands in the fl_admin to shutdown the system.

```
shutdown all
```

B.2.6 NOTES

- The above process is related to network configuration and independent of the FL application.
- The application folder should be located on the admin's site and there is no need to be sent to each participant.
- You should provide each client with a part of the dataset used to train your models.

Appendix C

Principal Threats and Mitigation Strategies in Federated Learning

C.1 Threats

C.1.1 Data poisoning attacks

In data poisoning attacks within FL systems, adversaries aim to manipulate the training data by introducing a mix of genuine and falsified data. These attacks are categorized into two types: clean-label and dirty-label. Clean-label attacks [243] involve altering the training data without changing the labels, as the data's classification is correctly verified. In contrast, dirty-label attacks [244], also known as backdoor attacks, involve tampering with both the data and its labels to induce model misclassification. An example of this is the label-flipping attack, where labels of legitimate training data are switched to incorrect classes. The impact of data poisoning on the final model in FL depends on the level of participant involvement in the attack and the volume of compromised data.

C.1.2 Model poisoning attacks

On the contrary, model-poisoning attacks in FL involve either partially or completely altering the model during the training phase. The aim here is to either corrupt the local model's weights before they are transmitted to the central server or to embed clandestine backdoors within the global model [245]. Studies [246] suggest that in FL systems, model-poisoning attacks tend to be more potent than data-poisoning attacks. This is primarily because the central server's aggregator, not being privy to the specifics of the local updates, lacks the capacity to identify irregularities or verify the accuracy of the updates it receives.

C.1.3 Model inversion attacks

Also known as member inference attacks, are designed to deduce the identities of the individuals in the training dataset [247]. These attacks investigate the feasibility of extracting specific training data details from a trained ML model. Such concerns are particularly pertinent in the context of legal compliance, such as when data storage limitations are in place or inter-institutional data exchange is restricted. As a result, model inversion has become a significant area of research in today's data-centric landscape. Beyond privacy concerns, model inversion offers insights into the workings of a ML model, helping to understand its decision-making process and the nature of information retained within the model. Model inversion can be applied to various predictive systems like linear regression, decision trees, neural networks, or any ML model that generates predictions from input data. Essentially, model inversion reverses the model's function: instead of making predictions from inputs, it reconstructs input data that the model most confidently associates with a certain target class. Since the model has learned the characteristics of its training data, it's assumed that the reconstructed data could reveal information about the original training inputs. In such attacks, the adversary has access to the trained model but not the actual training data.

The impact of data poisoning attacks in image classification tasks is investigated in [248] using the benchmark dataset MNIST and CIFAR-10. Specifically, this work studies the impact of data poisoning attacks on FL models regarding various percentages of malicious participants, random and targeted label flipping and the time of the attack. A higher percentage of malicious nodes results in higher degradation of the model performance. Moreover, a targeted data poisoning attack is detected more difficult. Finally, the time that an attack is performed is a crucial aspect, while a model that is trained with malicious nodes up to a point can converge if enough time is given. The work in [249] presents two variants of data poisoning attacks, namely model degradation and targeted label attacks. Both of those attacks are based on synthetic images generated by GANs. Through experiments, it is observed that the GAN-based attacks manage to fool common federated defences. Specifically, the model degradation attack provokes around 25% accuracy degradation, while the targeted label attack results in label misclassification of 56%. The authors also introduce a mechanism to mitigate these attacks, which is based on clean-label training on the server side. A distributed backdoor attack as a data poisoning attack on FL systems is proposed in [250]. A local trigger is chosen by each adversary instead of a common global one. During inference, attackers exploit the local triggers to form global ones. This work compares this kind of attack to a centralised approach, and they conclude that it is more persistent than the centralised scenarios.

C.2 Defence mechanisms

C.2.1 Data Anonymisation

Data anonymization, a crucial initial step in data protection, involves transforming data so it cannot be linked back to individuals. This process can be executed by either removing sensitive information or introducing random values into the dataset. Typically, input data contains two kinds of identifiers: direct and indirect. Direct identifiers are clear attributes like names, locations, and phone numbers that can directly point to an individual. In contrast, indirect identifiers consist of variable combinations (such as occupation and educational background) that can indirectly identify an individual through cross-referencing. Understanding the data's characteristics is key to implementing effective anonymization strategies. These strategies should aim to use relevant pseudonyms for identifiers, ensuring the anonymized data remains ethically sound, reusable, and in compliance with data protection regulations. Nevertheless, the degree of anonymization often varies across applications and largely depends on the level of consent provided by participants for specific anonymization procedures.

Recent literature has introduced various techniques for data privacy through anonymization before analysis. Yet, anonymization has its privacy limitations, as adversaries often access auxiliary information about individuals in the data. A notable example is the statistical de-anonymization of the Netflix Prize movie-ratings dataset [251], where authors demonstrated this by cross-referencing with publicly available ratings on IMDb. Additionally, video anonymization efforts, like blurring identifiable features [252], face challenges if features aren't fully concealed and non-participants view the videos, necessitating explicit participant consent. These instances highlight that maintaining privacy through anonymization is complex, especially considering the extent of information adversaries might already possess [253].

C.2.2 Differential Privacy

Differential Privacy (DP) is a methodology that integrates randomness into certain aspects of a system's operations to enhance privacy [254]. In the context of FL, this approach can be applied to the learning algorithm, although it's versatile enough to be used in various algorithmic contexts. The key idea behind incorporating randomness into a learning algorithm is to obscure any discernible patterns that might be tied to either the model with its parameters or the training data. Without this layer of randomness, adversaries could potentially deduce insights about the learning parameters needed for convergence or predict the likelihood of certain parameters being chosen by the algorithm for a given dataset. DP effectively mitigates these risks. Broadly, differential privacy can be categorized into Local Differential Privacy (LDP), Centralized Differential Privacy (CDP), also known as Global DP, and Distributed Differential Privacy (DDP).

LDP operates by adding noise to data at the local node level, eliminating the need for a trusted central authority. In this model, users' data privacy is protected because the noise is added before data reaches any central server. Conversely, CDP relies on a trusted central aggregator to add noise to the raw data received from users, ensuring that responses to queries on this data cannot be reverse-engineered to reveal private information. While CDP typically offers more accuracy due to the centralized noise addition, it requires users to trust the central curator. DDP combines the features of both LDP and CDP, enhancing client privacy without the need for a fully trusted server and providing better utility than LDP. In DDP, the noise distribution is contributed by multiple participants and relies on cryptographic protocols to maintain privacy, balancing the benefits of both localized and centralized approaches. Applying differential privacy in DL presents several challenges. There's a notable trade-off between model performance and data privacy, where adding more noise to enhance privacy can reduce model accuracy. Often, higher noise levels (lower values of ϵ) necessary for strong privacy protection lead to the need for more data and extended training time to achieve acceptable accuracy. Additionally, integrating differential privacy into DL algorithms is complex, and there's a lack of clear guidelines on selecting an appropriate privacy budget (ϵ). Furthermore, empirical experiments on information leakage provide only a lower bound estimate, as more potent attacks may exist beyond current methodologies. This means that the true extent of potential information leakage, especially in the context of membership inference attacks, is difficult to determine and may surpass what has been experimentally observed to date.

C.2.3 Homomorphic Encryption (HE)

Homomorphic encryption is a cryptographic method that allows for performing standard mathematical operations on encrypted data (ciphertext), such that when this data is decrypted, the outcome is the same as if the operations had been applied to the original, unencrypted data [255]. However, when dealing with complex mathematical functions within the ciphertext space, homomorphic encryption can encounter various limitations, particularly in terms of encryption performance. To address the limitations of standard Homomorphic Encryption (HE) in DL, various techniques have been developed that modify statistical methods to align with the properties of homomorphic computation and employ reasonable approximations where traditional methods are infeasible [256]. Research indicates that the use of standard HE in DL can significantly slow down training due to the time-intensive encryption and decryption processes. To address this, various methods have been developed to enhance HE's efficiency. Techniques like modifying neural network activation functions to simpler polynomial forms have been explored to facilitate HE. Some methods focus on creating scale-invariant and discretized neural networks, which allow for more effective HE usage, though they may impact accuracy [257]. Additionally, innovative frameworks combine HE with other cryptographic techniques and computational optimizations, showcasing the evolving landscape of efficient and secure HE implementations in neural network applications.

C.2.4 Secure Multiparty Computation (SMC)

Secure Multiparty Computation (SMC) is a cryptographic method allowing multiple parties to collaboratively compute a function while keeping their individual inputs and outputs private [258]. SMC is designed to operate in a secure environment that ensures zero-knowledge proofs, meaning it can verify the integrity of computations without revealing any actual data. While universally verifiable zero-knowledge proofs (ZKPs) play a crucial role in ensuring the integrity and appropriateness of encrypted data, they often involve complex calculations that can be inefficient.

In practice, some scenarios may permit the limited disclosure of information, provided certain security conditions are met. For example, SMC has been adapted for training ML models like linear regression and neural networks using techniques like stochastic gradient descent, under assumptions of semi-honest participation. Moreover, multi-party computation (MPC) protocols have been explored for encrypting sensitive attributes in ML model training, demonstrating the possibility of fair model training without exposing sensitive data. These advanced SMC and MPC approaches are designed to maintain privacy, even under various assumptions about participant behavior, including semi-honest or malicious actors. However, in real-world applications, algorithms often lean towards simplicity and practicality, balancing the need for accuracy with the complexity of the computations involved.

Hybrid methods that merge SMC with DP are emerging to address the limitations of each approach when dealing with potentially dishonest participants. Such methods aim to mitigate the inference risks associated with SMC and the accuracy issues caused by DP's noise injection. For instance, one approach combines SMC with DP and introduces a tunable trust parameter to handle various trust scenarios in FL environments [259]. These hybrid techniques typically involve innovative encryption and secure communication strategies to protect data privacy without significantly compromising model accuracy. They range from incorporating third-party entities for key management to utilizing blockchain technology for a fully decentralized learning process, offering a more robust defense against adversarial and inference attacks in FL systems.

Appendix D

Public 3D action recognition datasets

This appendix provides detailed information on the main public datasets currently available, which are pivotal for the development and evaluation of 3D action recognition algorithms. The following Table D.1 encapsulates key aspects of each dataset, including their size, variety of actions, annotation details, and unique characteristics that set them apart. This compilation aims to facilitate easy comparison and selection of appropriate datasets for specific research needs or application scenarios in the domain of 3D action recognition.

Table D.1: Public 3D action recognition datasets. *The authors in [1] used a capturing framework similar to the developed one (*i.e.* 3 Kinect sensors positioned in an arced configuration), but generated 80 non-identical views, by varying the height and the distance of the Kinect sensors from the subjects.

Datasets	Samples	Classes	Subjects	Views	Sensor	Modalities	Year
MSR-Action3D [260]	567	20	10	1	N/A	D+3DJoints	2010
CAD-60 [261]	60	12	4	-	Kinect v1	RGB+D+3DJoints	2011
RGBD-HuDaAct [262]	1189	13	30	1	Kinect v1	RGB+D	2011
MSRDailyActivity3D [263]	320	16	10	1	Kinect v1	RGB+D+3DJoints	2012
Act4 [264]	6844	14	24	4	Kinect v1	RGB+D	2012
Huawei/3DLife [265]	3740	22	17	5	Kinect v1	RGB+D+3DJoints	2013
CAD-120 [266]	120	10+10	4	-	Kinect v1	RGB+D+3DJoints	2013
3D Action Pairs [267]	360	12	10	1	Kinect v1	RGB+D+3DJoints	2013
Multiview 3D Event [268]	3815	8	8	3	Kinect v1	RGB+D+3DJoints	2013
Online RGB+D Action [269]	336	7	24	1	Kinect v1	RGB+D+3DJoints	2014
Northwestern-UCLA [270]	1475	10	10	3	Kinect v1	RGB+D+3DJoints	2014
UWA3D Multiview [271]	900	30	10	1	Kinect v1	RGB+D+3DJoints	2014
Office Activity [272]	1180	20	10	3	Kinect v1	RGB+D	2014
UTD-MHAD [273]	861	27	8	1	Kinect v1+WIS	RGB+D+3DJoints+ID	2015
UWA3D Multiview II [274]	1075	30	10	5	Kinect v1	RGB+D+3DJoints	2015
NTU RGB+D [1]	**56880**	**60**	40	**80***	Kinect v2	RGB+D+IR+3DJoints	2016
Formed dataset	25636	50	**132**	3	Kinect v2	RGB+D+3DFlow+3DJoints	2019

Appendix E

Dataset Preprocessing phase

E.1 Preprocessing steps

In the preprocessing phase of our dataset analysis, which aims to extract varied modalities such as skeleton, RGB, depth, and 3D flow, we adhere to a structured sequence of steps. This process begins with Decoding, involving the transformation of compressed image or video files into a more analyzable format. Subsequently, we undertake Mapping Depth to RGB, a crucial step that aligns depth data from sensors with RGB imagery to create an integrated, multi-dimensional dataset. Following this, Human Silhouette Estimation is performed to detect and isolate the human figure within these composite images. This isolation paves the way for Skeleton Representation, where we construct an intricate skeletal model that accurately portrays the subject's posture and movements. To ensure the subject remains the focal point of our analysis, Background Removal is employed, leveraging depth data to effectively distinguish the subject from their background. The next step, Cropping, is applied to maintain a consistent and focused view of the subject throughout the dataset. The final step in our preprocessing protocol is 3D Flow, where we analyze the subject's movements in a three-dimensional space, thus providing a comprehensive and multi-faceted understanding of the dataset.

Decoding: This is the process of converting data from one format to another. In the context of image or video processing, this often refers to the conversion of compressed image or video files into a format that can be easily processed. For example, decoding a video file into a series of individual frames.

Mapping Depth to RGB: In systems that use both depth sensors and RGB cameras, this step involves aligning the depth information (which indicates how far objects are from the sensor) with the RGB image. This is crucial because depth sensors and RGB cameras have different perspectives and fields of view. The mapping process aligns these two data sources so that the depth information corresponds accurately to the RGB image, allowing for more sophisticated analyses, like 3D reconstructions or enhanced object detection.

Human Silhouette Estimation: This step is focused on identifying the outline or contour of a human figure within an image or video frame. This is often done using

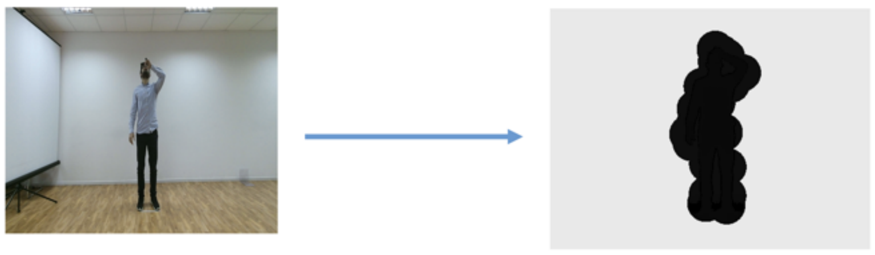


Figure E.1: Human silhouette: Map 3D joint coordinates to 2D space. Estimate human silhouette based on skeleton data.

techniques like background subtraction, edge detection, or DL models trained to recognize human shapes. The goal is to isolate the figure of the human from the rest of the image (as illustrated in Figure E.1. This step focuses on identifying the human figure within the combined depth and RGB data. The depth information can significantly enhance the accuracy of silhouette estimation compared to using RGB data alone.

Skeleton Representation: Once the human silhouette is estimated, the next step is often to create a skeleton representation. This involves identifying key points of the body (like joints) and connecting them in a way that represents the human skeleton. This is crucial for analyzing human posture, movement, and activities. Depth data can greatly enhance the accuracy and dimensionality of this model.

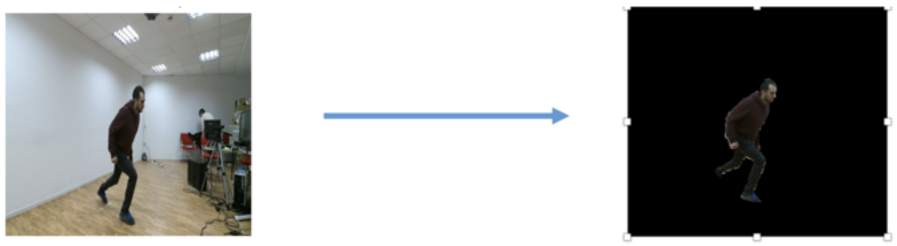


Figure E.2: RGB information: Map Depth to RGB pixels, and use their colour value to create RGB image in Depths' resolution. Remove background based on depth values around skeleton data.

Background Removal: This process involves separating the foreground (usually the main subject, like a human figure) from the background (as depicted in Figure E.2). This can be done using various techniques, such as chroma keying (green screen techniques), statistical methods (like Gaussian Mixture Models), or DL approaches. The combination of depth and RGB data makes background removal more effective. Objects can be more easily differentiated from the background based on their distance from the sensor, not just their color or brightness.

Cropping: Cropping is a simple yet essential step where irrelevant parts of the image or frame are cut out to focus more on the subject of interest. In motion analysis, cropping

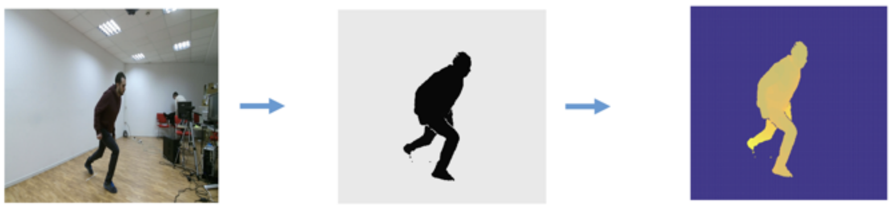


Figure E.3: Depth map based on skeleton representation: Using depth pixels in previous 2d estimation, find the corresponding 3d voxels based on 3d skeleton data).

is often used to center the human figure or to maintain a consistent size and position of the subject across frames. Cropping can be more accurately achieved by using depth data to maintain consistent focus on the subject across different frames (as presented in Figure E.3).

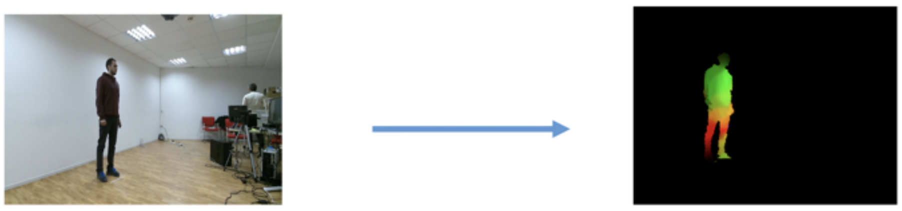


Figure E.4: 3D flow: estimates the actual motion field of the action.

3D Flow: This term refers to the estimation of three-dimensional movement within the scene. In the context of human motion analysis, it involves tracking the movement of the body or limbs in three dimensions over time (as illustrated in Figure E.4). This is typically achieved through advanced computer vision techniques and involves the use of depth sensors or stereo cameras to capture 3D data (dense real-time 3Dflow for RGB-D cameras). The algorithm is implemented on a GPU to achieve real-time performance.

E.1.1 Dataset Samples

The subsequent images showcase a variety of actions and modalities extracted from the dataset. These visual representations highlight the diverse range of activities and the different data modalities that have been derived through our comprehensive analysis process.

E.2 Ablation studies

Skeleton-tracking or 3D flow features? With respect to the type of information that is used for realizing 3D action recognition, it was experimentally shown that 3D flow information exhibited greater potential towards leading to increased action recognition performance, compared to the cases of skeleton-tracking and depth data. To provide

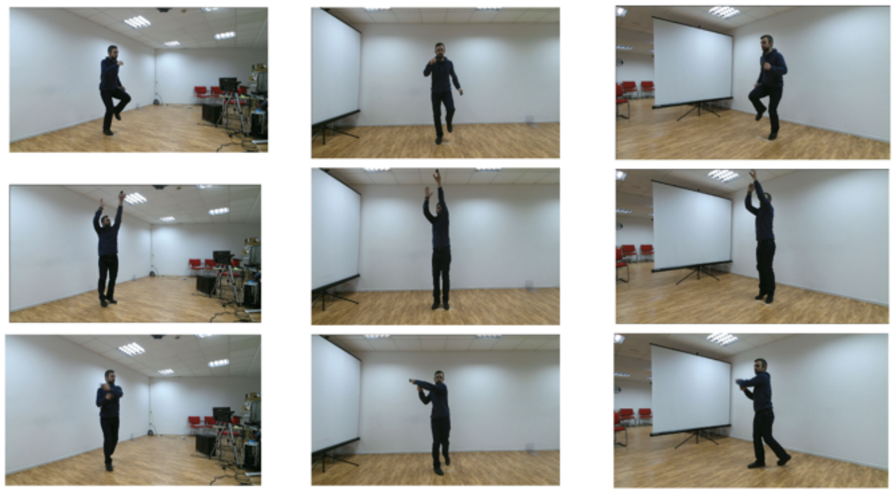
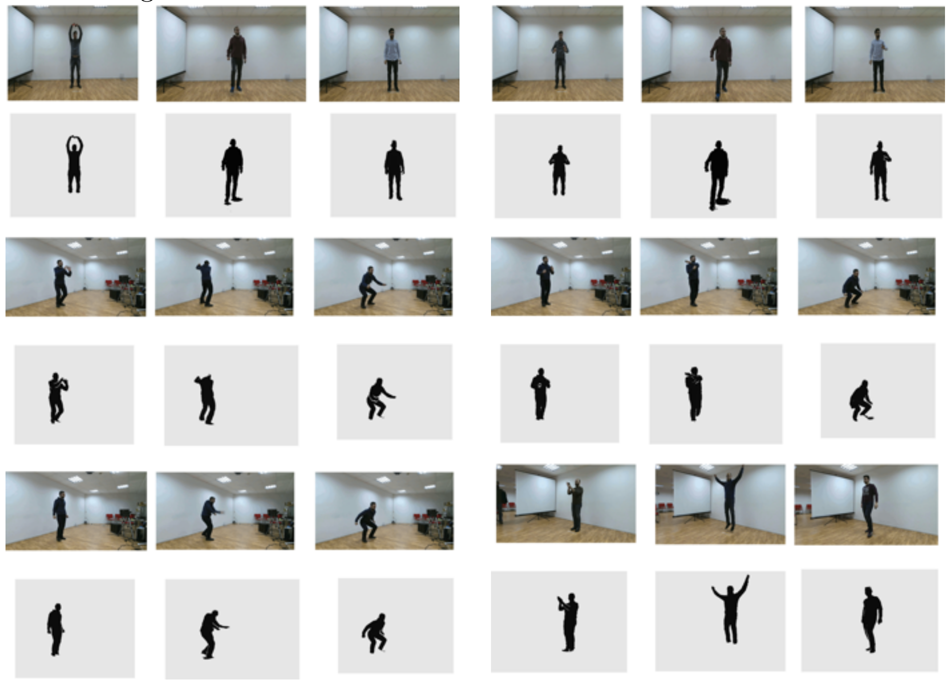


Figure E.5: RGB data captured from three distinct perspectives while the subject is engaged in performing three different actions.

better insight, the action recognition confusion matrix obtained from the proposed 3D flow and skeleton information is given in Figure E.3, respectively. In particular, experiments demonstrate that 3D flow features are advantageous for a wide set of actions, including arm-related movements (*e.g.* 'Hand waving', 'Pointing to something with finger', *etc.*) as well as more extensive whole-body movements (like 'Jump up' and 'Standing up') in both D1 and D2. On the other hand, skeleton-tracking approaches, which make extensive use of domain-specific knowledge, are shown to be advantageous when the performed action involves fine-grained motions (*e.g.* 'Reading', 'Writing', *etc.*). Also, in the case of slow-performing actions, skeleton-tracking methods seem to perform better compared to flow-based ones, taking advantage of the ability of the employed skeleton-tracker to detect the precise position of the human joints. Overall, flow-based methods exhibit increased potential for reaching improved recognition rates, as long as suitable and robust representations of the high-dimensional and often noisy 3D flow signal are efficiently computed. On the other hand, skeleton-tracking-based methods currently constitute the most popular category, mainly due to implementation simplicity (*i.e.* use of skeleton joint features); however, suffering from the limitations of the employed skeleton-tracker (*e.g.* presence of noise, failures of accurate detection of human joints, particular requirements for human posture, *etc.*).

Complementarity of modalities. Skeleton trackers are prone to noise and self-occlusions, while it is well-known that side-view skeletal data are less accurate than the front-view ones. Similarly, the efficiency of surface features is also significantly affected by varying capturing perspectives. Additionally, flow features are generally affected by the quality of the capturing settings (*e.g.* illumination conditions, distance from the capturing device, *etc.*). To this end, it is evident that either modality alone can be ambiguous in certain cases. However, when features from multiple modalities are combined, action

Table E.1: RGB and associated depth data samples obtained from various perspectives of a subject executing diverse actions.



recognition results are experimentally shown to be significantly improved. A critical question that needs to be answered though is the level at which multi-modal information should be fused. According to the current experimental evaluation, the late fusion scheme leads to the highest performance improvement [second (b) group of experiments in Tables 5.1 and 5.2, demonstrating the relatively increased complementarity of the high-order features. Summarizing all the above observations, the fundamental consideration of the current work that a truly robust system in the general case is necessary to efficiently and adaptively combine multiple information sources (surface, flow, skeleton-tracking) is also experimentally verified.

Table E.2: 3D flow fields representing the execution of four distinct actions (jumping, weight lifting, basketball shooting, and golf swinging) captured at four sequential time points, spanning from the initiation to the completion of each task.

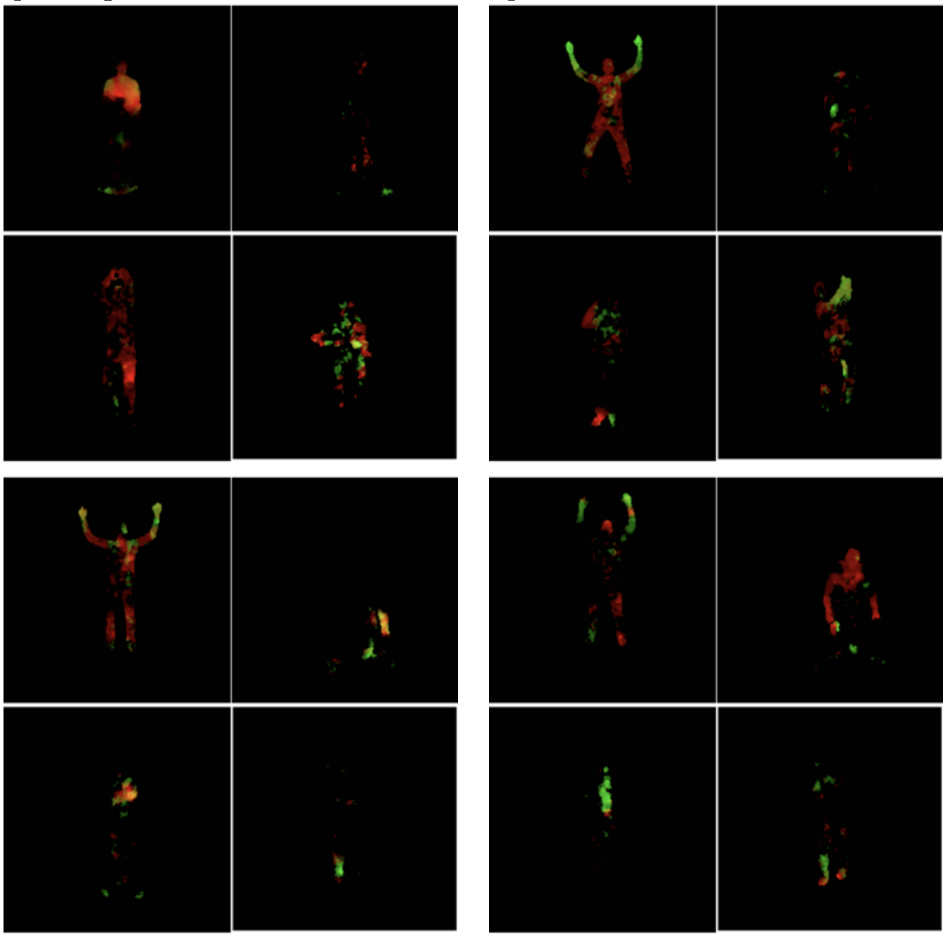
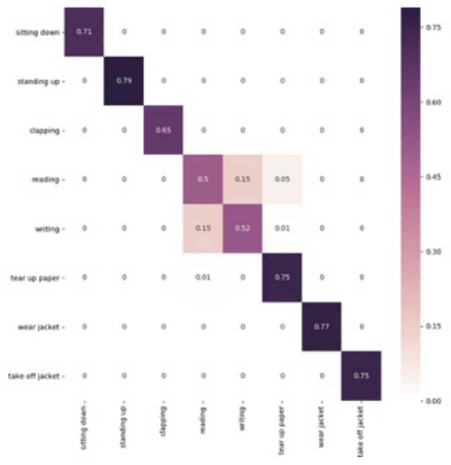
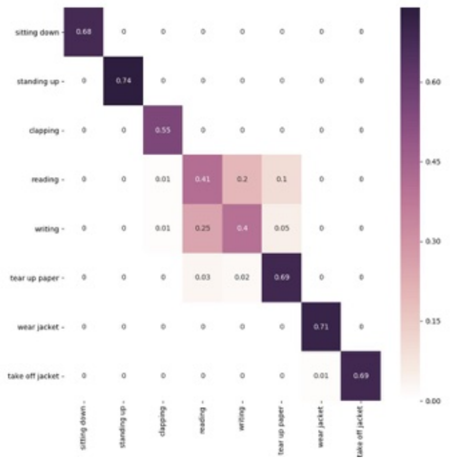


Table E.3: Confusion matrices derived from the implemented 3D flow and skeleton tracking methodologies.



	sitting down	standing up	clapping	reading	writing	tear up paper	wear jacket	take off jacket
sitting down	0.71	0	0	0	0	0	0	0
standing up	0	0.79	0	0	0	0	0	0
clapping	0	0	0.65	0	0	0	0	0
reading	0	0	0	0.5	0.15	0.05	0	0
writing	0	0	0	0.15	0.52	0.01	0	0
tear up paper	0	0	0	0.01	0	0.75	0	0
wear jacket	0	0	0	0	0	0	0.77	0
take off jacket	0	0	0	0	0	0	0	0.75



	sitting down	standing up	clapping	reading	writing	tear up paper	wear jacket	take off jacket
sitting down	0.68	0	0	0	0	0	0	0
standing up	0	0.74	0	0	0	0	0	0
clapping	0	0	0.55	0	0	0	0	0
reading	0	0	0.01	0.41	0.2	0.1	0	0
writing	0	0	0.01	0.25	0.4	0.05	0	0
tear up paper	0	0	0	0.03	0.02	0.69	0	0
wear jacket	0	0	0	0	0	0	0.71	0
take off jacket	0	0	0	0	0	0	0.01	0.69

Appendix F

Architecture Variability Challenges

F.1 Challenges in Identification and Excitation of Features

The core of the excitation technique is grounded in the observation that the most influential features are highlighted by the gradients during the backpropagation process. This approach begins with initial training at each local node using its own private dataset, a critical phase for developing accurate data representations. These representations lay the groundwork for the method's subsequent phases. Following this initial training, every node processes images from a shared dataset. For each image, the algorithm pinpoints the top features exhibiting the largest absolute gradients within the last block of the network' feature map. It is these prominent features, considered pivotal for the image's representation, that are then forwarded to the central server. This process ensures that the most critical aspects of the data are emphasized and aggregated globally, enhancing the overall learning and representation capability of the system. The proposed approach, involving the excitation of specific features based on gradients, proved to be extremely demanding in terms of both RAM and VRAM requirements, rendering it less suitable for immediate application, but essential for a thorough grasp of the research direction of the study. Addressing the challenges posed by the high resource demands of feature extraction with gradients in FL, a practical workaround has been developed. To overcome these limitations, an alternative method has been adopted. This method involves pooling the maximum values of the last feature map, a process that significantly reduces the computational load and memory requirements. After pooling, a random crop of 1920 features is selected from these pooled features (as depicted in Figure F.1). This approach maintains the essence of capturing the most significant aspects of the data while being much less resource-intensive. The training process then employs a similarity loss function on these 1920 features. The representation learning framework is applied between the globally aggregated representation and the current representation from the node undergoing training. This method ensures that the essential features are still being emphasized during the model training, but with a significantly reduced computational burden. The intuition behind performing this pooling and random cropping technique at the feature space is to enhance the model's focus while introducing variability and robustness in the learning process. By applying these operations in the feature space, the model is encouraged to learn representations that are invariant to changes in the spatial arrangement of features within the images.

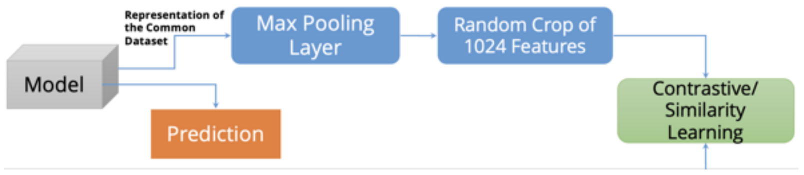

Representation of the Common Dataset
Max Pooling Layer
Random Crop of 1024 Features
Model
Contrastive/ Similarity Learning
Prediction

Figure F.1: Enhancing feature salience for similarity learning through pooling and cropping operations.

F.2 Enhanced Feature Extraction: Integrating Multi-Level Network Insights

The scope of the excitation process was expanded to include feature extraction not only from the final layer but also from blocks throughout the network. This extension allows for a more comprehensive and nuanced understanding of the image by incorporating a wider range of feature complexities and insights from across the network's architecture. As stated above, the process of feature elicitation serves as a pivotal step in harmonizing the local models with a central, aggregated global accumulation of the descriptors. It starts with an image passing through a neural network, which is segmented into distinct blocks, each responsible for extracting features at varying levels of complexity. The early blocks, such as Block1, typically capture elementary features like edges, while the deeper blocks, such as Block4, discern more intricate patterns. The outputs of these blocks undergo a selective process where a specific number of features are chosen based on their activation levels, which are indicative of their importance. These chosen features are then combined, with the deeper blocks contributing a larger share of features, reflecting their increased complexity and importance in characterizing the image. This process is illustrated in Figure F.2. The reasoning behind this specific excitation approach is grounded in empirical findings. Experiments have revealed that similarity metrics for features in the initial layers of both trained and untrained models are remarkably high, indicating homogeneity in the primitive features extracted by these layers. Consequently, as the network delves deeper, the need for capturing a broader and more complex range of features grows, prompting the selection of larger feature maps from the deeper layers—(128, 256, 512, 1024) respectively—to ensure a comprehensive representation.

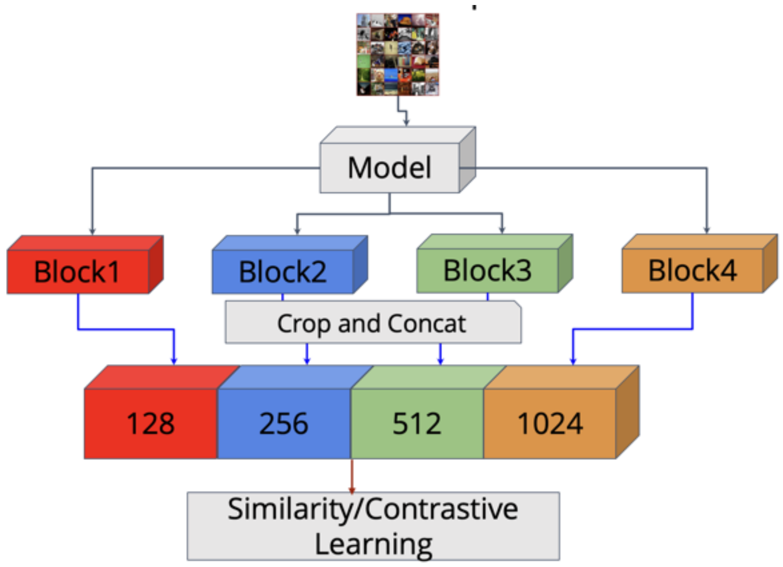


Figure F.2: Schematic overview of the enhanced feature excitation process, illustrating Multi-Level feature extraction across the NN.

F.3 Model Agnostic Block selection

This section outlines the methods used to align individual blocks of different network architectures. To promote the alignment of blocks from different model architectures, cosine similarity serves as the primary criterion, with the objective of minimizing the distances between representations within the framework's latent space with respect to angles. Consequently, the rationale behind the selection of specific architectural blocks is predicated upon the angular congruence of the feature maps with the canonical basis I^N. This methodology underscores the significance of angular relationships in the dimensional space, aiming to enhance the cohesion and compatibility of representations derived from various architectures. The divergence between each architectural block is carefully analyzed, and the difference between each block's angle distributions is measured using the Kullback-Leibler (KL) Divergence. Because of the difference in the total number of parameters in each feature map of the blocks, a probability distribution was created by grouping the angles of the features of each block into clusters. The assessment of KL-Divergence is performed over a comprehensive spectrum of cluster counts, spanning from 10 to $100,000$, to accommodate a diverse array of feature distributions. For simplicity reasons, the following tables display information for the $100,000$ because they create meticulous distributions. Additionally, the values change proportionally with the varying number of clusters. Given the inherent asymmetry of KL-Divergence, a bidirectional analysis is required for the block alignment selection process. In particular, the divergence between block A and block B, as well as the opposite, is assessed. The blocks that show the least amount of KL-Divergence, fulfil the following criterion:

$$\min\left(\text{Dist}(\text{Min}(\text{KL}(Block_A \| Block_B)), \text{Min}(\text{KL}(Block_B \| Block_A)))\right)$$

are prioritized for selection, underpinning the methodology's rigor in finding blocks that are best aligned with the latent space in terms of feature distribution similarities is this dual-sided evaluation, which guarantees a more robust and balanced selection criterion. To obtain precise and representative embeddings from the models while adhering to the principles of FL, each network undergoes training on a shared subset within the framework. The KL-Divergences across the angular distributions of each network are evaluated using randomly selected images from the test set of each dataset. The total number of the images selected is 3000.

F.3.1 CIFAR-10

The Tables listed below contain information on the KL-Divergence between different blocks of different networks regarding CIFAR-10[123] dataset. Each cell includes the divergence of the block of the column to the block of the row (Eg. $KL(EfficientNet_{Block1} || MobileNetV3_{Block1}) = 0.001835$).

Models		EfficientNet							MobileNetV3						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
EfficientNet	Block 1								**0.005879**	0.012651	0.070842	0.215922	0.140473	0.392318	0.044772
	Block 2								0.030818	**0.006274**	0.045828	0.179387	0.106278	0.354205	0.027828
	Block 3								0.281583	0.075217	**0.009656**	0.069355	0.031615	0.21361	0.034167
	Block 4								1.063728	0.390618	0.111132	**0.006046**	0.029001	0.06836	0.208333
	Block5								0.598605	0.199446	0.036188	0.024056	**0.010301**	0.127705	0.105826
	Block 6								2.112472	0.825165	0.371872	0.07075	0.164182	**0.012829**	0.419214
	Block 7								1.033316	0.360404	0.109932	0.016545	0.03084	0.071263	**0.01631**
MobileNetV3	Block 1	**0.000658**	0.008668	0.069162	0.215893	0.134941	0.364148	0.195752							
	Block 2	0.016597	**0.000375**	0.038273	0.170294	0.095326	0.312184	0.151957							
	Block 3	0.2037	0.075025	**0.001118**	0.068833	0.021778	0.181276	0.059095							
	Block 4	0.786691	0.410465	0.094625	**0.002396**	0.022825	0.043285	0.007566							
	Block5	0.454722	0.207749	0.027223	0.017547	**0.0008**	0.097498	0.01514							
	Block 6	1.670294	1.060446	0.453854	0.119542	0.255016	**0.011901**	0.096735							
	Block 7	0.142643	0.077629	0.049577	0.131956	0.08263	0.238474	**0.001073**							

Table F.1: The KL-Divergance of EfficientNet and MobileNetV3 under 100000 of total clusters on CIFAR-10 dataset.

Table F.1 suggests a minimal divergence between EfficientNet[195] and MobileNetV3[196] within identical blocks. The reason behind that observation is that both use inverted residual blocks as their main blocks, in combination with the same activation function GeLU[275], the distribution of both the features and the angles of the features are very similar and as a result the KL-Divergence is relatively similar. In this regard, the definitive factor for the selection of the blocks to be aligned is the divergence of the blocks of ResNet with EfficientNet and MobileNetV3.

Because of the similarities between the architectures of EfficientNet and MobileNetV3 the blocks with the least divergence with the ResNet are the same. The first blocks selected for alignment are Block 1 of ResNet to Block 1 from EfficientNet and MobileNetV3 considering that in the early layers, the models capture foundational features. Through

Models		ResNet				EfficientNet						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					0.00068	0.009418	0.070512	0.217583	0.136534	0.366056	0.197521
	Block 2					0.008634	0.001059	0.046743	0.184004	0.106979	0.328151	0.165089
	Block 3					0.129047	0.039654	0.007132	0.093103	0.037369	0.214194	0.079328
	Block 4					0.269671	0.178571	0.102454	0.14866	0.120816	0.220349	0.116449
EfficientNet	Block 1	0.000844	0.004542	0.044459	0.05661							
	Block 2	0.022239	0.000935	0.022616	0.036883							
	Block 3	0.263621	0.093574	0.007314	0.029765							
	Block 4	0.972336	0.45886	0.138387	0.164514							
	Block5	0.563572	0.241052	0.048686	0.079718							
	Block 6	1.823545	0.944402	0.371399	0.334205							
	Block 7	0.914604	0.424828	0.119697	0.125557							

Table F.2: The KL-Divergence of ResNet and EfficientNet with 100000 total clusters on CIFAR-10 dataset

Models		ResNet				EfficientNet						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					0.000521	0.010571	0.069895	0.216046	0.136486	0.394767	0.03977
	Block 2					0.017129	0.001193	0.046293	0.182598	0.106987	0.3567	0.02268
	Block 3					0.190005	0.033726	0.008253	0.092526	0.037184	0.241281	0.012748
	Block 4					0.493389	0.148775	0.132785	0.156051	0.12151	0.22951	0.008341
EfficientNet	Block 1	0.000305	0.005873	0.047327	0.059397							
	Block 2	0.026366	0.001416	0.020607	0.035027							
	Block 3	0.261355	0.092127	0.007181	0.030475							
	Block 4	0.969529	0.456105	0.138102	0.163456							
	Block5	0.566486	0.240318	0.048223	0.078827							
	Block 6	2.097124	1.127375	0.452288	0.355203							
	Block 7	0.203899	0.084832	0.021394	0.005267							

Table F.3: The KL-Divergence of ResNet and MobileNetV3 with 100000 total clusters on CIFAR-10 dataset

the next layers the models compress the information capturing more complex patterns, so as a result Block 2 of ResNet to Block 2, Block 3 of ResNet to Block 6 from EfficientNet and MobileNetV3, are selected for alignment. Lastly in the latest stages of the network, high-level features are created and consequently, the last blocks of the network are chosen to be aligned as well. Tables F.2 and F.3 depict the relationships of the divergence between the models.

F.3.2 CIFAR-100

Respectively, similar blocks are selected for the experiments on CIFAR-100 [123]. Table F.4 demonstrates that the deviations between the blocks of MobileNetV3 and EfficientNet are minimal, leaving ResNet to be the decisive factor for the alignment process. Tables F.5 and F.6 present that, Block 1 is chosen from all the models correspondingly, again for the reason that the networks in the early stages are creating primitive features. Afterwards features from Block 2 and 3 of ResNet are aligned with Block 3 and 6 of EfficientNet and MobileNetV3, and the last blocks of EfficientNet and MobileNetV3 with Block 4 of ResNet.

Models		EfficientNet							MobileNetV3						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
EfficientNet	Block 1								**0.218016**	0.225205	0.279187	0.301317	0.35913	0.632259	3.439775
	Block 2								0.646587	**0.113911**	0.205709	0.608357	1.019514	1.460066	4.859446
	Block 3								0.523589	0.152365	**0.123351**	0.318149	0.549371	0.9263	4.155667
	Block 4								0.500756	0.213124	0.237336	**0.190708**	0.382663	0.711786	3.79824
	Block5								0.413825	0.244891	0.27113	0.209605	**0.219905**	0.490568	3.35256
	Block 6								0.830855	0.627157	0.655093	0.683079	0.553999	**0.484121**	3.23499
	Block 7								0.350184	0.854192	0.907638	0.671597	0.410047	0.503168	**0.310968**
MobileNetV3	Block 1	**0.315756**	0.481272	0.501972	0.686995	0.669719	1.943845	4.585647							
	Block 2	0.407339	**0.184537**	0.612249	1.181815	1.316692	2.034639	4.595812							
	Block 3	0.564628	0.585293	**0.531754**	1.127809	1.275242	2.079318	4.836214							
	Block 4	0.317165	0.264614	0.331715	**0.193839**	0.552991	1.087678	3.327951							
	Block5	0.410864	0.227647	0.258031	0.215528	**0.185277**	1.423999	2.804323							
	Block 6	0.206348	0.282273	0.273307	0.400506	0.391776	**0.202472**	2.241201							
	Block 7	3.942595	4.402738	4.219353	4.153589	4.014203	3.995513	**2.143859**							

Table F.4: The KL-Divergance of EfficientNet and MobileNetV3 under 100000 of total clusters on CIFAR-100 dataset.

Models		ResNet				EfficientNet						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					**1.527542**	2.93446	1.77745	1.727654	1.618638	1.640933	2.579126
	Block 2					2.240503	2.63346	**1.724869**	2.394215	2.275536	2.266125	2.846146
	Block 3					2.794784	3.213168	3.034002	2.964275	2.841339	**2.219722**	2.848801
	Block 4					5.070035	5.542254	5.351492	5.277848	5.136278	5.111487	**2.31144**
EfficientNet	Block 1	**0.750634**	1.694717	2.444256	5.633593							
	Block 2	1.961719	3.157287	4.121835	7.273275							
	Block 3	1.361294	**1.328863**	3.290359	6.600445							
	Block 4	1.095784	2.003874	2.855458	6.236753							
	Block5	0.927186	1.567833	2.322489	5.697388							
	Block 6	0.804173	1.470818	**2.171293**	5.482268							
	Block 7	1.561258	1.643336	2.817603	**2.739593**							

Table F.5: KL-Divergence of ResNet and EfficientNet under 100000 of total clusters on CIFAR-100 dataset

Models		ResNet				MobileNetV3						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					**1.527587**	1.987513	2.042162	1.820525	1.582829	1.69002	1.199633
	Block 2					2.787722	2.69214	**2.245111**	2.502557	2.331263	2.315515	2.747835
	Block 3					2.717752	3.275182	3.330914	3.076456	2.78985	**2.262766**	2.548909
	Block 4					5.03948	5.604024	5.665168	5.396529	5.086102	5.14348	**0.25019**
MobileNetV3	Block 1	**1.052453**	1.582353	1.967604	4.227482							
	Block 2	2.323507	3.564837	4.535146	7.637055							
	Block 3	2.407682	**2.252267**	4.772423	7.868003							
	Block 4	1.44966	2.535469	3.479917	6.869772							
	Block5	1.748393	1.439395	2.162265	5.571014							
	Block 6	1.936717	1.80364	**1.551285**	5.852958							
	Block 7	1.772577	0.580721	0.595913	**0.574538**							

Table F.6: KL-Divergence of ResNet and MobileNetV3 under 100000 of total clusters on CIFAR-100 dataset

F.3.3 MNIST

For the experiments conducted on MNIST, comparable segments were utilized. The data presented in Table F.7 reveals that the variations between the blocks of MobileNetV3 and EfficientNet are negligible, making ResNet the critical element in the selection procedure. The findings in Tables F.8 and F.9 indicate that Block 1 was selected across all models due to the networks' initial phase of generating fundamental features.

Subsequently, features from Blocks 2 and 3 in ResNet were matched with Blocks 5 and 6 in EfficientNet and MobileNetV3, while the terminal blocks of EfficientNet and MobileNetV3 were aligned with Block 4 of ResNet.

Models		EfficientNet							MobileNetv3						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
EfficientNet	Block 1								**0.375764**	0.225656	0.284473	0.306678	0.363175	0.630568	3.429651
	Block 2								0.657979	**0.111261**	0.204987	0.609633	1.024846	1.459824	4.860414
	Block 3								0.536652	0.154311	**0.180737**	0.320509	0.557993	0.92878	4.156884
	Block 4								0.508262	0.214294	0.239394	**0.257865**	0.381126	0.714614	3.785969
	Block5								0.427617	0.245588	0.271616	0.214523	**0.225214**	0.488582	3.346678
	Block 6								0.442059	0.326228	0.355437	0.282869	0.254402	**0.493589**	3.215776
	Block 7								0.319523	0.849829	0.903803	0.668827	0.410229	0.499954	**0.121892**
MobileNetv3	Block 1	**0.325667**	0.488054	0.511519	0.694075	0.673376	0.956554	1.596348							
	Block 2	0.405954	**0.133582**	0.606061	1.18025	1.310485	2.04354	4.622935							
	Block 3	0.622085	0.583395	**0.52985**	1.122583	1.272181	2.085007	4.864075							
	Block 4	0.719485	0.569809	0.73605	**0.498195**	0.558976	1.090538	3.348655							
	Block5	0.413348	0.225154	0.860232	0.675015	**0.184865**	1.421698	1.815149							
	Block 6	0.807721	0.927952	1.272558	0.863452	0.891632	**0.704155**	2.254381							
	Block 7	3.953618	4.406078	4.223471	4.149524	4.008356	3.976949	**1.248096**							

Table F.7: KL-Divergence of EfficientNet and MobileNetV3 under 100000 of total clusters on MNIST dataset

Models		ResNet				EfficientNet						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					**1.524062**	1.929616	1.774342	1.718894	2.611252	2.622048	1.576289
	Block 2					2.343066	2.633757	2.463694	2.394114	**2.275392**	2.752528	2.842223
	Block 3					2.802145	3.219258	3.036476	2.965269	2.834455	**2.105753**	2.649185
	Block 4					5.067357	5.532251	5.341099	5.26471	5.11854	5.074895	**2.263853**
EfficientNet	Block 1	**0.746333**	1.696947	2.427737	5.571439							
	Block 2	1.968233	3.17052	4.115116	7.229506							
	Block 3	1.366197	2.406117	3.292945	6.569581							
	Block 4	1.09441	2.009629	2.838689	6.189712							
	Block5	1.823052	**1.168775**	2.308678	5.644603							
	Block 6	0.802034	1.471349	**0.156008**	5.42015							
	Block 7	0.953877	2.639661	0.813958	**0.711726**							

Table F.8: KL-Divergence of ResNet and EfficientNet under 100000 of total clusters on MNIST dataset

Models		ResNet				MobileNetV3						
	Blocks	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block5	Block 6	Block 7
ResNet	Block 1					**1.525567**	1.983046	2.03604	1.815649	1.576418	1.685058	2.203933
	Block 2					2.681143	2.688445	2.747319	2.506776	**2.227699**	2.309004	2.745187
	Block 3					2.711325	3.265555	3.325846	3.077801	2.785244	**2.264283**	2.547512
	Block 4					5.009112	5.582396	5.647033	5.38646	5.066082	5.127199	**0.249085**
MobileNetV3	Block 1	**1.050996**	1.589903	2.971241	4.203535							
	Block 2	2.319361	3.573894	4.526082	7.590049							
	Block 3	2.402607	3.759075	4.763455	7.828729							
	Block 4	1.452412	2.541946	3.466919	6.82742							
	Block5	1.750935	**1.441558**	2.149365	5.522265							
	Block 6	1.030959	1.806461	**2.114404**	5.802558							
	Block 7	1.772484	1.577201	2.499037	**0.576745**							

Table F.9: KL-Divergence of ResNet and MobileNetV3 under 100000 of total clusters on MNIST dataset