Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (37^{th}cycle)

Addressing Heterogeneity in Federated Learning for Real-world Vision Applications

By

Eros Fanì
******

Supervisors:
Prof. Barbara Caputo
Dr. Marco Ciccone

Doctoral Examination Committee: (in alphabetical order)
Prof. Lorenzo Cavallaro, University College London (UCL), *Referee*
Prof. Carlo Ciliberto, University College London (UCL), *Referee*
Prof. Sophie Fosson, Politecnico di Torino (PoliTo), *President*
Prof. Sandra Pieraccini, Politecnico di Torino (PoliTo)
Prof. Novi Quadrianto, University of Sussex (UoS)

Politecnico di Torino
2025

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Eros Fanì
2025

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Abstract

Federated Learning (FL) has emerged as a promising approach for training machine learning models using decentralized data sources while maintaining users' privacy. Unlike traditional centralized learning, FL allows users to train models collaboratively without needing to collect their raw data on a central server, making FL particularly suitable for privacy-sensitive applications such as healthcare, finance, and autonomous systems. However, FL faces several challenges, such as statistical heterogeneity, which arises in the presence of variations in data distributions across clients. In computer vision applications, factors such as class and domain imbalance are primary sources of heterogeneity, leading to notable performance degradation and instability during federated training.

This manuscript addresses the challenges of statistical heterogeneity in FL, particularly in relation to computer vision tasks. First, it introduces a novel FL algorithm, Federated Recursive Ridge Regression (`Fed3R`), designed to mitigate the effects of heterogeneity. `Fed3R` constructs a ridge regression-based classifier equivalent to the centralized ridge regression solution. The parameters obtained from `Fed3R` can be used to initialize the classifier for further fine-tuning with other FL algorithms, significantly improving their efficiency and robustness. Then, the importance of initialization in Personalized Federated Learning (PFL) is explored, demonstrating how well-initialized classifiers can substantially improve the performance and efficiency of personalized models. Furthermore, a new method is introduced, Only Local Labels (`OLL`), which enhances local personalization by selectively filtering classifier neurons.

Additionally, FL is studied in the context of autonomous driving for the semantic segmentation task. A new benchmark, `FedDrive`, is introduced to analyze the impact of domain shift and class imbalance in this scenario. Furthermore, a new FL problem, Federated Source-Free Domain Adaptation `FFreeDA`, is proposed,

addressing scenarios where clients lack labeled data. Finally, a novel algorithm, Learning Across Domains and Devices (`LADD`), is designed to address (`FFreeDA`), leveraging semi-supervised learning and clustered FL techniques.

Extensive experimental evaluations validate the proposed methods, highlighting their effectiveness in addressing statistical heterogeneity in FL. By advancing the understanding and practical methodologies for FL for computer vision applications, this manuscript contributes to the broader goal of making FL more robust, scalable, and applicable to real-world scenarios. The findings provide insights into the interplay between FL algorithms, statistical heterogeneity, and computer vision tasks, paving the way for future research in this rapidly evolving field.

Contents

Chapter 1

Introduction

Artificial Intelligence (AI) [1] is becoming essential to modern technology, powering applications across various fields, including healthcare [2], finance [3], and autonomous systems [4]. One of the most significant advancements in AI is deep learning [5], which has transformed areas such as natural language processing [6] and computer vision [7]. In particular, computer vision has dramatically benefited from large-scale deep learning models, enabling its application to medical image analysis [8], self-driving cars [9], industrial defect detection [10], and facial recognition [11], among others. These breakthroughs have been made possible by the availability of massive datasets and powerful computational resources, allowing for the centralized training of high-capacity neural networks.

Traditional deep learning typically relies on large, centralized datasets, where a massive amount of data is collected and stored on a central server for model training. However, in today's digital landscape, smartphone users, wearable devices, Internet of Things (IoT) sensors, and other distributed sources generate a significant portion of data at the edge. For example, mobile phones produce vast portions of images and text, smartwatches continuously monitor health metrics, and autonomous vehicles gather real-time sensor data. To use this data for training, it would need to be transferred to a central server, which raises serious privacy concerns, as sensitive user information could be exposed or misused. Additionally, regulations such as the General Data Protection Regulation (GDPR) [12] impose strict limitations on data sharing.

These challenges have led to the development of Federated Learning (FL) [13] as a privacy-enhancing solution. FL enables collaborative model training using decentralized data sources without requiring the raw data to leave local devices. FL allows individual devices or institutions to train a global model without the necessity to collect data from edge devices. Instead, only the model updates, such as gradients or parameters, are shared with the central server. The server then aggregates these updates to improve the global model. This approach enhances privacy and reduces the need for expensive data transfers and centralized storage.

FL expands the traditional Distributed Learning (DL) [14] setting, where multiple computational nodes work together to train a shared model, differing from it in some key aspects. Indeed, while traditional DL approaches typically assume uniform data distribution and stable network connectivity, FL operates in a more constrained environment. In FL, clients, such as mobile devices, edge sensors, or institutions, often possess highly diverse datasets and face challenges like intermittent connectivity and varying computational resources. These factors make FL more complex than standard DL, necessitating specialized algorithms and strategies to ensure effective client collaboration.

One of the fundamental challenges in FL is statistical data heterogeneity [15] (sometimes simply referred to as statistical heterogeneity or data heterogeneity), which occurs because data distributions differ across various clients. Unlike centralized training, where all data comes from a single distribution, FL often deals with non-independent and identically distributed (non-i.i.d.) data. For example, different hospitals collaborating together in a federated medical imaging system may collect data from distinct patient populations, each exhibiting varying disease prevalence, imaging protocols, or sensor types. Similarly, in a FL scenario for smartphone applications, user-generated images may vary significantly based on geographical location, device type, or individual usage patterns. This heterogeneity creates issues such as client drift [16], where local model updates diverge from each other, leading to instability in the convergence of the global model. Addressing statistical heterogeneity is essential to ensure that federated models generalize well across all clients and do not disproportionately favor certain data distributions while neglecting others.

FL is a rapidly evolving research area, with increasing studies exploring its challenges like statistical heterogeneity. FL has recently attracted broad interest in

the application to computer vision tasks [17]. However, the application of FL to these scenarios introduces factors and characteristics to the FL problem, interacting with and contributing to statistical heterogeneity in complex ways. For example, in an autonomous driving application, variations in scene quality, lighting conditions, sensor types, or annotation styles among different clients can exacerbate the non-i.i.d.-ness of the data, resulting in unexpected behaviors during federated training. Additionally, certain vision tasks, such as object detection and segmentation, may be more sensitive to these heterogeneous conditions than others. This sensitivity may require specialized approaches to address the associated challenges effectively.

Understanding how different levels and types of heterogeneity impact vision models is crucial for designing robust algorithms for computer vision tasks. Furthermore, uncovering hidden or non-obvious interactions between heterogeneity and specific computer vision tasks could lead to innovative solutions that enhance the effectiveness of FL in real-world applications. Addressing these open questions will be vital for advancing the practical implementation of FL as it continues to develop.

1.1 Contributions and outline

This manuscript aims to provide new solutions addressing statistical heterogeneity in FL, specifically focusing on computer vision applications. The manuscript is structured as follows:

- Chapter 2 introduces Federated Learning. FL has emerged as a solution to the growing demand for data-driven models while addressing privacy concerns and regulatory challenges.

  First, Chapter 2 outlines the motivations for FL, highlighting the differences between FL and traditional DL. Next, a taxonomy of FL is presented, distinguishing between horizontal FL, vertical FL, and federated transfer learning and between cross-silo and cross-device FL. Then, the chapter formally introduces FL and two fundamental FL algorithms present in the existing literature, Federated Averaging (`FedAvg`) [13] and Adaptive Federated Optimization (`FedOpt`) [18], and the core challenges associated with FL, focusing on statistical heterogeneity, which, in the context of computer vision applications,

primarily stems from class and domain imbalance. Finally, the chapter discusses the privacy risks related to FL.

- Chapter 3 presents Federated Recursive Ridge Regression (`Fed3R`), a novel FL algorithm immune to statistical heterogeneity. `Fed3R` constructs a closed-form FL classifier from latent feature embeddings originating from a pre-trained model without disclosing any private client data. `Fed3R` demonstrates to be cost-efficient and performative. Moreover, it can be applied together with other gradient-based FL algorithms.

  First, Chapter 3 introduces the image classification task, how ridge regression can be repurposed to classification, and a technique to approximate kernel ridge regression for resource-constrained applications. Then, the concepts of client drift and classifier data recency bias are introduced, leading to the introduction of `Fed3R` to solve these problems and showing that `Fed3R` parameters can serve as an initialization for the classifier, which can be fine-tuned using other FL algorithms. Finally, several experiments are presented, focusing on performance, speed, communication, and computation costs.

- Chapter 4 studies the importance of robust initialization in Personalized Federated Learning (PFL) [19], a variant of FL where clients develop their own local models tailored to their specific data distributions. In PFL, the main objective is not to create a global model that performs well across all clients but rather to assist clients in constructing the optimal local model, eventually benefiting from global updates influenced by data from other clients.

  This chapter analyzes the trade-off between leveraging global knowledge and focusing on local personalization, proposing a structured, systematic, and analytical analysis in which the training process is divided into four phases, each progressively more tailored to ultimately personalize the models of each client. This structured training strategy highlights the critical importance of model initialization in PFL. It shows that even in PFL settings, classifiers initialized with `Fed3R` parameters can significantly enhance performance.

  Additionally, a new method for full personalization, Only Local Labels (`OLL`), is introduced. `OLL` is a simple and efficient strategy that dramatically reduces classification errors by appropriately filtering the neurons of the local classifier. Experiments demonstrate that the combination of `Fed3R` initialization and full personalization using `OLL` yields the best results, evidenced by reduced

communication and computational costs, faster convergence, and improved weighted accuracy.

- Chapter 5 explores a use case for FL, *i.e.,* its application in autonomous driving within the context of the semantic segmentation task. It introduces `FedDrive`, a novel benchmark designed for the semantic segmentation task in the autonomous driving context. `FedDrive` investigates how statistical heterogeneity may arise from two primary sources: domain imbalance, which arises from different visual features of the scenes, and class imbalance, which arises when clients have access to the classes in different proportions. `FedDrive` systematically compares various style transfer and domain generalization techniques to tackle these issues and introduces 12 challenging FL datasets designed to amplify different types of heterogeneity and distribution shifts between the training and test sets.

  Additionally, Chapter 5 introduces a new real-world FL problem for semantic segmentation, `FFreeDA`, where clients only have access to unlabeled data while the server can utilize a source dataset before the training starts. To address `FFreeDA`, a new algorithm, `LADD`, is proposed, leveraging semi-supervised learning techniques and the clustered Federated Learning paradigm to address this problem effectively. Two new FL datasets are also introduced to study the `FFreeDA` problem.

  While Chapters 3 and 4 address the FL problem in computer vision, they use image classification as the clients' task for its simplicity, allowing the focus to remain on studying FL itself rather than the complexities and nuances of a specific application. Consequently, their findings are not strictly tied to image classification but are broadly applicable to FL in general. Instead, this chapter's primary focus is not on algorithmic approaches to addressing the heterogeneity challenges in FL in general but on a more complex scenario that presents its unique challenges related to heterogeneity in FL.

- Finally, Chapter 6 summarizes the contributions of this manuscript, highlighting the open challenges alongside future research directions.

1.2 Publications list

The author's publications are listed below in reverse chronological order (the most recent, first). The publications used as sources for drafting this manuscript are highlighted with a **bold** title.

- *A. Licciardi, D. Leo,* ***E. Fanì****, B. Caputo, M. Ciccone. "Interaction-Aware Gaussian Weighting for Clustered Federated Learning." International Conference on Machine Learning 2025 (****ICML25****)* [20].
- ***E. Fanì****, R. Camoriano, B. Caputo and M. Ciccone. "****Resource-Efficient Personalization in Federated Learning with Closed-Form Classifiers.****" IEEE Access, 2025* [21].
- ***E. Fanì****, R. Camoriano, B. Caputo and M. Ciccone. "****Accelerating Heterogeneous Federated Learning with Closed-form Classifiers.****" International Conference on Machine Learning 2024 (****ICML24****)* [22].
- *M. Dutto, G. Berton, D. Caldarola,* ***E. Fanì****, G. Trivigno, C. Masone. "Collaborative Visual Place Recognition through Federated Learning." International FedVision Workshop in Conjunction with the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024 (****FedVision CVPR24****)* [23].
- ***E. Fanì****, R. Camoriano, B. Caputo and M. Ciccone. "****Fed3R: Recursive Ridge Regression for Federated Learning with strong pre-trained models.****" International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with Neural Information Processing Systems 2023 (****FL@FM NeurIPS23****)* [24].
- ***E. Fanì****, M. Ciccone, B. Caputo. "****FedDrive v2: an Analysis of the Impact of Label Skewness in Federated Semantic Segmentation for Autonomous Driving.****" 5th Italian Conference on Robotics and Intelligent Machines, 2023. (****IRIM23****)* [25].
- D. Shenaj*, ***E. Fani****, M. Toldo, D. Caldarola, A. Tavera, U. Micheli, M. Ciccone, P. Zanuttigh, B. Caputo. "****Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning.****" IEEE/CVF Winter Conference on Applications of Computer Vision, 2023 (****WACV23****)* [26].

- L. Fantauzzo*, ***E. Fanì****, *D. Caldarola, A. Tavera, F. Cermelli, M. Ciccone, B. Caputo. "**FedDrive: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving.**" IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022. (**IROS22**)* [27].

* equal contribution.

Preprint articles:

- *G. D. Németh, **E. Fanì**, Y. J. Ng, B. Caputo, M. A. Lozano, N. M. Oliver, N. Quadrianto. "FedDiverse: Tackling Data Heterogeneity in Federated Learning with Diversity-Driven Client Selection" 2025* [28].

Chapter 2

Federated Learning

Federated Learning (FL) is a relatively new paradigm for distributed machine learning training. Firstly introduced by researchers at Google in the context of mobile devices and edge computing through a blog post [29] in 2017, and later published in the Artificial Intelligence and Statistics conference [13], the goal is to enable collaborative training of machine learning models across a network of devices without centralizing the data, thus enhancing privacy and security.

This section provides an introduction to FL. Specifically:

- Section 2.1 provides an overview of when and why FL has emerged, which are its motivations, and the differences with the more classical DL framework.
- Section 2.2 classifies the different types of FL.
- Section 2.3 formally introduces the FL framework, presenting the vanilla `FedAvg` [13], and its generalized version, the `FedOpt` [18] algorithm.
- Finally, Section 2.4 introduces the main challenges in FL.

2.1 Motivations

Nowadays, training machine learning models require an enormous quantity of data, and this data typically needs to be stored in a centralized *server*. However, a large portion of this data is generated by users and may disclose private information if it

is collected for training. Therefore, FL emerges as a potential solution to prevent privacy leaks while still allowing machine learning model training.

2.1.1 The need for data

The performance of machine learning models typically improves with larger datasets, as the increased volume of data enables them to capture complex patterns and reduce overfitting [31, 32]. This principle has been consistently observed across various domains, where the availability of extensive data significantly enhances the model's ability to generalize and produce accurate predictions. Figure 2.1 demonstrates the increasing demand for data over time. Modern language datasets now require up to 10^{13} data points, while vision datasets need as much as 10^{10} data points.

Fortunately, the rising demand for data aligns with the increasing global production of data. As illustrated in Figure 2.2, the amount of data generated worldwide each year continues to grow. Specifically, the world is producing 2.5 exabytes (EB) per day, which is equivalent to 2.5×10^6 terabytes (TB) of data, and a recent study [33] predicted that, for the end of 2024, the world generated a total of 147 zettabytes (ZB) of data. This is equivalent to 147 trillion gigabytes, or 147×10^{21} bytes, which is an enormous quantity when compared to 2010, just fourteen years earlier, when the data generated worldwide was only 2 ZB [34]. This data primarily comes from internet-related sectors, including social media, communication methods like messaging, video calls, and emails, digital photography, services such as music streaming, mobile payment systems, IoT devices, and others [34].

2.1.2 The need for privacy

Although the world is generating such a huge amount of data on a daily basis, it is difficult and often prohibitive to collect this data in a central server to train a machine learning model. Indeed, data is often generated by private users (*e.g.,* data from social media) and can potentially expose private information (*e.g.,* data associated with hospital patients), raising privacy and ethical issues. Moreover, governments are increasing their awareness of privacy and the value of personal data worldwide, enacting laws and regulations to protect individual privacy rights. Indeed, since the enactment of the General Data Protection Regulation (GDPR) [12] in 2018 in Europe,

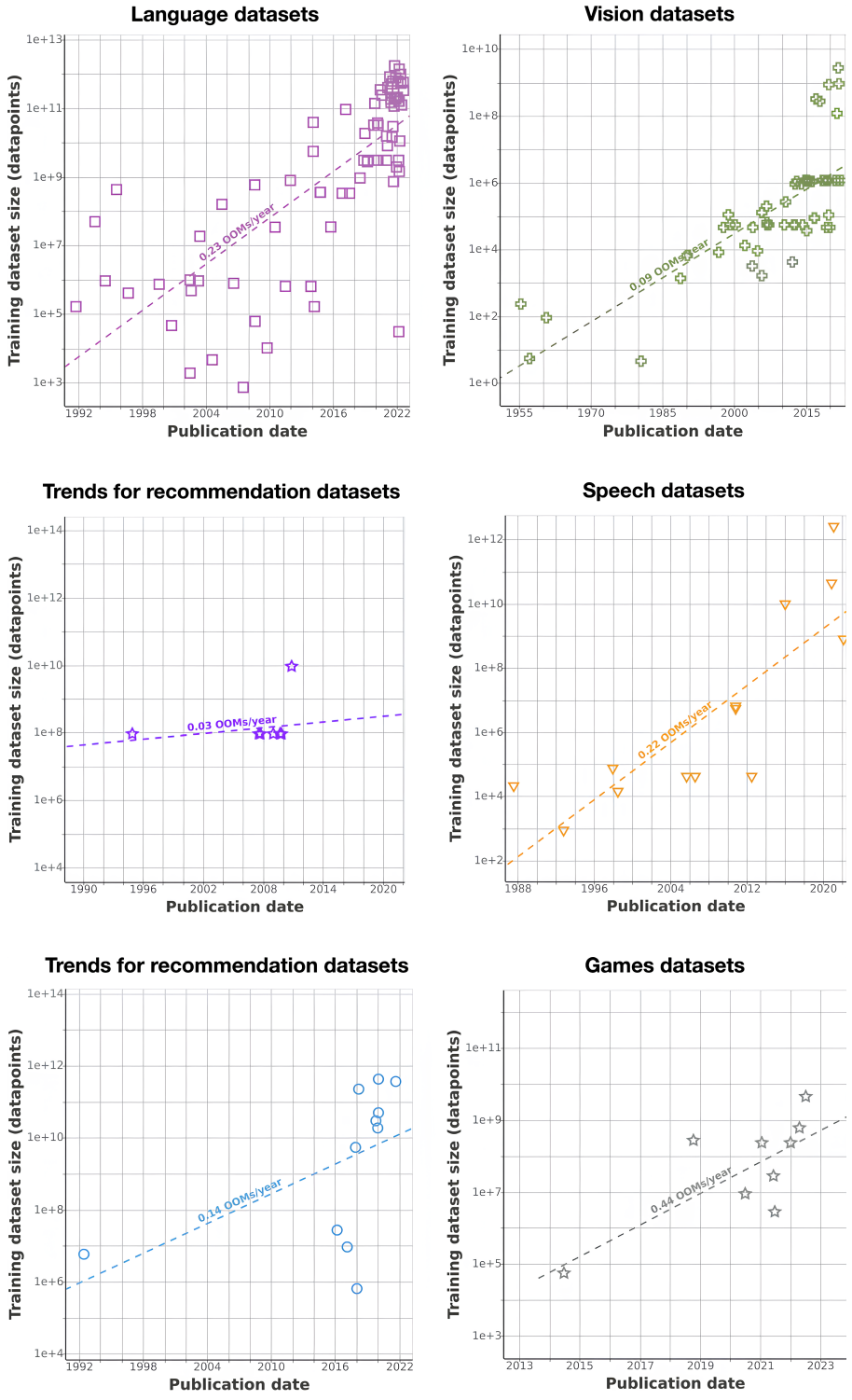


Figure 2.1: Training dataset size (in number of datapoints) by year among six distinct domains of application. Source: [30].

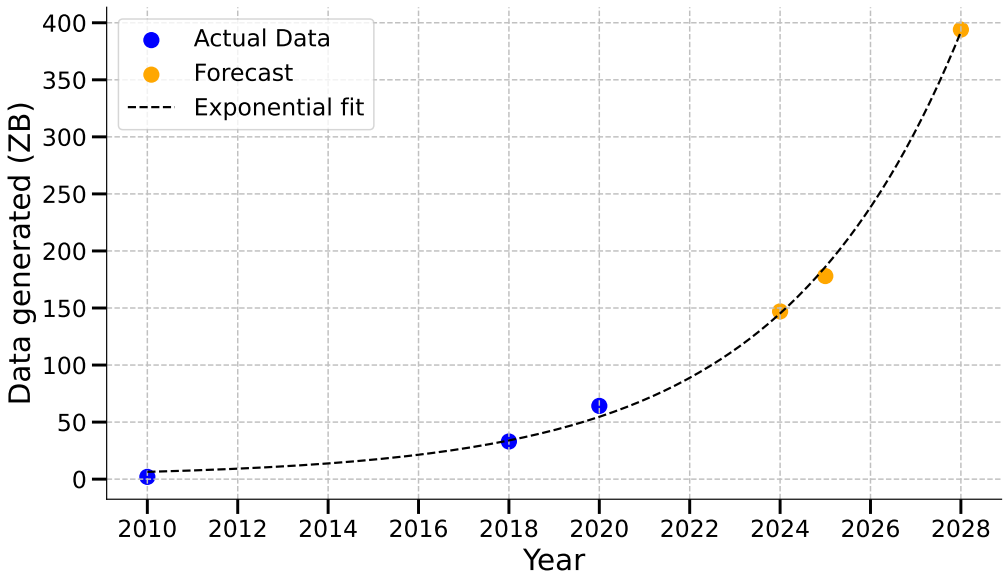


Figure 2.2: Amount of data generated worldwide per year. The amount of data generated globally each year is increasing exponentially. The data has been sourced from public resources [33–38]. The forecasted data is an average derived from these sources.

other countries were inspired to enact similar data protection rules [39]. As of 2024, over 160 privacy laws have been introduced worldwide [40], and it was predicted that by the end of 2024, 75% of the global population to have their personal data protected under these regulations [41]. There is growing awareness among people regarding the importance of data privacy. According to [39], 86% of individuals in the U.S. believe that data privacy is important, and 72% advocate for increased government regulation concerning personal data management.

Due to these challenges, companies often find it difficult to collect data from users. Additionally, there are significant costs associated with the data collection process, including expenses related to network infrastructure, scalability, and latency issues. A major concern is that users may be unwilling to share their private data with companies. For example, owners of autonomous vehicles might hesitate to disclose their car's location, smartphone users may prefer not to share personal photos, and healthcare providers might refrain from sharing sensitive patient information.

2.1.3 Federated Learning as a potential solution

FL is emerging as a promising solution for leveraging the vast amounts of data generated by private users, institutions, and companies — referred to as *clients* in the context of FL. This approach enables exploiting private data for machine learning model training while reducing privacy risks by eliminating the need to collect data on a central server [15]. It is a versatile paradigm that supports a wide range of machine learning tasks, including classification [42, 43], regression [44], recommendation systems [45], semantic segmentation [46], and others [47–49].

Some of its real-world applications include training neural networks to classify medical images without the need to centralize sensitive patient data for brain tumor segmentation [50, 51] and chest X-ray diagnosis [52] in the healthcare sector. Additionally, it is used in finance, where it supports fraud detection [53] and credit scoring [54] while ensuring client privacy. It also enhances smart devices by powering personalized keyboard suggestions [29] and speech recognition [55]. Furthermore, FL plays a significant role in autonomous systems [56], enabling the collaborative training of navigation algorithms without jeopardizing the proprietary data of individual vehicles. This wide-ranging applicability underscores FL's potential to transform industries where data privacy and decentralized computation are crucial.

2.1.4 Distributed vs. Federated Learning

FL can be seen as an extension of the Distributed Learning (DL) framework. The authors of [14] summarize the key differences between these two scenarios. The most significant difference is that DL does not have privacy concerns. In DL, data is centralized and then divided among computing nodes, typically in a balanced, i.i.d. manner, aiming to enhance the training process by maximizing speed and performance while minimizing training costs.

In contrast, FL is designed with privacy in mind. It assumes that devices generate their own data, which is never shared with other devices or the server. The goal of FL is to utilize the private data generated by these devices, shifting the training process to the edge and leveraging their computational power. This distinction is crucial: while data in DL is centralized and distributed to computing nodes, FL seeks to exploit data generated by private devices without compromising privacy. Therefore, devices in

FL can vary widely in local data distribution and computational capabilities, leading to unique challenges such as dealing with non-i.i.d. data distributions, unreliable communication networks, and resource-constrained devices.

While DL benefits from high-speed, low-latency connections and resource-rich systems, FL operates on devices like smartphones or IoT devices, necessitating the development of communication-efficient and robust algorithms. Additionally, FL must confront issues related to adversarial threats and untrustworthy participants. More details on the specific challenges of FL are discussed in Sections 2.4.1 to 2.4.3.

2.2 Federated Learning taxonomy

Depending on the distribution of data across the clients, the type of data the clients possess, the total number of clients, their reliability, and the available communication and computation budget, FL systems can be distinguished in different ways. Sections 2.2.1 and 2.2.2 give an overview of the types of FL systems.

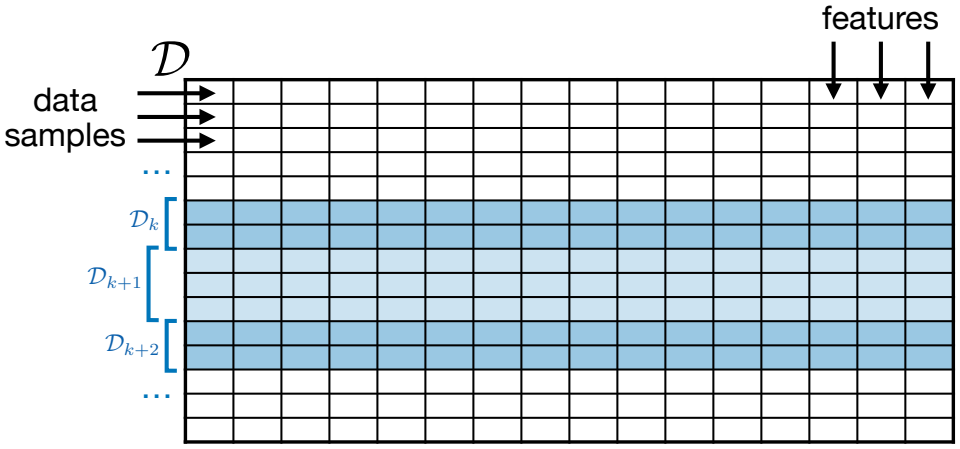
2.2.1 Horizontal FL, Vertical FL, Federated Transfer Learning

According to how the data is distributed across the clients, FL is subdivided into three categories: Horizontal FL (HFL), Vertical FL (VFL), and Federated Transfer Learning (FTL) [57]. The different types are pictorially illustrated in Figure 2.3 for tabular data.

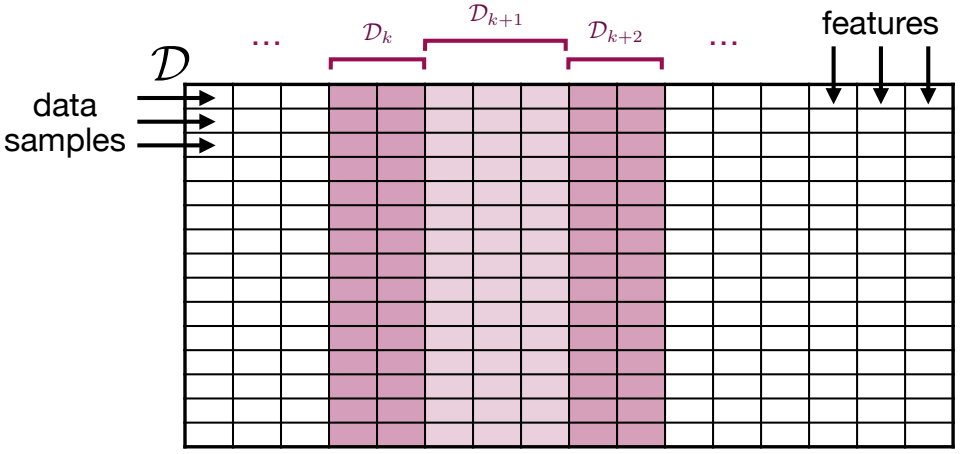
Figure 2.3a illustrates how local datasets are partitioned in HFL. In this case, each client possesses different samples, but all samples share the same set of features. For example, different banks may have information about the same types of users, but each bank serves different customers.

In contrast, Figure 2.3b shows that, in VFL, clients have access to the same samples, but each one has a unique set of features related to those samples. An example of this could involve two hospitals that treat the same patients; one hospital may have data such as blood pressure, heart rate, and blood sugar levels, while the other has information on average body temperature and oxygen saturation.

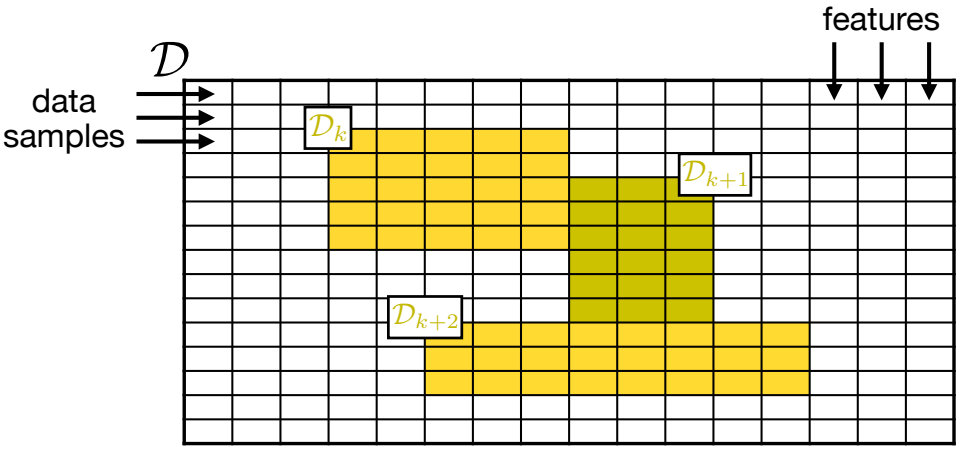
Finally, Figure 2.3c illustrates FTL, a hybrid case where some samples or features overlap can occur among different client datasets.



(a) Horizontal Federated Learning.



(b) Vertical Federated Learning.



(c) Federated Transfer Learning.

Figure 2.3: Different types of FL according to how data is distributed across the clients. For simplicity, these illustrations represent clients with tabular data, where rows are the samples and the columns are the features.

The focus of this manuscript is on HFL, which is widely adopted across various domains, particularly in real-world applications where organizations with similar data structures seek to collaborate while preserving privacy. HFL has gained significant traction, as highlighted by its use in diverse industries such as finance [58, 59], healthcare [50–52], and retail [60, 61], where entities often collect analogous types of data [57, 15].

Furthermore, HFL is especially relevant for computer vision applications, where tasks such as image classification [62, 13, 15], object detection [63, 64], and facial recognition [65, 66] benefit from collaborative training on distributed datasets without sharing raw images. For example, federated approaches have been applied to medical imaging [67] and autonomous driving [63, 25, 27] systems, demonstrating their practical utility and effectiveness [14].

The following of this manuscript will always implicitly refer to HFL unless specified otherwise.

2.2.2 Cross-silo and cross-device Federated Learning

Depending on the number of clients participating in the federation and on the characteristics of these clients, it is possible to distinguish between cross-silo FL and cross-device FL [14].

Cross-silo FL involves collaboration among a small number of participants, such as organizations or institutions. Each typically has access to substantial computational resources and large datasets. These participants often operate over reliable, high-bandwidth networks, such as corporate or private data center connections, which makes communication more efficient and less constrained than in other settings.

In contrast, cross-device FL focuses on large-scale collaboration among potentially millions of edge devices, such as smartphones, IoT devices, or wearables. These devices are usually resource-constrained, with limited computational power, storage, and energy. They are also frequently offline or connected via unreliable and low-bandwidth networks, which adds complexity to communication and synchronization.

Table 2.1 summarizes the main differences between cross-silo and cross-device FL. The following of this manuscript always implicitly refers to cross-device FL unless specified otherwise. The next sections formally introduced the general

Table 2.1: Comparison between cross-silo and cross-device FL [14].

	Cross-silo FL	**Cross-device FL**
Number of clients	small ($\sim 10^2, 10^3$)	large ($> 10^4$, potentially millions)
Size of the local datasets	large	small
Client reliability	total	unknown
Client availability	high	low
Client resources	high	unknown

FL framework and training pipeline, with a focus on two pioneering algorithms: Federated Averaging (`FedAvg`) [13] and its generalization, Adaptive Federated Optimization (`FedOpt`) [18].

2.3 Formal introduction to Federated Learning

Let $\mathcal{S}$ be the server that orchestrates the training procedure, and let $\mathcal{K}$ be a set of K clients. Each *client* k has access to a private *local dataset* $\mathcal{D}_k$ of $n_k \in \mathbb{N}$ training images, *i.e.,* $|\mathcal{D}_k| = n_k$. The local datasets $\mathcal{D}_k$ are made of *samples* (x, y), where $x \in \mathcal{X}$ is the *input* and $y \in \mathcal{Y}$ is its associated *target*, $\mathcal{X}$ is the *input space* and $\mathcal{Y}$ is the *target space*. All the clients $k \in \mathcal{K}$ have access to the same *model* $f : \mathcal{X} \to \mathcal{Y}$, parameterized by some *parameters* $\theta \in \Theta$, where Θ is the *parameters space*.

Each local dataset can be interpreted as a set of realizations of *random samples* $(\boldsymbol{x}, \boldsymbol{y})$ drawn from the *local distribution* $p_k(\boldsymbol{x} = x, \boldsymbol{y} = y)$. The FL objective is to find the optimal parameters $\theta^* \in \Theta$ that minimize the expected risk:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim p}[\mathcal{L}(f(x;\theta), y)], \tag{2.1}$$

where $p = \sum_{k \in \mathcal{K}} q_k p_k$ is the *global distribution* which is a mixture of the local distributions p_k, with weights $q_k \in [0, 1]$ such that $\|q\|_1 = 1$, and $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a loss function.

Equivalently, Equation 2.1 can be rewritten as:

$$\theta^* = \arg\min_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim\sum_{k\in\mathcal{K}} q_k p_k}[\mathcal{L}(f(x;\theta),y)] = \tag{2.2}$$

$$= \arg\min_{\theta\in\Theta} \sum_{k\in\mathcal{K}} q_k \mathbb{E}_{(x,y)\sim p_k}[\mathcal{L}(f(x;\theta),y)]. \tag{2.3}$$

Empirically, the true probability density functions p and $p_k \forall k \in \mathcal{K}$ are unknown, and the server does not have access to the local datasets $\mathcal{D}_k$. Therefore, the following (*centralized*) Empirical Risk Minimization (ERM) problem:

$$\theta^* = \arg\min_{\theta\in\Theta} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f(x;\theta),y), \tag{2.4}$$

where $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$, cannot be addressed. Conversely, in FL, each client individually contributes to the final solution by aiming to solve the following Federated ERM problem:

$$\theta^* = \arg\min_{\theta\in\Theta} \sum_{k\in\mathcal{K}} q_k \sum_{(x,y)\in\mathcal{D}_k} \mathcal{L}(f(x;\theta),y). \tag{2.5}$$

A typical FL algorithm is scheduled in $T \in \mathbb{N}$ *rounds*. Figure 2.4 graphically describes one round. During each round $t \in [T]$[1], the server shares a message $\mathcal{M}^t_{\mathcal{S}\to k}$ with a subset of *active clients* $\mathcal{K}^t \subseteq \mathcal{K}$. This message typically includes at least the *global parameters* $\theta^t \in \Theta$ of the current round t. The clients initialize their local copy of the model with the parameters θ^t (or with values that are a function of the message $\mathcal{M}^t_{\mathcal{S}\to k}$) and perform local optimization, obtaining the updated parameters θ^{t+1}_k. Then, each active client shares a message $\mathcal{M}^{t+1}_{k\to\mathcal{S}}$ with the server. This message shall not disclose any privacy-preserved information and typically includes at least the updated parameters (or some values that are a function of the updated parameters). Finally, the server updates the global model using the received messages, obtaining a new global model θ^{t+1} for the next round.

[1] In this manuscript, $[x] = \{1, 2, ..., x\}, x \in \mathbb{N}$.

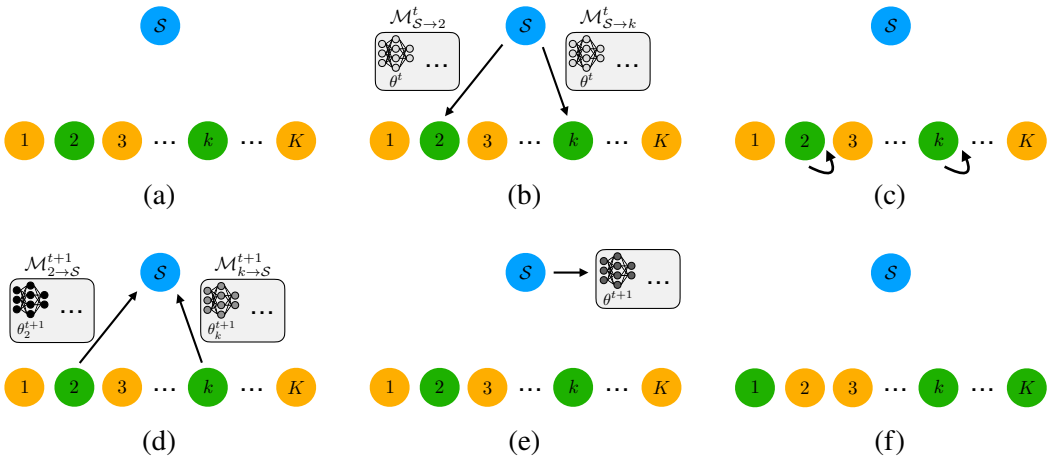


Figure 2.4: An illustration of a FL round. The active clients are represented in green, the inactive clients are represented in orange, and the server is represented in blue. a) Round t begins; the server samples a subset of active clients. b) The server shares a message with the active clients. c) The active clients train the model locally using the information shared from the server. d) The active clients share a message with the server. e) The server uses the received messages to update the global model and eventually to perform additional operations. f) The server samples a new subset of active clients for round $t+1$.

2.3.1 Federated Averaging (`FedAvg`)

Federated Averaging (`FedAvg`) is the standard method used in FL, introduced in the foundational paper by McMahan *et al.* [13]. In each round t of the `FedAvg` algorithm, the server randomly samples a subset $\mathcal{K}^t$ of $\kappa \in [K]$ clients according to a uniform distribution, with the subset size remaining constant across all rounds. Then, the server then shares the global model parameters with each selected client; specifically, $\mathcal{M}^t_{\mathcal{S}\to k} = \{\theta^t\},\ \forall k \in \mathcal{K}^t$. Once the clients in $\mathcal{K}^t$ receive the global model parameters, they initialize the local copy of the model with the received global parameters and train it using the local datasets $\mathcal{D}_k$ for a fixed given number of epochs $E \in \mathbb{N}$ using Stochastic Gradient Descent (SGD) [68], obtaining the updated local parameters θ^{t+1}_k. Finally, each client $k \in \mathcal{K}^t$ sends its updated local parameters θ^{t+1}_k to the server, *i.e.*, $\mathcal{M}^{t+1}_{k\to\mathcal{S}} = \{\theta^{t+1}_k\}\ \forall k \in \mathcal{K}^t$, and the server aggregates them with a weighted average as follows:

$$\theta^{t+1} = \sum_{k\in\mathcal{K}^t} \frac{n_k}{n_t}\theta^{t+1}_k, \tag{2.6}$$

where $n_t = \sum_{k\in\mathcal{K}^t} n_k$, obtaining the new global model for the next round $t+1$.

The complete algorithm is presented in Algorithm 1.

Algorithm 1 - Federated Averaging (`FedAvg`) [13]

Require: $\mathcal{S}$, $\mathcal{K}$, f, initial model parameters θ^1, E, batch size B, learning rate η

Server:

for each round $t \in [T]$ **do**
  Uniformly sample $\mathcal{K}^t \subseteq \mathcal{K}$ such that $\left|\mathcal{K}^t\right| = \kappa$
  for each client $k \in \mathcal{K}^t$, *in parallel* **do**
    Send the global parameters to client k: $\mathcal{M}^t_{\mathcal{S}\to k} = \{\theta^t\}$
    $\mathcal{M}^{t+1}_{k\to\mathcal{S}} = \{\theta^{t+1}_k\} = \texttt{ClientUpdate}(k, E, B, \eta, \theta^t)$
  end for
  $\theta^{t+1} = \sum_{k\in\mathcal{K}^t} \frac{n_k}{n_t}\theta^{t+1}_k$
end for

Client:

`ClientUpdate`$(k, E, B, \eta, \theta^t)$ **{** *// run on client k*
Initialize local parameters: $\theta_k = \theta^t$
$\mathcal{B} = \{$split $\mathcal{D}_k$ into batches of size $B\}$
for each $e \in [E]$ **do**
  for each $b \in \mathcal{B}$ **do**
    $\theta_k \leftarrow \theta_k - \eta\nabla\mathcal{L}(b;\theta_k)$
  end for
end for
Return $\mathcal{M}^{t+1}_{k\to\mathcal{S}} = \{\theta_k\}$
}

2.3.2 Generalizing `FedAvg`: adaptive federated optimization (`FedOpt`)

The authors of [18] show that `FedAvg` is a specific instance of a more general FL framework. Indeed, the `FedAvg` aggregation rule presented in Equation 2.6 can be reformulated as follows:

$$\theta^{t+1} = \sum_{k \in \mathcal{K}^t} \frac{n_k}{n_t} \theta_k^{t+1} = \sum_{k \in \mathcal{K}^t} \frac{n_k}{n_t} \left(\theta_k^{t+1} - \theta^t + \theta^t \right) =$$
$$= \sum_{k \in \mathcal{K}^t} \frac{n_k}{n_t} \theta^t + \sum_{k \in \mathcal{K}^t} \frac{n_k}{n_t} \Delta_k^{t+1} = \frac{1}{n_t} \theta^t \sum_{k \in \mathcal{K}^t} n_k - \Delta^{t+1} = \theta^t - \Delta^{t+1}, \tag{2.7}$$

where $\Delta_k^{t+1} := \theta_k^{t+1} - \theta^t$ and $\Delta^{t+1} := -\sum_{k \in \mathcal{K}^t} \frac{n_k}{n} \Delta_k^{t+1}$. With this formulation, [18] demonstrates that the aggregation step in `FedAvg` is equivalent to applying SGD using the *pseudo-gradient* Δ^{t+1}, with a *server* learning rate $\eta_{\mathcal{S}} = 1$.

Based on this observation, they propose a new algorithm, `FedOpt`, where the messages from the clients to the server are $\mathcal{M}_{k \to \mathcal{S}}^{t+1} = \{\Delta_k^{t+1}\}$. The server uses the received values to compute the pseudo-gradient Δ^{t+1}, which is in turn used with any desired optimizer to compute the new global model θ^{t+1}.

The complete algorithm is presented in Algorithm 2.

2.4 Federated Learning challenges

This section outlines the primary challenges associated with FL. Specifically, Section 2.4.1 addresses the issues related to statistical data heterogeneity, the main FL challenge addressed in this manuscript. Moreover, Section 2.4.2 and Section 2.4.3 briefly discusses system heterogeneity and the privacy risks that still impact the FL framework, despite its design aimed at minimizing these risks.

2.4.1 Statistical heterogeneity

A key challenge in FL is *statistical data heterogeneity* (or, shortly, *statistical heterogeneity*), which refers to the issue of having data distributed in a non-i.i.d. manner among clients. These variations in data distribution stem from differences in user habits [69], geographical locations [70, 19], and device preferences.

Figure 2.5 shows the effects of different levels of statistical heterogeneity. Higher levels of statistical heterogeneity degrade performance, slowing training, making

Algorithm 2 - Adaptive Federated Optimization (`FedOpt`) [18]

Require: $\mathcal{S}$, $\mathcal{K}$, f, initial model parameters θ^1, E, batch size B, client learning rate η, server learning rate $\eta_{\mathcal{S}}$

Server:

for each round $t \in [T]$ **do**
  Uniformly sample $\mathcal{K}^t \subseteq \mathcal{K}$ such that $\left|\mathcal{K}^t\right| = \kappa$
  for each client $k \in \mathcal{K}^t$, *in parallel* **do**
    Send the global parameters to client k: $\mathcal{M}^t_{\mathcal{S}\to k} = \{\theta^t\}$
    $\mathcal{M}^{t+1}_{k\to\mathcal{S}} = \{\Delta^{t+1}_k\} = \texttt{ClientUpdate}(k, E, B, \eta, \theta^t)$
  end for
  $\Delta^{t+1} = -\sum_{k\in\mathcal{K}^t} \frac{n_k}{n_t}\Delta^{t+1}_k$
  $\theta^{t+1} = \texttt{ServerOpt}(\theta^t, \Delta^{t+1}, \eta_{\mathcal{S}})$ *// It could be any optimizer, e.g., SGD*
end for

Client:

`ClientUpdate`$(k, E, B, \eta, \theta^t)$ **{** *// run on client k*
Initialize local parameters: $\theta_k = \theta^t$
$\mathcal{B} = \{$split $\mathcal{D}_k$ into batches of size $B\}$
for each $e \in [E]$ **do**
  for each $b \in \mathcal{B}$ **do**
    $\theta_k \leftarrow \theta_k - \eta\nabla\mathcal{L}(b; \theta_k)$
  end for
end for
$\Delta^{t+1}_k = \theta^{t+1}_k - \theta^t$
Return $\mathcal{M}^{t+1}_{k\to\mathcal{S}} = \{\Delta^{t+1}_k\}$
}

it more unstable, and increasing the number of communication rounds needed to converge [75], therefore growing the communication and computation costs.

Statistical heterogeneity can arise from different sources, depending on the characteristics of the data collection or generation process of the clients [14]. In the following, the possible sources of statistical heterogeneity are described for two clients $h, k \in \mathcal{K}, h \neq k$. In general, if any of the types of statistical heterogeneity is present for at least two clients $h, k \in \mathcal{K}$, and $|\mathcal{K}| > 2$, the federation is affected by the specific type of statistical heterogeneity to some extent.

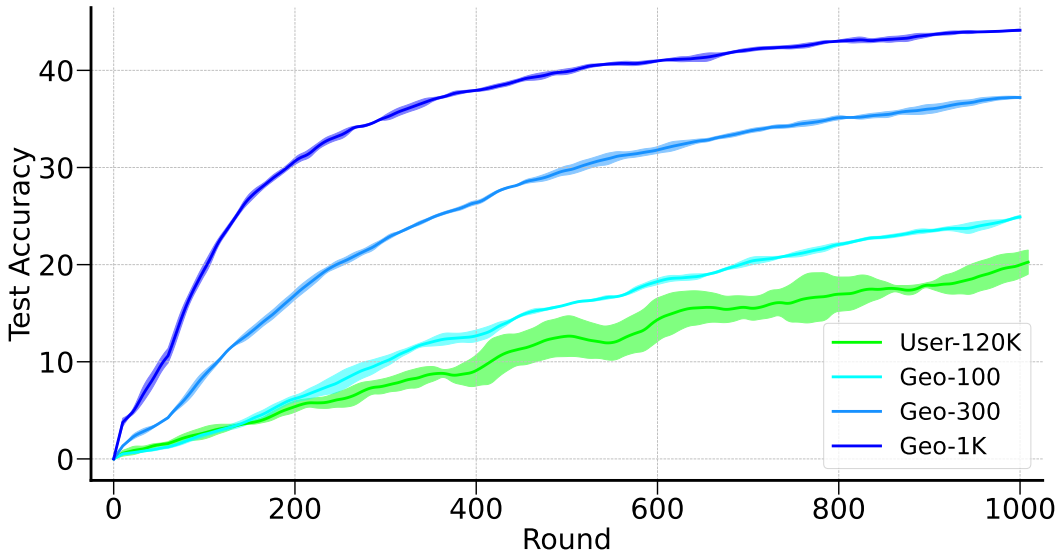


Figure 2.5: Statistical Heterogeneity effects on test accuracy. Experiments have been conducted using three different random seeds, and the confidence regions are displayed within one standard deviation. The model is a MobileNet V2 [71] pre-trained on ImageNet [72]. The model parameters are fixed, except for the last layer, *i.e.,* the classification layer, which is fine-tuned using `FedAvg`. The centralized dataset is iNaturalist [73], and it is partitioned into clients as in [74], constituting four distinct federated scenarios which are, from the most heterogeneous to the least: User-120K, Geo100, Geo300, Geo1K. More details on these scenarios are shown in Table 3.1.

Class imbalance. Also known as *label distribution skew*, class imbalance occurs when clients have access to samples whose classes are typically different or present with different proportions. Formally, the marginal local target distribution $p_k(\boldsymbol{y} = y)$ of a client k may vary from the marginal local target distribution $p_h(\boldsymbol{y} = y)$ of client h, *i.e.,* $p_k(\boldsymbol{y} = y) \neq p_h(\boldsymbol{y} = y)$, although the conditional distribution of the inputs given the targets can be shared, *i.e.,* $p_k(\boldsymbol{x} = x|\boldsymbol{y} = y) = p_h(\boldsymbol{x} = x|\boldsymbol{y} = y)$. For instance, in a wildlife image classification task, a client might have thousands of deer pictures but only a handful showing bears, while another client may have thousands of bears pictures but only a handful showing deers. The statistical heterogeneity of the experiments shown in Figure 2.5 is primarily generated by class imbalance.

Domain imbalance. Domain imbalance refers to the heterogeneity that arises when different clients observe data belonging to distinct domains — due to variation in culture, environment, device type, personal habits, or other contextual factors.

Specifically, in the presence of domain imbalance, clients may exhibit different features associated with a given target.

Domain imbalance can be subdivided into two types of heterogeneity, originally introduced by [14]:

- **Feature distribution skew.** Also known as *covariate shift*, it refers to clients having different input features distributions, even if the conditional label distribution is the same. Formally, the marginal local input distribution $p_k(\boldsymbol{x} = x)$ may vary from the marginal local input distribution $p_h(\boldsymbol{x} = x)$, *i.e.,* $p_k(\boldsymbol{x} = x) \neq p_h(\boldsymbol{x} = x)$, although the conditional distribution of the targets given the inputs is the same, *i.e.,* $p_k(\boldsymbol{y} = y|\boldsymbol{x} = x) = p_h(\boldsymbol{y} = y|\boldsymbol{x} = x)$. For instance, in a speech recognition task, the same phrases may be spoken with varying accents, background noise levels, or microphone qualities across clients.

- **Concept shift.** This type of domain imbalance occurs when the distributions of input features and labels are misaligned. There are two types of concept shift:

  - *Same label, different features.* This case is central to domain imbalance, as it captures how domains can diverge in their input-output mappings for the same label semantics. Formally, it happens when $p_k(\boldsymbol{x} = x|\boldsymbol{y} = y) \neq p_h(\boldsymbol{x} = x|\boldsymbol{y} = y)$, even if $p_k(\boldsymbol{y} = y) = p_h(\boldsymbol{y} = y)$. For instance, in an agricultural monitoring system, the label "healthy crop" may correspond to vastly different satellite image patterns across regions due to differences in soil color, irrigation style, or lighting conditions.
  - *Same features, different labels.* It is the opposite case, where $p_k(\boldsymbol{y} = y|\boldsymbol{x} = x) \neq p_h(\boldsymbol{y} = y|\boldsymbol{x} = x)$, even if $p_k(\boldsymbol{x} = x) = p_h(\boldsymbol{x} = x)$. For instance, in a personal finance app, the same transaction description might be labeled as "transportation" by one user and "business expense" by another, based on their spending context.

Quantity imbalance. Also known as *quantity skew*, quantity imbalance occurs when clients have access to datasets of different sizes, *i.e.,* $n_k \neq n_h$.

A summary of the different types of statistical heterogeneity is shown in Table 2.2.

Table 2.2: Summary of the types of statistical heterogeneity with respect to variations in conditional and marginal local distributions.

Statistical heterogeneity type			$p(\boldsymbol{x} = x)$	$p(\boldsymbol{y} = y)$	$p(\boldsymbol{x} = x \mid \boldsymbol{y} = y)$	$p(\boldsymbol{y} = y \mid \boldsymbol{x} = x)$	Example
Class imbalance				different	shared		In a wildlife image classification task, clients have different numbers of deer and bear images.
Domain imbalance	Feature distribution skew		different			shared	In a speech recognition task, the same phrases may be spoken with varying accents, background noise levels, or microphone qualities across clients.
	Concept shift	Same labels, different features		shared	different		In an agricultural monitoring system, the label "healthy crop" may correspond to different satellite image patterns due to differences in soil color, irrigation style, or lighting conditions.
		Same features, different labels	shared			different	In a finance app, the same transaction might be labeled as "transportation" by one user and "business expense" by another, based on their spending context.
Quantity imbalance			Clients have access to datasets of different sizes.				

A plethora of solutions has been proposed to address statistical heterogeneity in the literature. One of the simplest yet foundational ideas is the one of `FedProx` [76], which introduces a proximal term to align local updates with the global model. Other famous examples include `Scaffold` [16] and `Mime` [77], which use control variates to mitigate local update bias due to class imbalance, `FedDyn` [62], which adds a dynamic regularizer to the local objective to align local gradients with the global objective, and `FedNova`, [78] that proposes to normalize the client updates. Some methods align the clients' features using contrastive learning [79, 80], augment client data to improve generalization [81], or track the misalignment of local updates using drift variables [82].

2.4.2 System heterogeneity

System heterogeneity refers to the diversity and variability in the computational and communication capabilities of the devices involved and the conditions of their networks [14, 76]. Indeed, the infrastructure of the FL system is often not homogeneous because participating clients may be of various natures, such as smartphones or IoT devices, leading to various challenges.

For example, the clients' resources can vary significantly in terms of processing power, memory, and storage capacity. This disparity can lead to slower devices, known as *stragglers* [76], which may delay the overall training progress. Additionally, the communication between these devices and the central server can be affected by

differences in network bandwidth, latency, and reliability. Some devices may be connected to fast, stable Wi-Fi networks, while others may rely on slower, intermittent mobile connections.

Moreover, the participation of devices in the training process can be unpredictable. Factors such as battery constraints, user activity, or network disconnections might cause devices to join or leave the training unexpectedly. This irregular participation complicates the scheduling of tasks and the aggregation of models. Devices with limited data plans or high-cost connectivity options may contribute to the training process less frequently, which affects their representation in the global model.

2.4.3 Privacy risks

FL implicitly enhances privacy by ensuring that data remains on users' devices. However, it still faces several privacy risks.

One significant concern is inference attacks, where adversaries analyze gradients or model updates to infer sensitive information about local data. For example, gradients can sometimes disclose individual data points or even reconstruct input samples, especially in models that are prone to overfitting [83]. Additionally, attackers may perform membership inference to determine if specific data points were part of the training set [84] or conduct attribute inference to extract sensitive characteristics, such as age or gender, even if those attributes were not included in the training procedure [85].

Collusion among participants presents another significant risk. When multiple parties cooperate, they can combine their shared updates to extract information about another participant's data, increasing the likelihood of privacy breaches [86]. Furthermore, FL's reliance on shared updates makes it vulnerable to poisoning attacks, where malicious participants can manipulate the global model or embed backdoors that activate under specific conditions [87].

Additionally, communication can play a role in privacy risks. Communication protocols may leak metadata that inadvertently reveals details about a participant's dataset [88]. The potential for adversaries to replay, intercept, or modify communication channels adds further challenges to maintaining secure and private training [14].

Finally, while techniques such as differential privacy [89] and secure multiparty computation [90] offer some protections, they come with trade-offs. For instance, differential privacy can reduce model performance due to the noise it introduces, while secure computation methods often increase computational overhead, which may be impractical for resource-constrained devices. These limitations emphasize the ongoing challenges of balancing privacy and efficiency in FL systems.

Chapter 3

The effectiveness of closed-form classifiers for Federated Learning

Part of the content of this chapter, including the methodologies and the results, is adapted, with permission, from the following papers:

- ***E. Fanì**, R. Camoriano, B. Caputo and M. Ciccone. "**Accelerating Heterogeneous Federated Learning with Closed-form Classifiers.**" International Conference on Machine Learning 2024 (**ICML24**)* [22].
- ***E. Fanì**, R. Camoriano, B. Caputo and M. Ciccone. "**Fed3R: Recursive Ridge Regression for Federated Learning with strong pre-trained models.**" International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with Neural Information Processing Systems 2023 (**FL@FM NeurIPS23**)* [24].
- ***E. Fanì**, R. Camoriano, B. Caputo and M. Ciccone. "**Resource-Efficient Personalization in Federated Learning with Closed-Form Classifiers.**" **IEEE Access**, 2025* [21].

Recent FL literature highlights the benefits of using pre-trained models during the training process [91]. These models help mitigate a challenge known as *client drift* [16], which hampers training and is a direct consequence of statistical heterogeneity. However, it is almost always necessary to train the last layer of a neural network from scratch, despite starting from pre-trained models, as the source

task is generally misaligned with the target task. This presents a significant issue. Indeed, recent works [92, 93] emphasize that the classifier is the most critical layer in FL and is particularly vulnerable to data recency bias [94] and client drift.

This chapter first provides a brief introduction to image classification [1], ridge regression, and the concepts of client drift and data recency bias. It then presents a new algorithm called Federated Recursive Ridge Regression (`Fed3R`), which adapts the ridge regression solution to FL. This algorithm provides an exact, closed-form solution that is immune to statistical heterogeneity.

This chapter is structured as follows:

- Section 3.1 provides a concise introduction to the image classification task.
- Section 3.2 introduces the ridge regression task, including a recursive formulation for the linear ridge regression solution and kernel ridge regression, which implicitly maps the input space into a higher-dimensional feature space to achieve linear separability.
- Section 3.3 introduces the concepts of client drift and classifier data recency bias in FL, providing an overview of the strategies adopted in the existing literature to address these issues.
- Section 3.4 presents the Federated Recursive Ridge Regression (`Fed3R`) algorithm.
- Section 3.5 introduces a variant of `Fed3R` which approximates KRR by means of the random Fourier features [97].
- Section 3.6 proposes a synchronous version of `Fed3R` which improves the privacy of the clients.
- Section 3.7 shows that `Fed3R` parameters can also be used as robust initialization parameters for the classification layer of the model.

[1] In this manuscript, image classification is the task associated with the `Fed3R` algorithm. Therefore, the method is described accordingly, and the experiments are performed on this task, although in principle `Fed3R` can be utilized for various tasks. This task is also the primary focus of well-known FL benchmarks [95, 96], because it is simple enough to facilitate the analysis of the challenges of FL, isolating them from the specific difficulties associated with the particular task. An analysis of a specific instance of a real-world computer vision problem for FL will be discussed in Chapter 5.

- Section 3.8 describes existing related works on applications of ridge regression to FL.
- Section 3.9 presents the empirical experiments to validate the effectiveness of the `Fed3R` family of algorithms.
- Section 3.10 discusses the impact of the `Fed3R` algorithms and possible directions for future work.

3.1 The image classification task

In image classification, the objective of a model $f : \mathcal{X} \to \mathcal{Y}$ is to correctly predict the ground truth label $y \in \mathcal{Y}$ associated with each input image $x \in \mathcal{X}$. Each RGB image can always be converted to a $3 \times W \times H$ tensor; therefore, the input space $\mathcal{X}$ is typically $\mathbb{R}^{3\times W\times H}$, where 3 are the RGB channels, W is the width of the image, and H is its height. The target space $\mathcal{Y}$ is a finite set of C possible categories $\mathcal{C}$ which, for instance, could be $\{dog, cat, hamster, wolf, squirrel\}$, and each image could be associated with any of these labels, *e.g.,* dog.

Since working with raw categories is not practical, they are converted to one-hot encoding vectors. For example, if we consider the above (ordered) set of categories, a dog label would be represented as $[1, 0, 0, 0, 0]$, a $hamster$ as $[0, 0, 1, 0, 0]$, and a $squirrel$ as $[0, 0, 0, 0, 1]$. This method prevents any unintended ordinal relationships between the categories, which could mislead algorithms into interpreting a meaningful order where none exists. In other words, each label is always converted into its one-hot encoding vector, that is a vector in the set $\{e_c \in \{0, 1\}^C \mid e_{cc'} = \delta_{cc'}, \forall\ 1 \leq c, c' \leq C\}$, where $\delta_{cc'}$ is the Kronecker delta, defined as $\delta_{cc'} = 1$ if $c = c'$, 0 otherwise. Conversely, the output of the model f for any given input image $x \in \mathcal{X}$ is a probability vector $\hat{y} \in \Delta^{C-1}$, where $\Delta^{C-1} = \left\{v \in \mathbb{R}^C \mid v_c \geq 0\ \forall c \in [C], \sum_{c=1}^{C} v_c = 1\right\}$ is the standard probability simplex of dimension $C - 1$.

For practical reasons, for the image classification task, it is assumed that $\mathcal{Y} \subseteq \Delta^{C-1}$ in the following sections of this manuscript. With this definition of $\mathcal{Y}$, given the prediction $\hat{y} = f(x; \theta)$ of a model f, parameterized by θ, for a given input image $x \in \mathcal{X}$, the predicted category is the one in the position $\arg\max_{c\in[C]} \hat{y}_c$. For instance, if, with the above example, the prediction for an image $x \in \mathcal{X}$ is $\hat{y} = [0.1, 0.4, 0.2, 0.0, 0.3]$, the predicted class would be cat.

One popular choice for the loss function $\mathcal{L}$ for the image classification task is the Cross-Entropy (CE) loss [98], which seeks to maximize the likelihood of the observed data given the predicted probabilities for each class, and has the following form:

$$\mathcal{L}_{\text{CE}}(\hat{y}, y) = -\sum_{c=1}^{C} y_c \log \hat{y}_c, \tag{3.1}$$

where the predicted probabilities $\hat{y}_c$ are obtained by normalizing the predicted logits $\tilde{y}_c$ through the softmax function, $\forall c \in [C]$:

$$\hat{y}_c = \frac{e^{\frac{\tilde{y}_c}{\tau}}}{\sum_{c'=1}^{C} e^{\frac{\tilde{y}'_c}{\tau}}}, \tag{3.2}$$

where $\tau \in \mathbb{R}^+$ is the temperature hyper-parameter, which is typically set to 1. In particular, $\tau > 1$ produces a softer probability distribution, making the probabilities more uniform, while $\tau < 1$ produces a sharper distribution, making the model more confident in its predictions.

The most common metric to measure the performance of a model trained to solve the image classification is the *accuracy* which, given any evaluation dataset $\mathcal{D}$, is defined as follows:

$$\texttt{Accuracy} = \sum_{(x,y)\in\mathcal{D}} \frac{\mathbb{1}\left[\arg\max_{c\in[C]} \hat{y}_c = \arg\max_{c\in[C]} y_c\right]}{n}, \tag{3.3}$$

where $n \in \mathbb{N}$ is the number of images in $\mathcal{D}$, *i.e.,* $|\mathcal{D}| = n$, and $\mathbb{1}$ is the indicator function. In other words, the accuracy is the proportion of correctly predicted labels among all the evaluated samples.

3.2 Ridge regression

This section provides the necessary background related to ridge regression to understand the `Fed3R` family of algorithms presented later in this manuscript. Specifically:

- Section 3.2.1 introduces regression and the linear least squares problem.
- Section 3.2.2 firstly describes the linear least squares solution and then discusses ridge regression.
- Section 3.2.3 presents the recursive ridge regression formulation suitable for tasks such as incremental learning where data is available in batches.
- Section 3.2.4 shows how to find a ridge regression solution in higher dimensions without explicit feature mapping, thanks to the kernel trick.
- Finally, Section 3.2.5 presents the random Fourier features approximation of kernel ridge regression.

3.2.1 The regression task and the least squares problem

Differently from classification, the objective of the regression task is to predict the correct continuous scalar, or more than one scalar, associated with each input. One popular loss function for the regression task is the *L2 loss*, also known as *squared loss*:

$$\mathcal{L}_{\text{SQ}}(\hat{y}, y) = \|\hat{y} - y\|^2 , \tag{3.4}$$

where $\hat{y} = f(x; \theta)$. Both y and $\hat{y}$ belong to the Hilbert space $\mathcal{Y}$, and $\|\cdot\| = \sqrt{\langle \cdot, \cdot \rangle}$.

Consider random samples $(\boldsymbol{x}, \boldsymbol{y})$ drawn from the joint distribution $p(\boldsymbol{x} = x, \boldsymbol{y} = y)$. Using the L2 loss in Equation 3.4, the regression problem assumes the form of a least squares problem:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim p} \|f(x; \theta) - y\|^2 , \tag{3.5}$$

which can empirically be formulated as:

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{(x,y)\in \mathcal{D}} \|f(x; \theta) - y\|^2 , \tag{3.6}$$

where $\mathcal{D}$ is the training dataset. If we assume that $\mathcal{Y} \subseteq \mathbb{R}^C$, $C \in \mathbb{N}$, Equation 3.4 can be rewritten as follows:

$$\mathcal{L}_{\text{SQ}}(\hat{y}, y) = \|\hat{y} - y\|_2^2 = (\hat{y} - y)^2, \tag{3.7}$$

and the (empirical) least squares problem assumes the following form:

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in \mathcal{D}} (f(x;\theta) - y)^2 \,. \tag{3.8}$$

3.2.2 From linear least squares to ridge regression

Suppose that f is a linear model, *i.e.,* $f(x;\theta = \{W, \beta\}) = W^\top x + \beta$, where $W \in \mathbb{R}^{p \times C}$ and $b \in \mathbb{R}^C$. Note that it is possible to remove the intercept β from the definition of f, by integrating it within the input x and matrix W:

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad \tilde{W} = \begin{bmatrix} W \\ \beta \end{bmatrix}. \tag{3.9}$$

Therefore, it is possible to redefine f as $f(x;\theta = \{W, \beta\}) = \tilde{W}^\top \tilde{x}$. In the following, without loss of generality, we can ignore β and assume that $f(x;\theta = \{W\}) = W^\top x$.

With this linear model, the empirical least squares problem assumes the following form:

$$W^* = \arg\min_{W \in \mathbb{R}^{p \times C}} \sum_{(x,y) \in \mathcal{D}} (W^\top x - y)^2. \tag{3.10}$$

Stacking all the n inputs x into a matrix $X \in \mathbb{R}^{n \times p}$ and all the targets into a matrix $Y \in \mathbb{R}^{n \times C}$, $\forall (x, y) \in \mathcal{D}$, it is possible to rewrite Equation 3.10 in matrix form:

$$W^* = \arg\min_{W \in \mathbb{R}^{p \times C}} \|Y - XW\|_2^2 \,, \tag{3.11}$$

which admits the following closed-form solution [99–101]:

$$W^* = (X^\top X)^{-1} X^\top Y. \tag{3.12}$$

If the problem is ill-conditioned, *i.e.,* if the condition number of $X^\top X$ is high, this solution could be numerically unstable. Moreover, this simple linear predictor is generally prone to overfitting [102]. To mitigate this issue, an L2 regularization term (controlled by a Tikhonov hyper-parameter $\lambda \in \mathbb{R}^+$) can be incorporated into the objective function, leading to the Ridge Regression (RR) problem [103], also known as the regularized least squares problem. The RR problem can be formalized as:

$$W^* = \underset{W \in \mathbb{R}^{p \times C}}{\arg\min} \|Y - XW\|_2^2 + \lambda \|W\|_2^2, \tag{3.13}$$

and admits the following closed-form solution:

$$W^* = (X^\top X + \lambda I_p)^{-1} X^\top Y, \tag{3.14}$$

where I_p is the $p \times p$ identity matrix.

Since $X^\top X + \lambda I_p \succ 0$ for any $\lambda > 0$, no additional assumptions on the rank or the dimensions of the matrix X are required for the optimal solution W^* to exist [103]. Furthermore, the RR solution converges in probability to the optimal Bayes classifier as n tends to infinity [104–106].

Since the target space for image classification is a subset of $\mathbb{R}^C$, and as shown in [101], the classification problem can be effectively treated as a regression problem.

3.2.3 Recursive ridge regression

In many practical scenarios, such as Continual and Incremental Learning [107, 108], data becomes available sequentially rather than all at once. Similarly, in decentralized settings like FL, data is distributed across multiple clients rather than being centralized in one location. As a result, Equation 3.14 cannot be directly used to obtain the optimal linear parameters W^*. However, recursive RR [109] provides a practical approach to update the optimal solution as new data is collected or shared. This method utilizes the linearity of Equation 3.14 and employs techniques such as Sherman-Morrison-Woodbury or Cholesky updates [110, 111] for efficient computation.

Specifically, if the samples are sequentially available in batches $\{(X_t, Y_t)\}_{t=1}^{T}$ of size $n_t \in \mathbb{N}\ \forall t \in [T]$, $T \in \mathbb{N}$, the solution of Equation 3.14 can be recursively reconstructed as follows:

$$\begin{cases} M_{t+1} = P_t X_{t+1}^\top (\lambda I_{n_{t+1}} + X_{t+1} P_t X_{t+1}^\top)^{-1} \\ P_{t+1} = P_t - M_{t+1} X_{t+1} P_t \\ W_{t+1} = M_{t+1}(Y_{t+1} - X_{t+1} W_t) \end{cases} \tag{3.15}$$

where $P_0 = \frac{1}{\lambda} I_p$ and $W_0 = 0_{p \times C}$, where $0_{p \times C}$ is a $p \times C$ matrix of zeros. The equivalent optimal solution W^* is $W^* = W_T$.

While Equation 3.15 offers an effective solution for incremental learning, it is not suitable for privacy-constrained applications where data cannot be centralized or communicated. The reason is that Equation 3.15 requires direct access to the data batches (X_{t+1}, Y_{t+1}) in order to compute the updated matrices M_{t+1}, P_{t+1}, and W_{t+1}. This direct access would violate the fundamental principle of decentralized and privacy-constrained environments, where data must remain local.

3.2.4 The kernel trick and kernel ridge regression

The kernel trick [112, 113] is a technique in machine learning that allows algorithms to operate in a higher-dimensional or even infinite-dimensional feature space without the need to transform data points into that space explicitly. Instead of performing this transformation, it directly computes the inner product of the data points in the transformed feature space by using a kernel function. This approach saves both computational effort and memory.

Suppose that exists a mapping function $\phi : \mathcal{X} \to \mathcal{X}'$ to map data from an input space $\mathcal{X}$ to a higher-dimensional feature space $\mathcal{X}'$. In the feature space $\mathcal{X}'$, a model can achieve linear separation of the data. However, explicitly computing $\phi(x),\ x \in \mathcal{X}$ can be costly or infeasible, especially in high-dimensional or infinite-dimensional spaces. Instead, it can be advantageous to use a kernel function $\texttt{ker} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\texttt{ker}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ for some $x_i, x_j \in \mathcal{X}$, to compute the inner products of transformed vectors, as this eliminates the need for explicit mapping when the input samples only appear in inner product form.

The RR objective function can be rearranged to express the inner products explicitly. This approach eliminates the need to use the mapping function ϕ to transform the inputs into $\mathcal{X}'$ to operate in a higher-dimensional feature space, thereby allowing for the application of the kernel trick. Indeed, let $r := (X^\top)^{-1}W$, assuming $X^\top$ is invertible. Then, the RR objective $F(W; X, Y)$ can be reformulated as follows[2]:

$$\begin{aligned} F(W; X, Y) &= \|Y - XW\|^2 + \lambda \|W\|^2 = \\ &= \left\|Y - XX^\top r\right\|^2 + \lambda \left\|X^\top r\right\|^2 = \\ &= \left\|Y - XX^\top r\right\|^2 + \lambda \langle X^\top r, X^\top r\rangle = \\ &= \left\|Y - XX^\top r\right\|^2 + \lambda (X^\top r)^\top (X^\top r) = \\ &= \left\|Y - XX^\top r\right\|^2 + \lambda r^\top XX^\top r. \end{aligned} \tag{3.16}$$

If $\mathtt{K} \in \mathbb{R}^{n \times n}$ is the kernel matrix such that $\mathtt{K}_{ij} = \mathtt{ker}(x_i, x_j)$ for all the inputs in the dataset $\mathcal{D}$, it is possible to substitute $XX^\top$ with $\mathtt{K}$, obtaining the following kernel ridge regression (KRR) problem [114, 115]:

$$r^* = \underset{r \in \mathbb{R}^{n \times C}}{\arg\min} \|Y - \mathtt{K}r\|^2 + \lambda r^\top \mathtt{K} r, \tag{3.17}$$

that admits the following closed-form solution:

$$r^* = (\mathtt{K} + \lambda I_n)^{-1} Y, \tag{3.18}$$

from which it is possible to obtain the optimal parameters W^*:

$$W^* = X^T (\mathtt{K} + \lambda I_n)^{-1} Y. \tag{3.19}$$

Using the linear kernel function $\mathtt{ker}(x_i, x_j) = \langle x_i, x_j \rangle$, this solution is equivalent to Equation 3.14.

A popular choice for the kernel function is the Radial Basis Function (RBF) kernel:

[2] the subscript "2" for the Euclidean norm is omitted here for readability.

$$\texttt{ker}(x_i, x_j) = e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}}, \tag{3.20}$$

where $\sigma \in \mathbb{R}^+$ is the bandwidth hyper-parameter. The RBF kernel implicitly maps the input data to an infinite-dimensional space and results in smooth and continuous decision boundaries [116].

3.2.5 Approximation of kernel ridge regression using random Fourier features

KRR utilizes kernels to implicitly map input data into a higher-dimensional space, enabling the modeling of complex, nonlinear relationships. However, KRR can become computationally expensive, particularly with large datasets, because it requires computing and inverting a kernel matrix, which scales quadratically with the number of data points: conversely to the covariance matrix $XX^\top$ of (linear) RR, which has a space complexity of $\mathcal{O}(p^2)$, the space complexity of the kernel matrix is $\mathcal{O}(n^2)$. This is a prohibitive issue, as for large datasets, where $n \gg p$, it becomes impractical to store the kernel matrix or to compute the exact KRR solution. To mitigate this issue, an approximation of KRR is often employed, providing a more scalable solution.

One of such approximation methods is random Fourier features (RFF) KRR [97]. With the RFF approximation, it is possible to explicitly map the inputs using an appropriately defined mapping function ϕ to approximate the RBF kernel. Specifically:

$$\phi(x) = \sqrt{\frac{2}{P}} \begin{bmatrix} \cos(\omega_1^\top x + b_1) \\ \cos(\omega_2^\top x + b_2) \\ \vdots \\ \cos(\omega_P^\top x + b_P) \end{bmatrix}, \tag{3.21}$$

where $\omega \in \mathbb{R}^{p \times P}$ is a random matrix whose rows ω_i are random vectors drawn from a Normal distribution $\mathcal{N}\left(0, \frac{2}{\sigma} I_P\right)$, $\sigma \in \mathbb{R}^+$ is the bandwidth of the RBF kernel, and $b \in \mathbb{R}^P$ is a random bias vector whose values are randomly drawn from a uniform distribution $\mathcal{U}(0, 2\pi)$, and $P \in \mathbb{N}, P > p$.

Using the RFF approximation, it is possible to compute the optimal W^* solution in a P-dimensional feature space using the same formulation of the RR solution in Equation 3.14:

$$W^* = (\phi(X)^\top \phi(X) + \lambda I_P)^{-1} \phi(X)^\top Y, \tag{3.22}$$

where $\phi(X) \in \mathbb{R}^{n \times P}$ is the matrix of the stacked feature mappings $\phi(x)$. This formulation involves calculating and inverting a $P \times P$ matrix. The approximation of the KRR solution using the RBF kernel becomes increasingly accurate as P approaches infinity. Consequently, the RFF approximation provides a trade-off between the accuracy of the KRR solution and the increased spatial and computational complexity necessary to store the RFF and calculate the explicit mapping of each input to the higher feature space.

3.3 Client drift and classifier data recency bias in Federated Learning

Training models in federated settings introduces unique challenges, as introduced in Section 2.4. These challenges are particularly significant, especially in cross-device scenarios, where thousands or even millions of devices participate, each with limited availability, resources, and heterogeneous data distributions [14].

The authors of [16] highlight that statistical heterogeneity can lead to a phenomenon known as *client drift*. This issue occurs when local updates become biased toward the specific data of individual clients. As a result, the local model updates generated by clients during training can significantly diverge from one another. This divergence primarily arises from variations in the data distributions of the clients, which undermines the effectiveness of simple aggregation strategies, such as that used in `FedAvg` [13], which is shown in Equation 2.6. Consequently, this divergence negatively impacts convergence speed, communication efficiency, and overall model performance.

Several strategies have been proposed in the literature to address this issue. One approach is to mitigate the divergence of local models by using regularization techniques [15, 62] or stochastic variance reduction methods [16, 117] to better

align local model updates. Other methods utilize the history of previous updates by incorporating momentum or adaptive optimizers during server aggregation [118, 119, 18]. Additionally, some strategies introduce client-side momentum [120, 77] to reduce client drift and steer local updates toward the direction of the global model. Some works aim to replicate the behavior of models trained on i.i.d. data by combining stochastic variance reduction with client-side momentum [77].

Statistical heterogeneity and client drift affect the different layers of the models with various severity levels. In particular, [92] shows that clients' bias towards local data distributions is more pronounced in the deeper layers of the model, particularly in the last one, the prediction head. Other studies have noted that biased classifiers and misaligned features create a harmful cycle [121, 93]. For instance, [122] suggests that learning the feature extractor and the classifier in two separate phases may be beneficial in FL, as observed in other fields [107, 123]. Moreover, [91] points out that discriminative features are essential for effectively training models, as convergence speed improves when gradients are better aligned.

Recent research has highlighted the advantages of using pre-trained feature extractors in FL to reduce the negative impacts of data heterogeneity and speed up convergence [91]. However, the classifier, which is often trained from scratch, remains unstable in cross-device scenarios due to partial participation from clients and non-i.i.d. data. This can create a detrimental cycle where a weak classifier adversely affects the updates of the feature extractor, leading to poorer feature maps being provided to the classifier and deteriorating the training process [121, 93]. This issue is further complicated in the final prediction layer by the *data recency bias* (or catastrophic forgetting) [94, 124–126], which occurs when the model is overly influenced by the most recently active clients. This is particularly problematic in cross-device FL characterized by non-i.i.d. and class-imbalanced data [127, 128, 75, 129].

Several studies [91, 92, 130] have proposed solutions to mitigate the classifier data recency bias by retraining only the classifier on the server while using virtual features. More recently, [93] introduced a fixed synthetic classifier based on the simplex geometry of the logit space induced by neural collapse [131]. Additionally, [132] proposed `FedNCM`, an algorithm that constructs a Nearest Class Mean classifier to avoid gradient updates by constructing a global classifier using class prototypes computed from local client data.

3.4 Federated Recursive Ridge Regression (Fed3R)

To tackle the challenges of data recency bias and client drift while maximizing the advantages of pre-trained feature extractors in FL, here is introduced Federated Recursive Ridge Regression (Fed3R). This novel method constructs a classifier that is immune to statistical heterogeneity. Indeed, Fed3R utilizes an online formulation of RR to compute the classifier efficiently using a closed-form solution, leveraging feature maps generated by a pre-trained feature extractor.

In Fed3R, each client computes and sends its local RR statistics to the server, which then aggregates them to obtain a classifier equivalent to the centralized RR solution presented in Equation 3.14. This approach ensures exact aggregation, requires only a single communication round per client, and is immune to variations in client sampling order and statistical heterogeneity. Additionally, Fed3R circumvents the training process by fixing the parameters of the feature extractor and computing optimal parameters for the classifier using a closed-form solution, thereby preventing client drift.

Let $\mathcal{C}_k$ be the set of the C_k classes present in client $k \in \mathcal{K}$, such that $\bigcup_{k\in\mathcal{K}} \mathcal{C}_k = \mathcal{C}$. Here and in the next chapter we assume that the model f, parameterized by θ, can be decomposed into a feature extractor $\varphi : \mathcal{X} \to \mathcal{Z}$, and a classifier $\psi : \mathcal{Z} \to \mathcal{Y}$, *i.e.*, $f = \psi \circ \varphi$, and that φ is parameterized by θ_φ, ψ is parameterized by θ_ψ, and $\theta_\varphi \cup \theta_\psi = \theta$. Fed3R leverages fixed, pre-trained feature extractor parameters θ_φ to map the input data into the latent feature space $\mathcal{Z} \subseteq \mathbb{R}^d$, $d \in \mathbb{N}$. The Fed3R objective is to determine the optimal parameters θ_ψ^* for the classifier ψ in a federated setting where clients collaboratively contribute to the solution.

First, observe that the RR solution presented in Equation 3.14 could be rewritten using the latent space $\mathcal{Z}$ as the input space for RR:

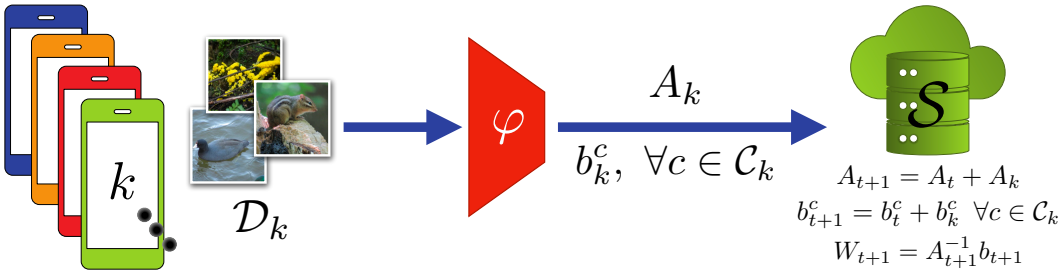


Figure 3.1: The `Fed3R` algorithm in a nutshell. Each client $k \in \mathcal{K}$ extracts its local RR statistics A_k and b_k^c, $\forall c \in \mathcal{C}_k$, and shares them with the server, where they are aggregated, and the server updates the `Fed3R` classifier accordingly.

$$
\begin{aligned}
W^* &= \left(\sum_{(x,y)\in\mathcal{D}} \varphi(x)\varphi(x)^\top + \lambda I_d \right)^{-1} \sum_{(x,y)\in\mathcal{D}} \varphi(x) e_y^\top = \\
&= \left(\sum_{i=1}^{n} \varphi(X_i)\varphi(X_i)^\top + \lambda I_d \right)^{-1} \sum_{i=1}^{n} \varphi(X_i) Y_i^\top = \\
&= (Z^\top Z + \lambda I_d)^{-1} Z^\top Y = \\
&= (A + \lambda I_d)^{-1} b,
\end{aligned} \tag{3.23}
$$

where $A = Z^\top Z \in \mathbb{R}^{d\times d}$ is the covariance matrix of samples in the latent space, $Z \in \mathbb{R}^{n\times d}$ is the matrix of mapped input samples with rows $Z_i = \varphi(X_i)$ and $b = Z^\top Y \in \mathbb{R}^{d\times C}$ is the sum of the samples mapped into the latent space aggregated per class, as e_y represents the one-hot encoding vector for the class y.

The complete `Fed3R` algorithm is shown in Algorithm 3 and pictorically described in Figure 3.1. The core idea of `Fed3R` is to consider each local dataset $\mathcal{D}_k$ as a separate batch of data and recursively update the ridge regression solution. However, a direct application of the recursive RR formulation in Equation 3.15 would require clients to share their raw data with the server[3], violating privacy constraints.

To address this challenge, `Fed3R` leverages the structure of the A and b matrices that can be cumulatively updated without re-computing them from scratch [100, 133]. Indeed, these matrices can be expressed as the sum of individual contributions from

[3]Specifically, Equation 3.15 requires direct access to X_t and Y_t.

Algorithm 3 - Federated Recursive Ridge Regression (Fed3R).

Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$
$\mathcal{S}$ initializes $A_0 = \lambda I_d$ and $b_0 = 0_{d \times C}$
// run on client k
for each client $k \in \mathcal{K}$ in parallel **do**
  $Z_k = \varphi(X_k), \quad A_k = Z_k^\top Z_k$
  $b_k^c = \sum_{(x,y=c) \in \mathcal{D}_k} \varphi(x), \forall c \in \mathcal{C}_k$
  Send A_k and $b_k^c \ \forall c \in \mathcal{C}_k$ to the server: $\mathcal{M}_{k \to \mathcal{S}} = \{A_k, b_k^c \ \forall c \in \mathcal{C}_k\}$
end for
// run on the server $\mathcal{S}$
Every time $\mathcal{S}$ receives A_k and b_k:
  $A_{t+1} = A_t + A_k$
  $b_{t+1}^c = b_t^c + b_k^c \ \ \forall c \in \mathcal{C}_k$
  $W_{t+1} = A_{t+1}^{-1} b_{t+1}$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \ \ \forall c \in \mathcal{C}$

each client's local dataset $\mathcal{D}_k$:

$$A = \sum_{(x,y) \in \mathcal{D}} \varphi(x)\varphi(x)^\top = \sum_{k \in \mathcal{K}} \sum_{(x,y) \in \mathcal{D}_k} \varphi(x)\varphi(x)^\top = \sum_{k \in \mathcal{K}} Z_k^\top Z_k = \sum_{k \in \mathcal{K}} A_k, \quad (3.24)$$

$$b = \sum_{(x,y) \in \mathcal{D}} \varphi(x) e_y^\top = \sum_{k \in \mathcal{K}} \sum_{(x,y) \in \mathcal{D}_k} \varphi(x) e_y^\top = \sum_{k \in \mathcal{K}} Z_k^\top Y_k = \sum_{k \in \mathcal{K}} b_k, \quad (3.25)$$

where Z_k, A_k and b_k are defined as Z, A, and b but for the local dataset $\mathcal{D}_k$ of client $k \in \mathcal{K}$. Notably, Equations (3.24) and (3.25) are true for all the possible partitions of $\mathcal{D}$ into subsets $\mathcal{D}_k$ such that $\mathcal{D} = \bigcup_{k \in \mathcal{K}} \mathcal{D}_k$. Therefore, once all clients have communicated their A_k and b_k statistics to the server, it can compute the Fed3R solution, which is equivalent to the centralized RR solution independently from the particular federated clients' split, ① **making the Fed3R algorithm immune to statistical heterogeneity** and ② **invariant to the order the clients share their statistics with the server**.

Moreover, clients can optimize the transmission of b_k to minimize communication overhead. Indeed, instead of sending the full matrix, which may contain many zero columns associated with classes not present in the local dataset, clients only transmit the non-zero column vectors b_k^c for $c \in \mathcal{C}_k$. Once the server receives the column

vectors b_k^c, it sums them in the corresponding column c of the global b matrix. Once the server $\mathcal{S}$ has received the statistics A_k and b_k^c from all the clients and updates the classifier parameters, ③ **the Fed3R solution is mathematically equivalent to the optimal centralized classifier** W^*. Consequently, ④ **the Fed3R classifier inherits all the properties and guarantees of the** (centralized) **RR classifier** presented in Section 3.2.2.

Finally, to address the class imbalance of the global distribution in realistic scenarios, the final classifier parameters W are normalized by dividing each column by its norm: $W^c \leftarrow W^c / \|W^c\|, \forall c \in \mathcal{C}$. This normalization acts as a reweighing mechanism since, without this normalization, more frequent classes tend to increase the magnitude of the associated values in W. In other words, this scaling increases the importance of under-represented classes.

Differently from gradient-based FL algorithms, **Fed3R** ⑤ **does not rely on common assumptions such as bounded variance of stochastic gradients or smoothness of clients' loss landscapes** [14, 77, 16, 62] making it readily applicable to real-world scenarios. Moreover, conversely to most FL algorithms that require multiple samplings of the same clients, Fed3R only requires each client to share its statistics with the server once, drastically reducing communication costs and accelerating training. Namely, ⑥ **each client is sampled only once**.

3.5 Fed3R with random Fourier features approximation (Fed3R-RF)

Fed3R allows the construction of a linear classifier based on the quality of the feature maps extracted by a pre-trained feature extractor. Therefore, its performance is dependent on the similarity of the dataset distribution on which the feature extractor was originally pre-trained with the distribution of the clients' datasets. Because of this, feature maps generated from the images of the local datasets using the pre-trained feature extractor may not be linearly separated effectively.

A possible solution is to construct a federated version of KRR. However, as already discussed in Section 3.2.5, computing the exact KRR solution could be prohibitive due to the high computational and allocation costs.

Algorithm 4 - Federated Recursive Ridge Regression with random Fourier features approximation of KRR with RBF as the kernel function. (Fed3R-RF).

Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$
RFF normal vectors $\omega \in \mathbb{R}^{d \times D}$, random bias vector $b \in \mathbb{R}^D$
Mapping function ϕ defined as in Equation 3.21
$\mathcal{S}$ initializes $A_0 = \lambda I_D$ and $b_0 = 0_{D \times C}$
// run on client k
for each client $k \in \mathcal{K}$ in parallel **do**
  $\hat{Z}_k = \phi(\varphi(X_k)), \quad A_k = \hat{Z}_k^\top \hat{Z}_k$
  $b_k^c = \sum_{(x,y=c) \in \mathcal{D}_k} \phi(\varphi(x)), \forall c \in \mathcal{C}_k$
  Send A_k and $b_k^c \ \forall c \in \mathcal{C}_k$ to the server: $\mathcal{M}_{k \rightarrow \mathcal{S}} = \{A_k, b_k^c \ \forall c \in \mathcal{C}_k\}$
end for
// run on the server $\mathcal{S}$
Every time $\mathcal{S}$ receives A_k and b_k:
  $A_{t+1} = A_t + A_k$
  $b_{t+1}^c = b_t^c + b_k^c \ \ \forall c \in \mathcal{C}_k$
  $W_{t+1} = A_{t+1}^{-1} b_t$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \ \ \forall c \in \mathcal{C}$

To overcome this limitation, Federated Recursive Ridge Regression with RFF (Fed3R-RF) exploits RFF approximation, explicitly mapping the latent feature space to a higher-dimensional feature space with dimensionality $D > d$. This explicit mapping allows maintaining the same structure as the Fed3R algorithm and the same properties presented in Section 3.4. The complete Fed3R-RF algorithm is shown in Algorithm 4. Notably, Fed3R-RF inherits all the ①-⑥ properties of Fed3R.

3.6 Fed3R privacy properties and synchronous Fed3R (Fed3R-Sync)

Fed3R constructs a classifier by utilizing local client data. Instead of calculating gradient updates, it computes local statistics, specifically the local covariance matrix A_k and vectors b_k^c. These computations are performed on the local device, and only the aggregated statistics are communicated to the server, ensuring that raw client data is never transmitted. As a result, Fed3R fully complies with the standard FL

framework [13, 15], which requires the server never to access clients' local private data. This approach significantly reduces privacy and security risks compared to distributed learning solutions, where client data may be accessed.

Additionally, similar to the `Fed3R` method, prototype-based FL techniques transfer aggregated statistics (the prototypes) rather than model updates between the server and the clients. As noted in [134, 135], this approach enhances privacy compared to other FL algorithms. Specifically, [134] (`FedProto`) emphasizes that sharing prototypes improves privacy because they are obtained by averaging the low-dimensional latent representations of samples from the same class, an irreversible process. Consequently, attackers cannot reconstruct raw data from these prototypes.

In `Fed3R`, clients share only the aggregated statistics $(A_k; b_k^c \ \forall c \in \mathcal{C}_k)$ with the server. Here, A_k represents the sum of the outer products of all latent feature vectors computed from the local dataset, while each vector b_k^c is the sum of all latent feature vectors belonging to class c. This means that b_k^c is the non-averaged prototype for client k for class c:

$$A_k = \sum_{(x,y)\in\mathcal{D}_k} \varphi(x)\varphi(x)^\top, \quad b_k^c = \sum_{(x,y=c)\in\mathcal{D}_k} \varphi(x) \ \forall c \in \mathcal{C}_k. \tag{3.26}$$

Therefore, `Fed3R` offers the same privacy benefits as `FedProto` [134]. In terms of privacy risks, it is, at worst, comparable to other FL algorithms [13, 16, 18, 62] that prioritize other important aspects of the FL scenario, such as accuracy, robustness to heterogeneity, and efficiency, instead of focusing on privacy guarantees.

Additionally, it is important to highlight that a significant amount of research focuses on the security and privacy aspects of FL systems alone [136, 137], including the development of privacy-enhancing techniques and countermeasures against malicious attacks. The privacy of any FL algorithm, including `Fed3R`, can be significantly improved by integrating it with methods such as the Secure Aggregation protocol [138], Differential Privacy [139, 140], homomorphic encryption [141], and secure multi-party computation [142], among others.

One may still argue that revealing the classes to the server could constitute a potential privacy risk. In `Fed3R`, each client needs to share the b_k^c statistics with the server, which needs to be aware of the classes c for all the received vectors b_k^c to sum them appropriately into the global matrix b. However, in some applications, revealing

Algorithm 5 - Synchronous Fed3R (Fed3R-Sync)

Def: Y_k := matrix of all the stacked one-hot encoding labels of client k
Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$
$\mathcal{S}$ initializes $A_0 = \lambda I_d$ and $b_0 = 0_{d \times C}$
Divide clients in J disjoint groups of clients $\mathcal{K}_j \subseteq \mathcal{K}$
for each group of clients $\mathcal{K}_j$ in parallel **do**
  // run on client k
  for each client $k \in \mathcal{K}_j$ in parallel **do**
    $Z_k = \varphi(X_k), \quad A_k = Z_k^\top Z_k, \quad b_k = Z_k^\top Y_k$
  end for
  Aggregate with secure aggregation [138]:
    $A_j = \sum_{k \in \mathcal{K}_j} A_k, \quad b_j = \sum_{k \in \mathcal{K}_j} b_k$
  Send A_j and b_j to the server: $\mathcal{M}_{\mathcal{K}_j \rightarrow \mathcal{S}} = \{A_j, b_j\}$
end for
// run on the server $\mathcal{S}$
Every time $\mathcal{S}$ receives A_j and b_j:
  $A_{t+1} = A_t + A_j, \quad b_{t+1} = b_t + b_j$
  $W_{t+1} = A_{t+1}^{-1} b_t$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \quad \forall c \in \mathcal{C}$

this information is a potential privacy issue [137]. Nonetheless, any classifier good enough is already sufficient to reveal the classes of the clients [85]. Therefore, the potential Fed3R privacy issues related to disclosing the local classes with the server are comparable with the privacy issues of any FL algorithm where the clients share the classifier with the server, such as classical algorithms like FedAvg [13], Scaffold [16], or FedDyn [62].

Nevertheless, it is possible to further enhance the privacy of Fed3R in this sense, at the expense of slightly additional communication and computational costs, by requiring the clients to synchronize together, transforming the Fed3R algorithm into a synchronous one - a variant of Fed3R which in this manuscript is named "synchronous Fed3R" (Fed3R-Sync). Differently from Fed3R, in Fed3R-Sync, all the clients compute the entire matrices $b_k \in \mathbb{R}^{d \times C}$ instead of the single column vectors b_k^c, with a slight increase in communication and computation costs. Then, thanks to secure aggregation [138], the server does not have access to the individual clients' statistics. Additionally, similar to how RFF [97] are exploited in Fed3R-RF to map the latent space into a feature space with higher linear separability, it is possible to employ the

Algorithm 6 - Synchronous Fed3R-RF (Fed3R-RF-Sync)

Def: Y_k := matrix of all the stacked one-hot encoding labels of client k
Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$
RFF normal vectors $\omega \in \mathbb{R}^{d \times D}$, random bias vector $b \in \mathbb{R}^D$
Mapping function ϕ defined as in Equation 3.21
$\mathcal{S}$ initializes $A_0 = \lambda I_D$ and $b_0 = 0_{D \times C}$
Divide clients in J disjoint groups of clients $\mathcal{K}_j \subseteq \mathcal{K}$
for each group of clients $\mathcal{K}_j$ in parallel **do**
  // run on client k
  for each client $k \in \mathcal{K}_j$ in parallel **do**
    $\hat{Z}_k = \phi(\varphi(X_k)), \quad A_k = \hat{Z}_k^\top \hat{Z}_k, \quad b_k = \hat{Z}_k^\top Y_k$
  end for
  Aggregate with secure aggregation [138]:
    $A_j = \sum_{k \in \mathcal{K}_j} A_k, \quad b_j = \sum_{k \in \mathcal{K}_j} b_k$
  Send A_j and b_j to the server: $\mathcal{M}_{\mathcal{K}_j \to \mathcal{S}} = \{A_j, b_j\}$
end for
// run on the server S
Every time $\mathcal{S}$ receives A_j and b_j:
  $A_{t+1} = A_t + A_j, \quad b_{t+1} = b_t + b_j$
  $W_{t+1} = A_{t+1}^{-1} b_t$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \quad \forall c \in \mathcal{C}$

RFF also with the synchronous version of Fed3R. In this manuscript, the synchronous Fed3R-RF version is called Fed3R-RF-Sync. The complete Fed3R-Sync and Fed3R-RF-Sync algorithms are presented in Algorithms 5 and 6 respectively.

With Fed3R-Sync, the server can only know that a group of clients $\mathcal{K}_j$ has samples belonging to some specific classes by looking at the non-zero columns of the received matrices b_j, but cannot know how these classes are distributed among the clients $k \in \mathcal{K}_j$. Therefore, the larger these $\mathcal{K}_j$ groups are, the more difficult it is to understand the local clients' class distributions.

Moreover, secure aggregation even enhances the privacy for the covariance matrices A_k, which are asynchronously shared in the Fed3R algorithm. In addition, collecting the statistics from these groups of clients sequentially is similar to the concept of round for classical FL algorithms. Thus, the index t in Algorithm 5 can assume this facet.

Since the Fed3R-Sync solution is equivalent to the Fed3R solution, the two algorithms share the same properties ①-⑥ discussed in Section 3.4, and the same performances of the corresponding asynchronous versions. Finally, conversely to gradient-based FL methods, ⑦ **Fed3R-Sync is guaranteed to always achieve maximum accuracy in $\lceil \frac{K}{\kappa} \rceil$ rounds**. Consequently, a higher participation rate means faster convergence of Fed3R. This is not guaranteed, in general, for gradient-based FL algorithms.

3.7 Fed3R parameters for robust classifier initialization

While Fed3R provides a simple and efficient approach for learning a classifier in a federated environment, successfully addressing the challenges posed by statistical heterogeneity, its effectiveness ultimately depends on the quality of the pre-trained feature extractor. To improve the model's adaptability and performance for the target task, it is possible to incorporate a federated fine-tuning stage that utilizes gradient-based FL algorithms using the Fed3R parameters as a robust initialization for the classifier. Since the Fed3R classifier is already the optimal solution for the RR problem with the original feature extractor parameters, the model is fine-tuned using the CE loss, which typically provides better performance for classification tasks [102]. Additionally, to align the entropy of the predictions' distribution, the model is calibrated by tuning the temperature of the softmax function (see Equation 3.2). Fed3R parameters provide a stable initialization that can mitigate client drift and destructive interference during aggregation in heterogeneous federated settings, which can arise from inconsistencies in local updates and lead to classifier divergence, breaking the detrimental cycle described in [121].

In this manuscript, three distinct fine-tuning strategies are explored. The first possibility is to fine-tune both the feature extractor and the classifier. This strategy is suitable when both can benefit from further adaptation to the target task. Another option involves fine-tuning only the classifier while keeping the pre-trained features fixed. This approach is appropriate when the pre-trained features are already robust and general enough for the target task. Finally, a third option consists of fine-tuning only the feature extractor while keeping the Fed3R classifier fixed. Section 3.9.9

shows that this strategy effectively mitigates destructive interference, especially in cross-device scenarios with high statistical heterogeneity, where the classifier is often the most susceptible to divergent updates [92, 93].

3.8 Existing literature on ridge regression for distributed and Federated Learning

Recently, some works started to investigate the applicability of RR in the context of FL. In [143], an RR model is trained in a federated scenario with only two clients via knowledge distillation. Conversely, the `Fed3R` algorithms allow an indefinite number of clients to participate in the federation. Least square predictors have also been applied to the vertical FL [144, 145] and federated transfer learning setting [146]. On the contrary, the `Fed3R` algorithms are designed for the horizontal FL setting [57] to address the major issues of client drift and statistical heterogeneity. Additionally, RR variants have been proposed in the distributed learning and optimization fields [147–149]. Still, these settings are significantly different from FL since they lack privacy requirements and usually assume i.i.d. data.

3.9 Experiments

This section introduces the experiments to evaluate performance, communication, and computation costs of the `Fed3R` algorithms. Specifically:

- Section 3.9.1 presents the datasets, the taxonomy, and the implementation details on how the simulations have been performed.
- Section 3.9.2 describes how the communication and computation costs have been calculated.
- Section 3.9.3 provides a comparison between `Fed3R`, `Fed3R-RF` and `FedNCM` [132], another closed-form classifier for FL.
- Section 3.9.4 empirically shows that `Fed3R-Sync` is immune to statistical heterogeneity.

- Section 3.9.5 discusses two possible sampling strategies for `Fed3R-Sync` and shows how its speed of convergence is proportional to the number of clients participating in the training in the various rounds.
- Section 3.9.6 presents results showing how `Fed3R-Sync` and `Fed3R-RF-Sync` are much faster than FL baselines.
- Section 3.9.7 empirically compares the communication costs of `Fed3R` with the ones of `Fed3R-Sync`.
- Section 3.9.8 compares `Fed3R` with other FL baselines.
- Section 3.9.9 shows that `Fed3R` parameters can enhance the performance of other FL algorithms, effectively countering the negative consequences of statistical heterogeneity if used as initialization for the parameters of the classifier.
- Section 3.9.10 empirically demonstrates that `Fed3R` can be combined with other orthogonal methods to further improve performance and that `Fed3R` can improve the results also in the presence of domain imbalance.
- Section 3.9.11 shows how `Fed3R` can be repurposed to evaluate the quality of the feature extractor in FL.
- Section 3.9.12 shows the effects of removing the normalization from the `Fed3R` classifier.

3.9.1 Datasets, taxonomy and implementation details

Table 3.1 presents the core information regarding image classification datasets for FL employed in the experiments of this chapter. All these datasets provide an external dataset $\mathcal{D}_{\text{test}}$ for the evaluation, inaccessible during training, representative of all the local data distributions of the clients $k \in \mathcal{K}$.

Google Landmarks and iNaturalist are real-world, large-scale datasets for cross-device FL since they count thousands of clients and classes with skewed data distributions and thousands of distinct classes. For these two datasets, the splits adopted are the ones originally introduced by [74]. Specifically, for Google Landmarks, all the experiments are performed in the only available split, the Users-160k

Table 3.1: Summary of the information of the image classification datasets for FL used in the experiments presented in this chapter.

Dataset	Split	Avg. samples per client	K	C	Tot. number of samples
Google Landmarks [150, 74]	Users-160k	119.9	1262	2028	151373
iNaturalist [73, 74]	Users-120k	13.0	9275	1203	120300
	Geo-100	33.4	3606	1203	120300
	Geo-300	99.6	1208	1203	120300
	Geo-1k	326.9	368	1203	120300
	Geo-3k	889.7	135	1203	120300
Cifar100 [151, 118]	$\alpha = 0.0$	500	100	100	50000
	$\alpha = 0.1$	500	100	100	50000
	$\alpha = 0.5$	500	100	100	50000
	$\alpha = 1.0$	500	100	100	50000
	$\alpha = 10.0$	500	100	100	50000
	$\alpha = 1000.0$	500	100	100	50000

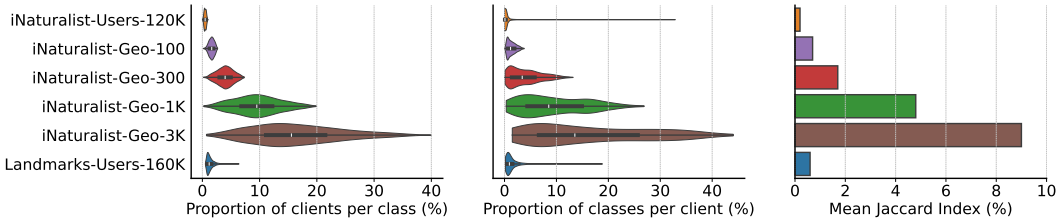


Figure 3.2: Statistical heterogeneity for the splits of iNaturalist and Google Landmarks, using three different metrics. On the left: distribution of the proportion of clients per class (relative to the total number of classes). In the middle: distribution of the proportion of classes per client (relative to the total number of clients). On the right: The Mean Jaccard Index. All values are presented as percentages.

split, while for iNaturalist, most of the experiments are performed on the Users-160k split, as it provides the largest number of clients (~10k clients, ten times more than Google Landmarks-Users-160k, which counts ~1k clients). Additionally, experiments on relatively low heterogeneous settings like iNaturalist-Geo-1k and iNaturalist-Geo-3k are included. In the following, depending on the context, "Google Landmarks" may implicitly refer to Google Landmarks-Users-160k, and "iNaturalist" to iNaturalist-Users-120k or to all or part of the considered iNaturalist splits.

Figure 3.2 shows the different levels of heterogeneity of the iNaturalist and Google Landmarks splits, comparing the proportion of the clients per class, the proportion of the classes per client, and the Mean Jaccard Index [152, 153] computed across the

class distributions of all the couples of clients as follows:

$$\bar{J} = \frac{1}{K^2} \sum_{k \in \mathcal{K}} \sum_{h \in \mathcal{K}} \frac{\mathcal{C}_k \cap \mathcal{C}_h}{\mathcal{C}_k \cup \mathcal{C}_h}. \tag{3.27}$$

The Mean Jaccard Index $\bar{J}$ quantifies the statistical heterogeneity by measuring the overlap between the class distributions of different client pairs. A higher $\bar{J}$ indicates a greater similarity between the client pairs' class distributions, while a lower value suggests more statistical heterogeneity.

In addition to Google Landmarks and iNaturalist, experiments on the simpler, less heterogeneous Cifar100 dataset on the Dirichlet splits proposed by [118] are performed. The level of statistical heterogeneity is controlled by a Dirichlet parameter $\alpha \in \mathbb{R}^+$. A lower α is associated with a higher level of statistical heterogeneity. The Cifar100 experiments are always implicitly conducted on the $\alpha = 0$ split, except when specified otherwise. In the $\alpha = 0$ split, each client has access to only one class, and the same class is present in only that client.

In all the experiments of this chapter, a MobileNetV2 [71] architecture is employed, and its feature extractor parameters θ_φ have been pre-trained on ImageNet-1k [72]. The training images are augmented with the same data augmentation strategies adopted to pre-train the model. In particular, a random resized crop to a dimension of 224×224 is applied for all the experiments except the Cifar100 experiments, where the images are randomly resized and cropped to 32×32 pixels.

All the experiments are performed with three distinct random seeds using NVIDIA A100-SXM4-40GB GPUs, and the results are provided in the form *mean* ± *std*, where *mean* and *std* are the arithmetic mean and the standard deviation of the results of experiments using the three different random seeds. In the figures, the confidence intervals light regions correspond to the intervals between *mean-std* and *mean+std*.

To empirically investigate the performance and efficacy of the proposed `Fed3R` methods, the experiments are divided into **at most** 2 distinct phases. The experimental phases are defined as follows:

- **Phase 1:** Classifier initialization (ψ`-init`). The classifier weights θ_ψ are initialized with `Fed3R`, `Fed3R-RF`, or `FedNCM` weights. In case the classifier is not initialized (*i.e.,* it is randomly initialized), this is denoted by ✗. If

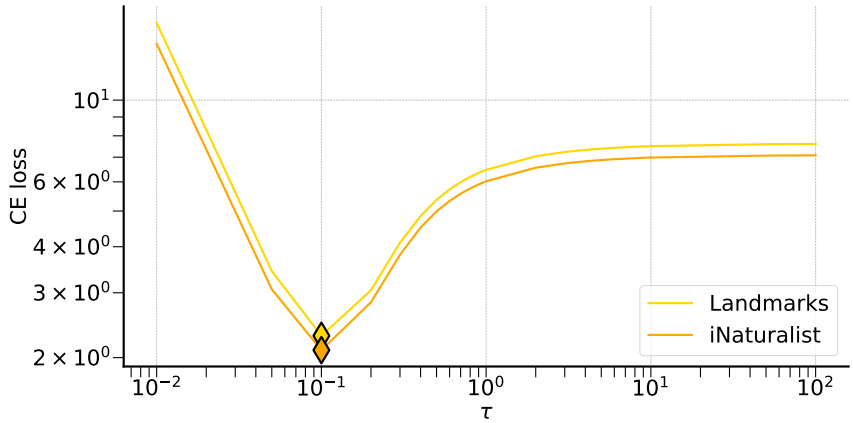


Figure 3.3: Tuning of the best temperature τ for the softmax classifier, after initializing its parameters with the `Fed3R` parameters, on Google Landmarks and iNaturalist. The y-axis shows the average CE loss on the training set. The temperature value that guarantees the lowest CE loss has been chosen for all the experiments. The best value is consistently 0.1.

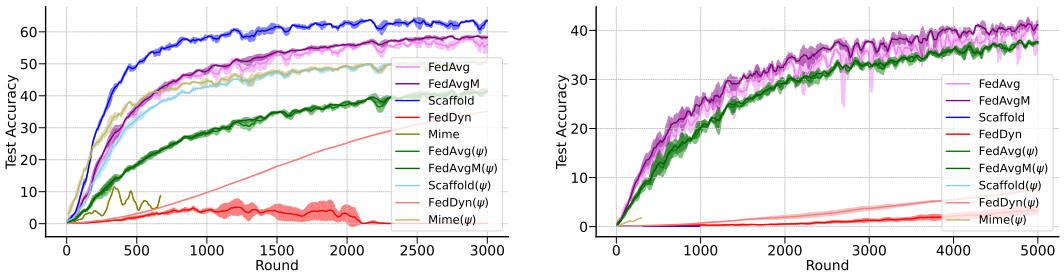


Figure 3.4: Best results of several FL baselines. The Google Landmarks results are presented on the left, while the iNaturalist results are shown on the right.

the classifier is initialized with `Fed3R-RF` weights, the number D of RFF is specified as `Fed3R-RF(`D`)`.

- For the `Fed3R` and `Fed3R-RF` experiments, it has been chosen $\lambda = 0.01$, as this value provided the best centralized RR results on both Google Landmarks and iNaturalist, although other values provided only slightly lower performance.
- For the `Fed3R-RF` experiments, it has been chosen $\sigma = 1000$, as this value provided the best centralized RR results on both Google Landmarks and iNaturalist.

- **Phase 2:** Federated Fine-Tuning (`FT`). This phase corresponds to standard federated training. Specifically, this phase is named *fine-tuning* because the model is always initialized with parameters from a pre-training on another large-scale dataset (*e.g.*, ImageNet [72]).

  - Always 10 clients are sampled per round. By default, the model is trained for 3000 rounds for the Google Landmarks experiments, 5000 rounds for the iNaturalist-Users-120k experiments, 1000 rounds for the iNaturalist-Geo-1k and iNaturalist-Geo-3k experiments, and 1500 rounds for the Cifar100 experiments. Otherwise, the number of rounds is specified by adding `-num_rounds` at the end of the algorithm name (*e.g.,* `FedAvg-1k` means that `FedAvg` is run for 1000 rounds).
  - The chosen local client optimizer is always SGD, with a weight decay of 4×10^{-5}.
  - The local learning rate is 0.1 for the `FT` experiments without any classifier parameter initialization. In case the classifier is initialized, the chosen learning rate is 0.01.
  - The chosen batch size is 50 samples, with 1 local epoch for Google Landmarks and Cifar100 experiments and 5 for the iNaturalist experiments.
  - Figure 3.3 shows the temperature tuning in the Google Landmarks and iNaturalist scenarios for the initialization of the classifier parameters with the `Fed3R` ones. A temperature of $\tau = 0.1$ has been selected for the `FT` experiments on both datasets when the classifier is initialized with `Fed3R` parameters, as this value consistently provided the lowest average training CE loss.
  - If not explicitly indicated, or if specified with the notation `FT(`f`)`, all the parameters of the model are fine-tuned. However, as described in 3.7, it is also possible only to fine-tune the parameters of the classifier or the parameters of the feature extractor. In these cases, the notations `FT(`φ`)` and `FT(`ψ`)`, respectively, are adopted.
  - As per the `FT` algorithms, the following have been chosen: `FedAvg` [13], `FedAvgM` [118], `Scaffold` [16], `Mime` [77], and `FedDyn` [62]. In particular, Figure 3.4 compares the best results of FL algorithms in the Google Landmarks and iNaturalist FL scenarios. All the algorithms

have been tuned using all the possible combinations of server learning rate $\in \{0.1, 1.0\}$, server momentum $\in \{0.0, 0.9\}$, client learning rate $\in \{0.01, 0.1\}$ and client weight decay $\in \{0.0, 4 \times 10^{-5}\}$. The plots in this figure show only the best results. As it is possible to observe, `FedDyn` and `Mime` fail to converge, while `Scaffold` is extremely slow on iNaturalist. Therefore, in the following, only `FedAvg`, `FedAvgM` and `Scaffold` are considered for the Google Landmarks experiments, while only `FedAvg`, `FedAvgM` are considered for the iNaturalist experiments.

3.9.2 Communication and computation costs estimation

The experiments presented in this and the following chapter sometimes include estimating the communication and computation costs necessary to achieve a target accuracy in addition to the performance. This section describes how these costs are estimated.

Total communication costs

The communication is described in terms of the total communication required to achieve any given target accuracy. To find this value, firstly, an estimation of the communication costs for each client in each round is necessary. The total communication costs for each round per client consist of two components: *downstream costs*, which involve communication from the server to the clients (costs of the message $\mathcal{M}^t_{\mathcal{S}\to k}$), and *upstream costs*, which involve communication from the clients to the server (costs of the message $\mathcal{M}^t_{k\to\mathcal{S}}$). These costs can vary depending on the chosen method.

Let c, b, and m represent the dimensions, in number of parameters, of the classifier, the feature extractor, and the whole model, respectively. Since the classifier is a linear layer, its size is equal to the product of the latent feature dimensionality and the number of classes in the dataset; therefore, $c = dC$. Thus, the total model size can be expressed as $m = b + dC$. Additionally, let $\bar{C} = \frac{1}{K}\sum_{k\in\mathcal{K}} C_k$. With these definitions, Table 3.2 summarizes how the downstream and upstream costs per round per client are calculated for each considered FL algorithm, along with any additional costs. In particular, in `Fed3R`, each sampled client does not need to download any information from the server, but it needs to upload the local statistics

Table 3.2: Estimation of communication costs per each sampled client in one round of the FL methods included in the experiments of Chapters 3 and 4. For `Fed3R`, `Fed3R-RF` and `FedNCM` is indicated the communication required for each client, as there is no concept of round and clients are required to communicate with the server only once.

Method	Downstream	Upstream	Total
`Fed3R`	0	$\frac{1}{2}d(d+1) + d\bar{C}$	$\frac{1}{2}d(d+1) + d\bar{C}$
`Fed3R-RF`	0	$\frac{1}{2}D(D+1) + D\bar{C}$	$\frac{1}{2}D(D+1) + D\bar{C}$
`Fed3R-Sync`	0	$\frac{1}{2}d(d+1) + dC$	$\frac{1}{2}d(d+1) + dC$
`Fed3R-RF-Sync`	0	$\frac{1}{2}D(D+1) + DC$	$\frac{1}{2}D(D+1) + DC$
`FedNCM`	0	dC	dC
`FedAvg`	$b + dC$	$b + dC$	$2(b + dC)$
`FedAvg(`φ`)`	b	b	$2b$
`FedAvg(`ψ`)`	dC	dC	$2dC$
`FedAvgM`	$b + dC$	$b + dC$	$2(b + dC)$
`FedAvgM(`φ`)`	b	b	$2b$
`FedAvgM(`ψ`)`	dC	dC	$2dC$
`Scaffold`	$2(b + dC)$	$2(b + dC)$	$4(b + dC)$
`Scaffold(`φ`)`	$2b$	$2b$	$4b$
`Scaffold(`ψ`)`	$2dC$	$2dC$	$4dC$

A_k and b_k^c $\forall c \in \mathcal{C}_k$ to the server. On average, communicating all the b_k^c vectors costs $d\bar{C}$, while communicating the A_k matrix costs $\frac{1}{2}d(d+1)$ since A is a symmetric matrix.

The total costs per round can be calculated by multiplying the costs per round for each client by the number of clients sampled in that round. Instead, for `Fed3R`, `Fed3R-RF`, and `FedNCM` algorithms, are evaluated the cumulative costs for every client that has already shared its statistics with the server. Finally, in all communication cost plots presented later in this chapter, the final values are multiplied by 4 to express the total cost in bytes, as it is assumed that all parameters are stored as FP32 values, which are 4 bytes each.

Average computation costs per client computation

In this manuscript, the average computation costs per client are prioritized over the total computation costs among all clients. This decision stems from the belief that

Table 3.3: Estimation of average computation costs per client per round per client of all the methods included in the experiments of Chapters 3 and 4, expressed in number of parameters.

Method	$\mathcal{T}/n_k$
`Fed3R`	$(F_\varphi + \frac{1}{2}d(d+1) + d\bar{C})$
`Fed3R-RF`	$(F_\varphi + \frac{1}{2}D(D+1) + d\bar{C})$
`Fed3R-Sync`	$(F_\varphi + \frac{1}{2}d(d+1) + dC)$
`Fed3R-RF-Sync`	$(F_\varphi + \frac{1}{2}D(D+1) + dC)$
`FedNCM`	F_φ
`FedAvg`	$3EF_f$
`FedAvg(`φ`)`	$3EF_f$
`FedAvg(`ψ`)`	$3E(F_\varphi + 3F_\psi)$
`FedAvgM`	$3EF_f$
`FedAvgM(`φ`)`	$3EF_f$
`FedAvgM(`ψ`)`	$3E(F_\varphi + 3F_\psi)$
`Scaffold`	$3EF_f$
`Scaffold(`φ`)`	$3EF_f$
`Scaffold(`ψ`)`	$3E(F_\varphi + 3F_\psi)$

Table 3.4: MobileNetV2 [71] forward MFLOPs.

Dataset	F_φ	F_ϕ	$F_\mathcal{M}$
Google Landmarks	332.9	2.6	335.5
iNaturalist	332.9	1.5	334.4
Cifar100	332.9	0.1	333.0

this statistic provides more insightful information regarding the budget required by a FL system developer for their clients.

To estimate the average computation per client, let $\boldsymbol{x}$ denote the number of times a specific client is sampled and $\mathcal{T}$ denote the total average cost per round for a single client. The cumulative average cost $\mathcal{T}_t$ from round 1 to round t is proportional to the expected number of times a specific client is sampled over t rounds, expressed as $\mathbb{E}[\boldsymbol{x}] = t\frac{K}{K}$. Therefore, $\mathcal{T}_t = \mathcal{T}\mathbb{E}[\boldsymbol{x}] = \mathcal{T}t\frac{K}{K}$.

Let F_* and B_* represent the costs of one forward pass and one backward pass of a single image through $*$, respectively. $*$ is one among f, φ, ψ. Analogously to [132], B_* is approximated as $B_* \simeq 2F_*$, and the costs are measured in FLOPs.

Table 3.5: Accuracy (%) of 4 distinct closed-form classifiers.

Dataset	**Fed3R**	**Fed3R-RF(5k)**	**Fed3R-RF(10k)**	**FedNCM**
Google Landmarks	49.6 ± 0.0	53.9 ± 0.0	**56.6** ± 0.0	36.2 ± 0.0
iNaturalist	45.1 ± 0.0	46.8 ± 0.0	**47.6** ± 0.0	32.2 ± 0.0
Cifar100	63.2 ± 0.0	66.2 ± 0.0	**67.5** ± 0.0	51.0 ± 0.0

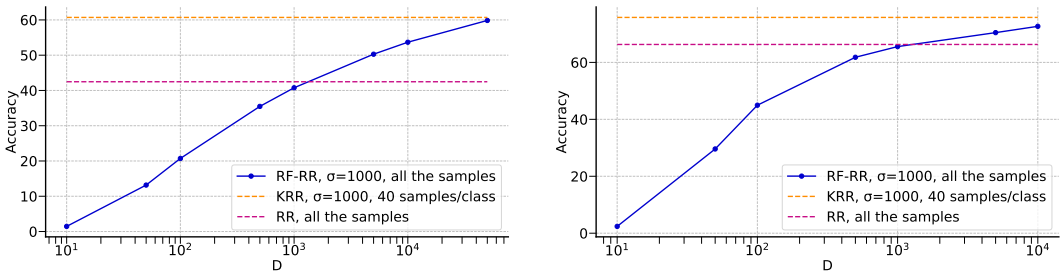


Figure 3.5: Evaluation of RR with RFF (RF-RR) using increasing values of D, alongside KRR and linear RR in the centralized Google Landmarks scenario. Due to computational limitations, a maximum of 40 samples per class was used for KRR. These results do not include the final normalization step for both Fed3R-RF and Fed3R. On the left side, the feature extractor is pre-trained on ImageNet [72], while on the right, it is fine-tuned on Google Landmarks. In both cases, increasing D results in better performance as KRR is better approximated. However, this improvement comes with increased computational and communication costs for Fed3R-RF.

Finally, let E denote the number of local epochs. The estimation of the costs $\mathcal{T}$ per each method is summarized in Table 3.3. In addition, the specific forward FLOPs of the MobileNetV2 model are summarized in Table 3.4.

In particular, In Fed3R, the clients need to forward the input images through the feature extractor once. Then, they have to compute the matrices A_k and the vectors b_k^c, $\forall c \in \mathcal{C}_k$. Since A_k is symmetric, computing A_k costs $\frac{1}{2}n_k d(d+1)$ FLOPs. Instead, computing b_k costs $n_k dC$ FLOPs. Therefore, $\mathcal{T} = n_k(F_\varphi + \frac{1}{2}d(d+1) + dC)$.

3.9.3 Comparison between closed-form classifiers

As previously mentioned in Section 3.3, similar to Fed3R, the FedNCM algorithm [132] constructs a closed-form classifier for FL based on Nearest Class Means. Table 3.5 provides a comparison of the Fed3R and Fed3R-RF classifiers with the FedNCM clas-

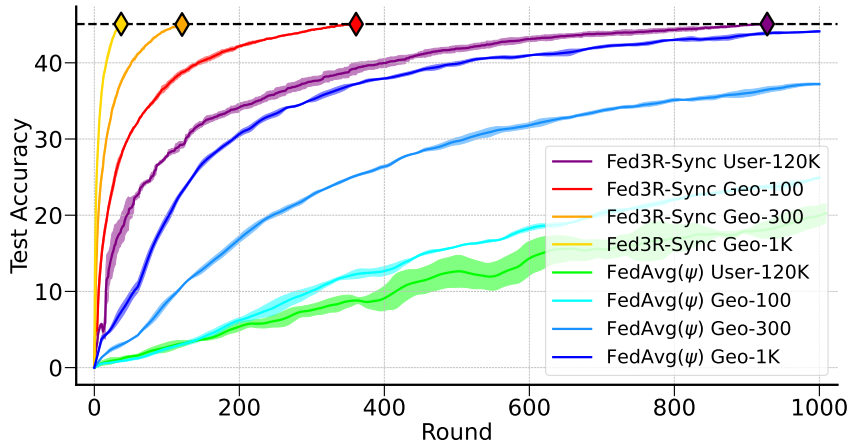


Figure 3.6: Empirical demonstration of the immunity to statistical heterogeneity of the Fed3R algorithm (property ①), using the Fed3R-Sync version. This plot compares Fed3R-Sync with FedAvg(ψ) on different splits of iNaturalist, with groups of $\kappa = 10$ clients, on different levels of statistical heterogeneity (from the split with the lowest statistical heterogeneity to the highest: Geo-1k, Geo-300, Geo-100, Users-120k).

sifier, showing that the Fed3R classifiers consistently provide superior performance compared to FedNCM.

The Fed3R-RF classifier has been tested with two different values for D: $D =$ 5k and $D =$ 10k, providing two different trade-offs between performance and computational cost. Indeed, as discussed in Section 3.5, a higher value of D leads to a better KRR approximation. Because of this, Fed3R-RF(10k) achieves the highest accuracy, as expected, due to its improved approximation of the KRR solution, as it is also shown in Figure 3.5. However, Table 3.3 indicates that the computational costs scale quadratically with D. Consequently, Fed3R-RF(10k) incurs approximately four times the cost of Fed3R-RF(5k).

3.9.4 Fed3R immunity to statistical heterogeneity

Figure 3.6 compares Fed3R-Sync with FedAvg(ψ) on four different iNaturalist splits with different levels of statistical heterogeneity.

Notably, despite the differences in the number of clients and data distribution across the clients across the various splits, Fed3R-Sync always reaches the same accuracy of 45.1% independently from the specific split, confirming that Fed3R-Sync is

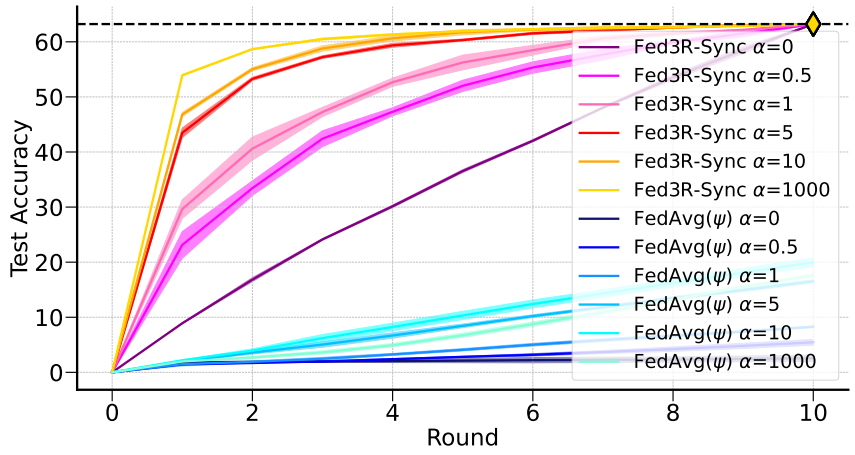


Figure 3.7: `Fed3R-Sync` vs. `FedAvg(`ψ`)` on different splits of Cifar100, with groups of $\kappa = 10$ clients sampled per round, with different levels of statistical heterogeneity (a lower value of α indicates a higher level of statistical heterogeneity).

immune to statistical heterogeneity, as discussed in Section 3.4. Indeed, `Fed3R-Sync` achieves the optimal RR classifier in exactly $\lceil \frac{K}{\kappa} \rceil$ rounds, as already discussed in Section 3.6. Given the equivalence of the `Fed3R-Sync` solution to the `Fed3R` solution, these results empirically prove that `Fed3R` is immune to statistical heterogeneity as well.

On the other hand, `FedAvg(`ψ`)` is much slower, noisier, and less performative than `Fed3R-Sync`, especially in the most heterogeneous iNaturalist-Users-120k split, highlighting how `FedAvg(`ψ`)` is negatively affected from statistical heterogeneity.

Similar considerations can be made by looking at Figure 3.7, which provides results on the Cifar100 dataset partitioned into 100 clients using different values of the Dirichlet parameter α, which controls the severity of the class imbalance among the clients and, by extension, the severity of the statistical heterogeneity.

Finally, Figure 3.8 empirically demonstrates how also `Fed3R-RF-Sync` is immune to statistical heterogeneity and how it guarantees superior accuracy compared to `Fed3R-Sync`.

3.9.5 Sampling strategy and participation rate of `Fed3R-Sync`

Figure 3.9 shows how `Fed3R-Sync` performance is invariant to the number of clients sampled at each round by construction. Therefore, with 10 clients sampled per round

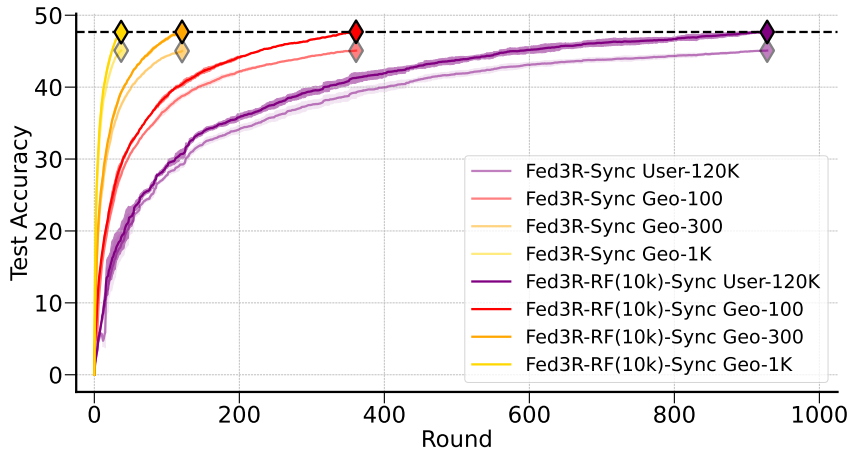


Figure 3.8: Empirical demonstration of the immunity to statistical heterogeneity of the `Fed3R-RF` algorithm (property ①), using the `Fed3R-RF-Sync` version. This plot compares `Fed3R-RF-Sync` (using 10k RFF) with `Fed3R-Sync` on different splits of iNaturalist, with groups of $\kappa = 10$ clients, on different levels of statistical heterogeneity (from the split with the lowest statistical heterogeneity to the highest: Geo-1k, Geo-300, Geo-100, Users-120k).

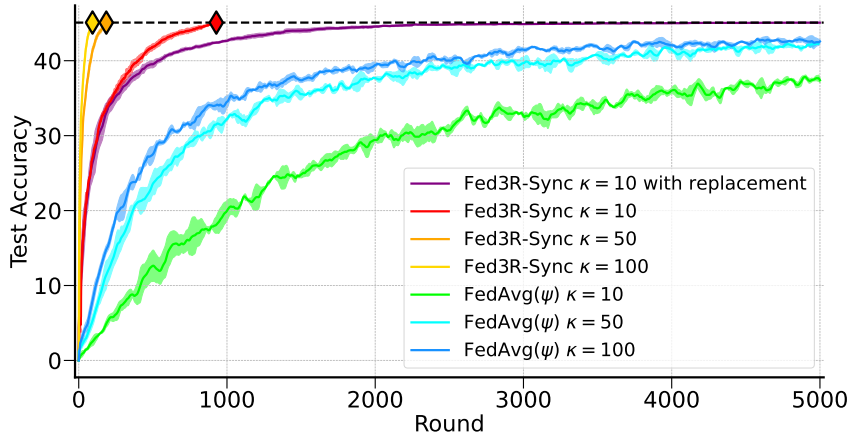


Figure 3.9: Comparison between `Fed3R-Sync` and `FedAvg(`ψ`)` using three different participation rates and two sampling strategies (*without replacement* if not differently specified), on the iNaturalist dataset.

and a total of 1262 clients for Google Landmarks and 9275 clients for iNaturalist, `Fed3R` always needs exactly 127 and 928 rounds to converge, respectively, which is an important advantage compared to traditional FL methods that benefit from multiple passes over the data.

Table 3.6: Average number of rounds needed to sample a given percentage of clients in the corresponding settings when clients are sampled *with replacement*, on different FL scenarios with different levels of statistical heterogeneity. Simulations were run 1k times and then averaged. The problem of finding the expected number of rounds necessary to uniformly sample a given percentage of distinct clients with replacement from a pool of available clients can be seen as an instance of Batch Coupon Collector's Problem [154–156].

Dataset	K	κ	**Participation Rate** (%)	**25%**	**50%**	**75%**	**100%**
iNaturalist	9275	10	0.1	267 ± 2	643 ± 5	1286 ± 12	9020 ± 1189
		20	0.2	134 ± 1	322 ± 3	643 ± 6	4494 ± 596
		50	0.5	54 ± 1	129 ± 1	257 ± 2	1809 ± 247
Google Landmarks	1262	10	0.8	37 ± 1	88 ± 2	175 ± 4	970 ± 155
		20	1.6	19 ± 1	44 ± 1	87 ± 2	483 ± 79
		50	4.0	8 ± 0	18 ± 1	35 ± 1	191 ± 32
Cifar100	100	10	10	3 ± 0	7 ± 1	14 ± 1	50 ± 12
		20	20	2 ± 0	4 ± 0	7 ± 1	24 ± 5
		50	50	1 ± 0	1 ± 0	3 ± 0	8 ± 2

As a worst-case analysis, Figure 3.9 also shows that `Fed3R-Sync` always guarantees better performance than `FedAvg(`ψ`)` in all the training rounds, even if sampling the clients in every round *with replacement*, although `Fed3R-Sync` does not have any advantages in sampling the same clients more times. Even in this case, `Fed3R-Sync` performance has already reached an accuracy near the optimal after only 1.5k rounds. Therefore, `Fed3R-Sync` does not really need to wait for all the clients in the federation to be available.

Moreover, Table 3.6 shows why partial participation with uniform sampling could be an issue in classical FL algorithms that are not immune to statistical heterogeneity. Indeed, sampling all the distinct clients at least once using uniform random sampling may require even 10k rounds. Therefore, it may take a long time before the model is trained on all the available data points, and some clients may never even be sampled. Conversely, since in `Fed3R-Sync` (and `Fed3R` and `Fed3R-RF`) all the clients are sampled once, it does not have this problem.

3.9.6 `Fed3R-Sync` and `Fed3R-RF-Sync` speed to target accuracy

Both `Fed3R-Sync` and `Fed3R-RF-Sync` are much faster than the baselines. For instance, `Fed3R-RF(10k)-Sync` on the Google Landmarks dataset achieves 40%

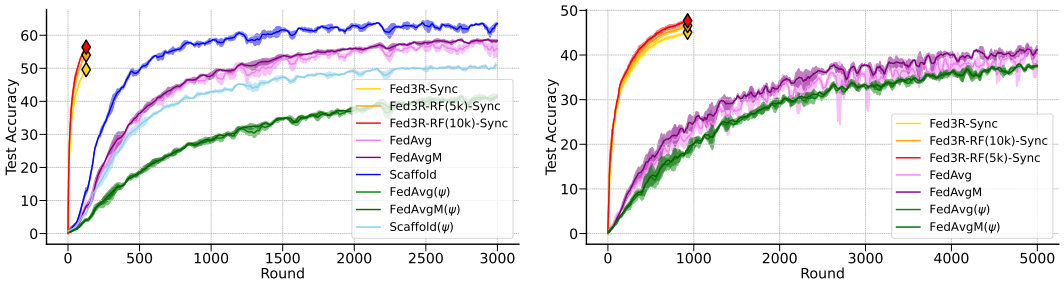


Figure 3.10: Number of rounds required by `Fed3R-Sync` and `Fed3R-RF-Sync` to target accuracy, compared with the baselines. The results for the Google Landmarks dataset are presented on the left, while the results for the iNaturalist dataset are presented on the right.

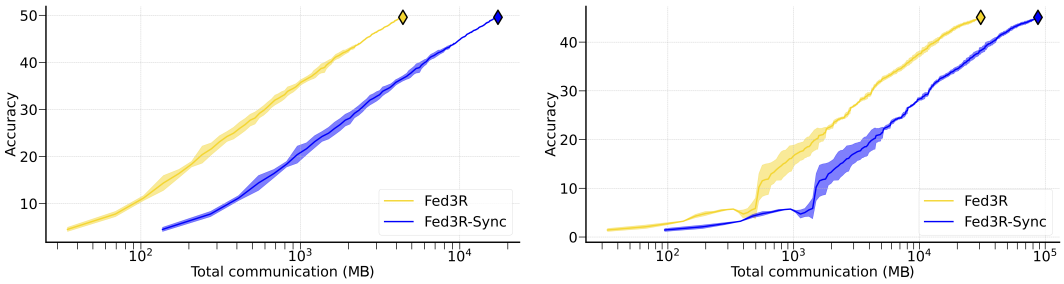


Figure 3.11: Comparison between the total communication costs of `Fed3R` algorithm with the `Fed3R-Sync` costs. The communication costs for Google Landmarks are shown on the left, while the communication costs for iNaturalist are shown on the right.

accuracy after 27.3 rounds on average, compared to the 528.7 (speedup ×19.3) needed by `FedAvg`, 285.7 needed by `Scaffold` (speedup ×10.5), and 2251.3 (speedup ×82.4) and 690.33 (speedup ×25.3) needed by `FedAvg(`ψ`)` and `Scaffold(`ψ`)`, respectively. Similar considerations can be made for other values of accuracy and for the iNaturalist experiments, as shown in Figure 3.10.

3.9.7 Communication costs of Fed3R and Fed3R-Sync

Figure 3.11 empirically shows that `Fed3R` is, on average, almost 10 times less communication expensive than `Fed3R-Sync` in both the splits. This is thanks to the fact that, in `Fed3R`, each client has to communicate only the $A_k \in \mathbb{R}^{d \times d}$ matrix and the $b_k^c \in \mathbb{R}^d$, $\forall c \in \mathcal{C}_k$ vectors associated with the local classes, compared with the

Table 3.7: Comparison of `Fed3R` with `FedNCM` [132], another closed-form classifier existing in literature, and other gradient-based FL algorithms at convergence, without classifier parameters initialization. ψ indicates the classifier, `Name_Alg`(ψ) indicates federated fine-tuning of the classifier only while keeping the pre-trained feature extractor parameters fixed, and `Name_Alg` indicates federated fine-tuning of the whole model. We use the N/A notation for the results of the experiments that failed to converge.

Dataset	Algorithm	Accuracy (%)	Total communication	Avg. computation per client
iNaturalist Users-120k	`Fed3R`	**45.1** ± 0.0	**30.7 GB**	4.0 GFLOPs
	`FedNCM`	32.2 ± 0.0	57.1 GB	**3.2 GFLOPs**
	`FedAvg(`ψ`)`	36.7 ± 0.4	616.9 GB	108.6 GFLOPs
	`FedAvgM(`ψ`)`	37.6 ± 0.2	616.9 GB	108.6 GFLOPs
	`Scaffold(`ψ`)`	N/A	N/A	N/A
	`FedAvg`	39.5 ± 3.2	1.5 TB	322.9 GFLOPs
	`FedAvgM`	39.3 ± 0.7	1.5 TB	322.9 GFLOPs
	`Scaffold`	N/A	N/A	N/A
iNaturalist Geo-1k	`Fed3R`	45.1 ± 0.0	**1.4 GB**	**108.9 GFLOPs**
	`FedNCM`	32.2 ± 0.0	2.3 GB	**108.5 GFLOPs**
	`FedAvg(`ψ`)`	44.1 ± 0.5	124.0 GB	550.2 GFLOPs
	`FedAvgM(`ψ`)`	43.9 ± 0.3	124.0 GB	550.2 GFLOPs
	`Scaffold(`ψ`)`	47.1 ± 0.4	248.0 GB	550.2 GFLOPs
	`FedAvg`	51.0 ± 0.1	296.0 GB	1.6 TFLOPs
	`FedAvgM`	51.1 ± 0.4	296.0 GB	1.6 TFLOPs
	`Scaffold`	**52.5** ± 0.2	592.0 GB	1.6 TFLOPs
iNaturalist Geo-3k	`Fed3R`	45.1 ± 0.0	**579.6 MB**	**297.6 GFLOPs**
	`FedNCM`	32.2 ± 0.0	831.5 MB	**296.7 GFLOPs**
	`FedAvg(`ψ`)`	46.0 ± 0.9	124.0 GB	1.5 TFLOPs
	`FedAvgM(`ψ`)`	46.9 ± 0.3	124.0 GB	1.5 TFLOPs
	`Scaffold(`ψ`)`	48.0 ± 0.3	248.0 GB	1.5 TFLOPs
	`FedAvg`	47.8 ± 3.5	296.0 GB	4.5 TFLOPs
	`FedAvgM`	**50.8** ± 1.2	296.0 GB	4.5 TFLOPs
	`Scaffold`	46.4 ± 3.6	592.0 GB	4.5 TFLOPs
Google Landmarks	`Fed3R`	49.6 ± 0.0	**4.4 GB**	40.0 GFLOPs
	`FedNCM`	36.2 ± 0.0	13.1 GB	**31.7 GFLOPs**
	`FedAvg(`ψ`)`	41.0 ± 1.6	618.8 GB	4.8 TFLOPs
	`FedAvgM(`ψ`)`	40.8 ± 1.2	618.8 GB	4.8 TFLOPs
	`Scaffold(`ψ`)`	51.7 ± 0.3	1.2 TB	4.8 TFLOPs
	`FedAvg`	57.7 ± 1.2	1.1 TB	14.1 TFLOPs
	`FedAvgM`	58.7 ± 0.8	1.1 TB	14.1 TFLOPs
	`Scaffold`	**63.4** ± 0.9	2.3 TB	14.1 TFLOPs

`Fed3R-Sync` algorithm that communicates the whole matrix $b_k \in \mathbb{R}^{d \times C}$ along with the matrix A_k.

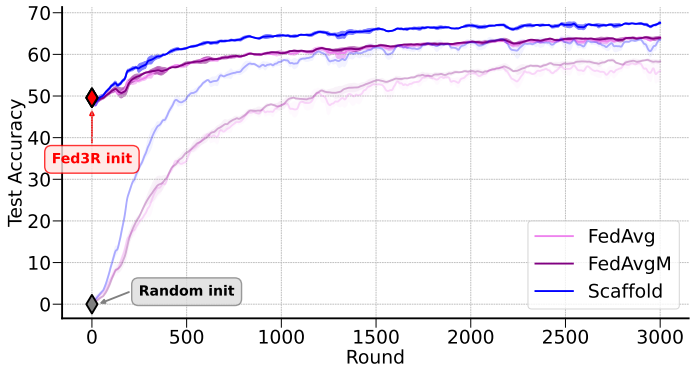
3.9.8 Comparison between Fed3R and the Federated Learning baselines

Table 3.7 shows that Fed3R and FedNCM **require 100 to 1000 times less computation and communication** than the baseline methods. However, **Fed3R outperforms FedNCM by approximately 13 percentage points in accuracy on both the Google Landmarks and iNaturalist datasets, while requiring similar communication and computation costs.** Additionally, despite Fed3R relying solely on the expressive power of the pre-trained feature extractor for the target FL task (*i.e.,* the feature extractor remains fixed and Fed3R only assembles a classifier), it achieves the highest accuracy on iNaturalist-Users-120k compared to the other methods. This holds even when compared with FedAvg and FedAvgM, which instead fine-tune the entire model, including the feature extractor. This result emphasizes the effectiveness of the Fed3R classifier in real-world, cross-device settings with high statistical heterogeneity.

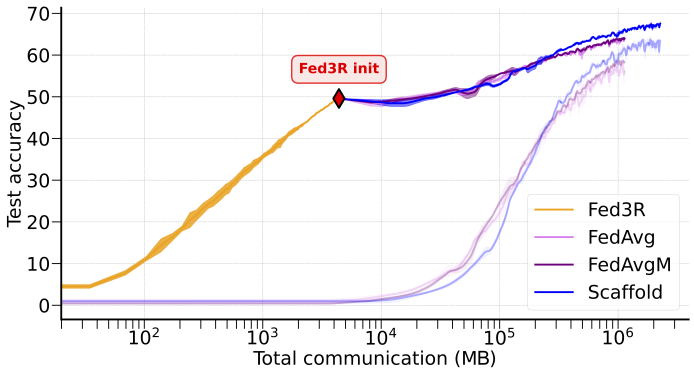
3.9.9 Fine-tuning after Fed3R initialization

Fed3R is sufficient to surpass baseline methods on highly statistically heterogeneous scenarios such as iNaturalist-Users-120k. However, in the Google Landmarks or iNaturalist-Geo-3k and iNaturalist-Geo-1k experiments, which are less affected by statistical heterogeneity, Fed3R alone is not enough to surpass the FL baselines, although its accuracy is still higher when the available budget of communication or computation is a constraint. Fortunately, as introduced in Section 3.7, Fed3R parameters W^* can serve as a robust initialization for the classifier's parameters θ_ψ, which can be then fine-tuned with any FL algorithm.

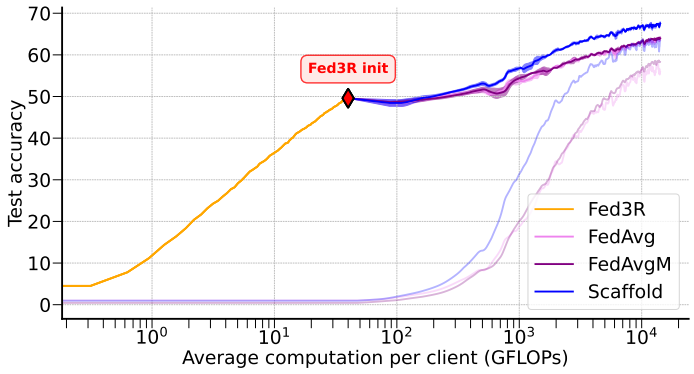
Figure 3.12 shows that this strategy improves accuracy compared with the same algorithms without Fed3R initialization while retaining computational and communication gains. Moreover, it stabilizes the training, accelerating convergence.



(a)

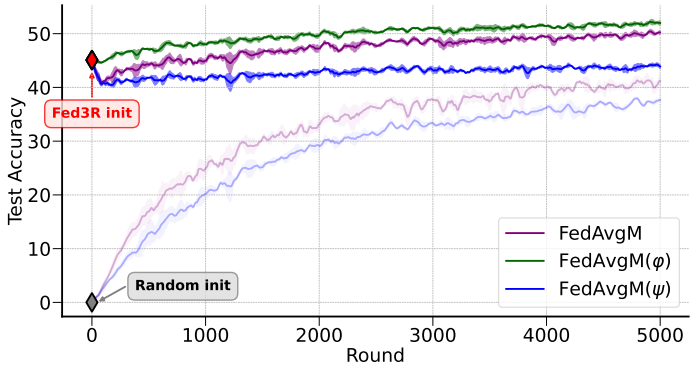


(b)

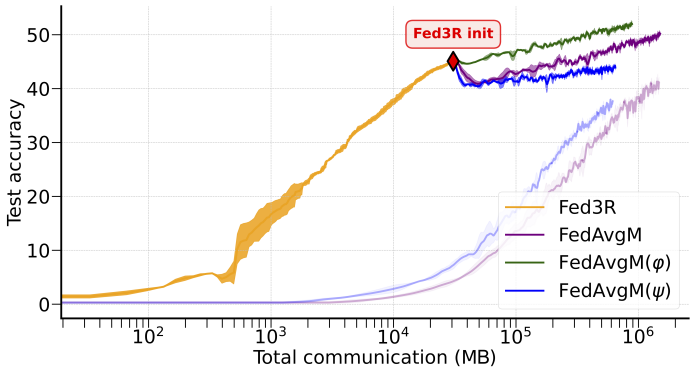


(c)

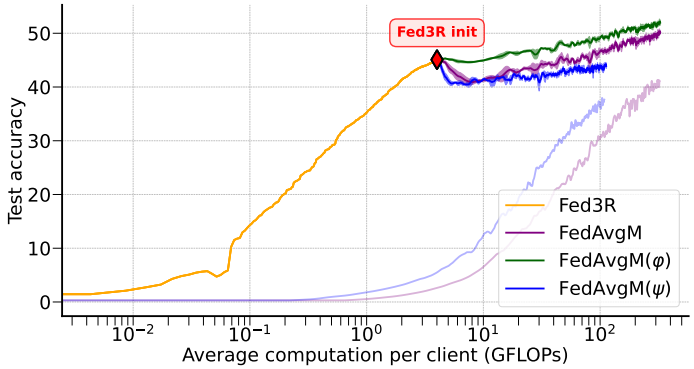
Figure 3.12: Comparison between FL algorithms on the Google Landmarks dataset, and the same algorithms fine-tuned after initializing the classifier parameters using `Fed3R`. Figure 3.12a: test accuracy by rounds. Figure 3.12b: test accuracy by total communication budget. Figure 3.12c: test accuracy by expected computation per client. The red marker indicates the `Fed3R` initialization point, which allows fine-tuning with any FL algorithm. Less intense colors are associated with the baseline algorithms without `Fed3R` initialization.



(a)



(b)



(c)

Figure 3.13: Comparison between different FT strategies on the iNaturalist dataset. The presented experiments use FedAvgM as the FT algorithm since the results with FedAvg were similar but noisier. The FedAvgM(φ) experiment without Fed3R initialization is omitted because this experiment achieves poor results as the classifier is randomly initialized and not fine-tuned. Figure 3.13a: test accuracy by rounds. Figure 3.13b test accuracy by total communication budget. Figure 3.13c: test accuracy by expected computation per client. The red marker indicates the Fed3R initialization point. Less intense colors are associated with the baseline algorithms without Fed3R initialization.

Table 3.8: Comparison between fine-tuning strategies, with and without Fed3R initialization, using different FL baselines as FT algorithms (Acc. (%)). Since the classifier parameters θ_ψ are randomly initialized when they are not initialized with the Fed3R parameters, the performance of the FT(φ) strategy would always be near 0. Therefore, they are omitted (indicated with the - symbol). Scaffold failed to converge in all the experiments in the iNaturalist scenario (indicated with N/A).

Dataset	ψ init	FT alg	FT(φ)	FT(ψ)	FT($\mathcal{M}$)
iNaturalist Users-120k	✗	FedAvg	-	36.7 ± 0.4	39.5 ± 3.2
	Fed3R	FedAvg	50.8 ± 0.2	42.0 ± 0.3	49.0 ± 0.6
	✗	FedAvgM	-	37.6 ± 0.2	39.3 ± 0.7
	Fed3R	FedAvgM	**51.5** ± **0.2**	43.5 ± 1.9	49.8 ± 0.8
	✗	Scaffold	-	N/A	N/A
	Fed3R	Scaffold	N/A	N/A	N/A
iNaturalist Geo-1k	✗	FedAvg	-	44.1 ± 0.5	51.0 ± 0.1
	Fed3R	FedAvg	53.7 ± 0.2	49.2 ± 0.4	53.6 ± 0.1
	✗	FedAvgM	-	43.9 ± 0.3	51.1 ± 0.4
	Fed3R	FedAvgM	53.8 ± 0.1	49.4 ± 0.2	53.3 ± 0.2
	✗	Scaffold	-	47.1 ± 0.4	52.5 ± 0.2
	Fed3R	Scaffold	53.7 ± 0.1	49.3 ± 0.2	**55.7** ± **0.1**
iNaturalist Geo-3k	✗	FedAvg	-	46.0 ± 0.9	47.8 ± 3.5
	Fed3R	FedAvg	50.8 ± 1.6	47.0 ± 1.4	50.9 ± 1.5
	✗	FedAvgM	-	46.9 ± 0.3	50.8 ± 1.2
	Fed3R	FedAvgM	50.1 ± 0.2	46.4 ± 0.3	51.0 ± 0.2
	✗	Scaffold	-	48.0 ± 0.3	46.4 ± 3.6
	Fed3R	Scaffold	49.3 ± 2.3	47.0 ± 1.2	**54.1** ± **1.7**
Google Landmarks	✗	FedAvg	-	41.0 ± 1.6	57.7 ± 1.2
	Fed3R	FedAvg	59.6 ± 0.2	56.7 ± 0.4	64.1 ± 0.4
	✗	FedAvgM	-	40.8 ± 1.2	58.7 ± 0.8
	Fed3R	FedAvgM	59.0 ± 0.2	56.2 ± 0.5	64.1 ± 0.2
	✗	Scaffold	-	51.7 ± 0.3	63.4 ± 0.9
	Fed3R	Scaffold	63.4 ± 0.1	58.0 ± 1.5	**67.4** ± **0.3**

Fine-tuning the entire model initialized from the Fed3R classifier works well in mildly heterogeneous settings, such as Google Landmarks. However, as shown in Figure 3.13, this approach (Fed3R+FT(f)) is not suitable for more heterogeneous datasets like iNaturalist, which contains ten times the number of clients. Indeed, fine-tuning the entire model can impair predictive performance and slow down convergence due to classifier bias, leading to destructive interference during the aggregation phase, as discussed in Section 3.7.

Figure 3.13 indicates that accuracy does not improve when only the classifier is fine-tuned in the (Fed3R+FT(ψ)) experiment because the feature extractor is not optimized for the target task, which highlights that the pre-trained features still lack sufficient expressiveness. Conversely, by keeping the classifier fixed and fine-tuning only the feature extractor, it is possible to mitigate the issues caused by heterogeneity effectively. This approach prevents destructive interference and enables the feature extractor to adapt to the Fed3R classifier and the target task, ultimately enhancing final accuracy.

Finally, Table 3.8 summarizes the results of the Google Landmarks and iNaturalist experiments, highlighting the consistent advantages of initializing the classifier with the parameters from Fed3R, showing that fine-tuning the model with Fed3R initialization results in significantly higher final accuracies when compared to training from scratch.

Additionally, as shown in Figures 3.12 and 3.13, methods that utilize Fed3R initialization outperform the baselines in terms of accuracy, communication efficiency, and computational cost consistently. These findings emphasize the importance of robust initialization in FL and demonstrate how Fed3R effectively serves this purpose. Providing optimal classifier parameters equivalent to the centralized RR solution, Fed3R ensures resistance to statistical heterogeneity and promotes efficient and effective learning in demanding federated environments.

3.9.10 Fed3R modularity and domain imbalance robustness

This section empirically demonstrates that Fed3R can be combined with other orthogonal methods to further improve performance. FedRDN [157] has been chosen as a representative pre-processing method to compare against –or combine with– Fed3R. FedRDN is a novel orthogonal FL method to address domain imbalance in

Table 3.9: Accuracy (%) on the FL test set of the iNaturalist dataset, using the Pers-Geo-100 training clients, of 5 different strategies including various combinations of `Fed3R`, `FedAvg`, and `FedRDN`. The datasets are perturbed to enhance the domain imbalance. The severity of the perturbation is controlled by a domain imbalance intensity parameter δ.

ψ **init**	**FT alg**	**Domain shift intensity** δ		
		1.2	1.5	2.0
✗	`FedAvg`	25.3 $\pm$ 1.2	24.9 $\pm$ 1.0	17.6 $\pm$ 2.3
✗	`FedAvg + FedRDN`	27.3 $\pm$ 2.6	25.7 $\pm$ 2.5	20.9 $\pm$ 0.7
`Fed3R`	✗	40.0 $\pm$ 0.0	33.6 $\pm$ 0.0	28.6 $\pm$ 0.0
`Fed3R`	`FedAvg`	44.4 $\pm$ 1.4	44.2 $\pm$ 1.2	40.2 $\pm$ 1.6
`Fed3R`	`FedAvg + FedRDN`	**47.1** $\pm$ **0.3**	**45.6** $\pm$ **2.3**	**41.7** $\pm$ **0.8**

FL. In this section, it is also shown that `Fed3R` is robust to FL scenarios with various domain imbalance intensities.

Experimental setup

For the training, the clients from the Pers-Geo-100 split of iNaturalist have been utilized (see Table 4.1, Figure 4.3 and Section 4.3.1 for information about this dataset). The global model is evaluated using the original iNaturalist test set.

To induce domain imbalance, the clients are divided into three equal-sized groups, each associated with a specific RGB channel. In the first group, the red channel of the images is multiplied by a domain imbalance intensity factor δ, while the green and blue channels are multiplied by the factor $1/\delta$. The domains of the other two groups are analogously shifted by enhancing the green and blue channels, respectively. The red, green, and blue channels of the test set are also enhanced in the same manner over three equally sized groups.

Results

Table 3.9 presents the results of the experiments on the robustness of `Fed3R` to domain imbalance and its compatibility with other orthogonal methods across three different levels of domain imbalance intensity.

Table 3.10: Quality of the feature extractors measured at convergence using Fed3R and Fed3R-RF. These results can also be interpreted as an application of Fed3R (Fed3R-RF) using fine-tuned parameters of the feature-extractor to enhance predictive performance further. Values represent the percentage accuracy.

Dataset	ψ init	FT alg	FT($\cdot$)	softmax	Fed3R	Fed3R-RF-5k	Fed3R-RF-10k
iNat.	-	FedAvg	$\mathcal{M}$	39.5 $\pm$ 3.2	53.1 $\pm$ 0.9	55.3 $\pm$ 1.1	56.7 $\pm$ 1.0
	Fed3R	FedAvg	$\mathcal{M}$	49.0 $\pm$ 0.6	52.2 $\pm$ 0.3	53.7 $\pm$ 0.2	54.9 $\pm$ 0.3
	Fed3R	FedAvg	φ	50.8 $\pm$ 0.2	54.6 $\pm$ 0.1	55.5 $\pm$ 0.1	**56.8** $\pm$ **0.2**
Google Landmarks	-	FedAvg	$\mathcal{M}$	57.7 $\pm$ 1.2	58.8 $\pm$ 2.1	64.4 $\pm$ 0.2	66.5 $\pm$ 0.2
	Fed3R	FedAvg	$\mathcal{M}$	64.1 $\pm$ 0.4	59.6 $\pm$ 0.2	62.8 $\pm$ 0.1	**64.6** $\pm$ **0.1**
	Fed3R	FedAvg	φ	59.6 $\pm$ 0.2	62.1 $\pm$ 0.1	62.5 $\pm$ 0.1	**64.6** $\pm$ **0.1**
	-	Scaffold	$\mathcal{M}$	63.4 $\pm$ 0.9	57.0 $\pm$ 1.9	66.5 $\pm$ 0.5	68.6 $\pm$ 0.5
	Fed3R	Scaffold	$\mathcal{M}$	67.4 $\pm$ 0.3	61.8 $\pm$ 0.1	64.4 $\pm$ 0.2	66.3 $\pm$ 0.1
	Fed3R	Scaffold	φ	63.4 $\pm$ 0.1	64.3 $\pm$ 0.2	66.7 $\pm$ 0.1	**68.7** $\pm$ **0.2**

FedAvg is significantly impacted by domain imbalance, resulting in declining accuracies as the intensity of the imbalance increases. FedRDN improves robustness to domain imbalance, mildly improving the accuracy of FedAvg by 1-3%. Nonetheless, Fed3R alone outperforms both FedAvg by itself and the combination of FedAvg with FedRDN by a considerable margin (e.g., +15% and +13% with respect to FedAvg and FedAvg + FedRDN, in the setting with $\delta = 1.2$). Additionally, as discussed in the main paper, the Fed3R classifier can be further fine-tuned with another FL algorithm, such as FedAvg; results show that the Fed3R + FT(FedAvg) strategy boosts accuracy in all three scenarios. Finally, adding FedRDN to the Fed3R + FT(FedAvg) strategy leads to an additional increase in accuracy (e.g., +3% with respect to Fed3R + FT(FedAvg), and +20% with respect to FedAvg + FedRDN, in the setting with $\delta = 1.2$).

3.9.11 Features quality evaluation through Fed3R

Thanks to its closed-form formulation, Fed3R classifiers can also be used as tools to assess the quality of feature extractors and the linear separability of the latent space easily. Indeed, the quality of these linear classifiers depends only on the separability of the latent feature space. Therefore, high Fed3R accuracies are proportional to the quality of the feature maps, and the Fed3R algorithm can be executed at the end of the training to evaluate the quality of the feature extractor.

Table 3.10 compares the quality of the feature extractor parameters of different FL algorithms at convergence using Fed3R. The results indicate an improvement in the quality of the features learned when the classifier is initialized with Fed3R parameters. Indeed, Fed3R helps stabilize the training process, mitigating destructive interference and catastrophic forgetting. Specifically, Fed3R+FT(f) and Fed3R+FT(φ) consistently achieve higher Fed3R accuracy than the corresponding baseline with random classifier initialization on both the Google Landmarks and iNaturalist datasets and for all the FT algorithms. Moreover, Fed3R+FT(φ) always outperforms Fed3R+FT(f), as fixing the classifier eliminates the classifier bias issue entirely.

Finally, Table 3.10 also shows that Fed3R or Fed3R-RF can provide the highest accuracy even if applied on feature extractors after the training, envisioning possible future directions for the applicability of the Fed3R and Fed3R-RF algorithms. Indeed, the Fed3R-RF(10k) classifier constructed using the features that can be extracted from the trained feature extractors consistently provides better results than the original classifier.

3.9.12 Normalization and global class imbalance

Section 3.4 describes that Fed3R, to tackle the presence of global class imbalance, normalizes the columns of the final RR classifier. This normalization is achieved by dividing each column of the RR classifier by its norm, which enhances the importance of under-represented classes. To evaluate the impact of this normalization, the final accuracy of the Fed3R classifier is assessed both with and without normalization across four variants of the Cifar100 dataset. Each variant introduces different levels of global class imbalance. The details of these variants are as follows:

- **Variant A**: Only half of the data is utilized for each class. This variant does not exhibit any global class imbalance.
- **Variant B**: The classes are randomly divided into two groups. For the first group, only 25% of the available data for each class is used, while for the second group, 75% of the available data for each class is retained.
- **Variant C**: The classes are randomly divided into four groups. The first group retains only 20% of the available data per class, the second group retains 40%, the third group retains 60%, and the fourth group retains 80%.

Table 3.11: Final accuracy of the Fed3R classifier with and without final normalzation.

Cifar100 variant	w/ norm	w/o norm
A	56.0 ± 0.0	56.0 ± 0.0
B	53.5 ± 0.3	44.7 ± 0.3
C	55.0 ± 0.4	48.7 ± 0.3
D	49.5 ± 0.6	44.0 ± 0.7

- **Variant D**: The classes are divided into ten random groups. From the first to the last group, only 5%, 15%, 25%, ..., and 95% of the available data for each class is retained, respectively.

All these variants have a total of 25k samples, that is exactly half of the total training samples of the original Cifar100 dataset (see Table 3.1).

The performance of the Fed3R classifier with and without the final normalization has been evaluated for each of these variants. As usual, each experiment has been repeated three times with different random seeds, which for these experiments affect only how the clients are grouped since Fed3R is a deterministic algorithm. Experiments have been conducted using the Dirichlet parameter $\alpha = 0$, although the final Fed3R accuracy is always the same, regardless of (federated) class imbalance. Results are shown in Table 3.11. As expected, there is no performance drop by removing the normalization in Variant A because this dataset has no global class imbalance. Instead, there is always a performance drop from 5% to 9% in the other cases, showcasing the importance of normalizing the Fed3R classifier in the presence of global class imbalance.

3.10 Broader impact

The Fed3R family of algorithms is immune by construction to statistical heterogeneity and can also serve as a robust initialization for subsequent fine-tuning with other FL algorithms. Findings reveal that the features produced during the fine-tuning stage exhibit greater robustness compared to those achieved by other methods at convergence, underscoring that in challenging cross-device settings, the quality of the feature extractor may serve as a bottleneck alongside the classifier's quality.

Moreover, Fed3R is geared towards minimizing communication and computation costs and accelerating convergence time while adhering to the privacy constraints inherent in FL while providing remarkable speed. By lightening the FL training load, Fed3R not only improves efficiency but also reduces the energy required for training. This not only benefits cost savings but also contributes to reducing environmental pollution associated with the energy production required for training models. This has the potential to significantly impact various applications across industries, making them more accessible and cost-effective.

The rapid execution and resource efficiency of the Fed3R method could lead to increased adoption of FL techniques, enabling advancements in fields ranging from finance and autonomous systems to healthcare and beyond. The next chapter will show that Fed3R can also be applied in the Personalized FL scenario.

Chapter 4

The importance of a robust initialization in Personalized Federated Learning

Part of the content of this chapter, including the methodologies and the results, is adapted, with permission, from the following paper:

- ***E. Fanì**, R. Camoriano, B. Caputo and M. Ciccone. "**Resource-Efficient Personalization in Federated Learning with Closed-Form Classifiers.**" **IEEE Access**, 2025* [21].

The previous chapter introduced `Fed3R`, an innovative algorithm that deploys a classifier immune to statistical heterogeneity due to the closed-form solution of RR. It also demonstrated how the parameters of `Fed3R` can serve as an effective initialization for further fine-tuning the classifier using any FL algorithm, addressing client drift and data recency bias issues.

This chapter builds upon that foundation by presenting thorough experiments showing how `Fed3R` parameters can enhance PFL training. In this context, the ultimate goal is to train a local model for each client tailored to solve its specific local distribution. The chapter demonstrates that `Fed3R` accelerates this process, reduces costs, and yields better overall model performance.

Furthermore, this chapter provides a detailed analysis of the significance of robust initialization in PFL training. This analysis extends beyond scenarios where

the initialization pertains solely to the classifier using `Fed3R` parameters. The experiments are structured into four phases, incorporating both FL and PFL training.

While each client ultimately aims to develop a single model suitable for its local distribution, the benefits of the knowledge that can be extracted from training a global model on all the clients through classical FL should not be overlooked, especially since clients, in cross-device scenarios, often have limited data. This chapter clearly illustrates the optimal trade-off between these two types of training, depending on the communication and computational budgets available.

Lastly, this chapter introduces Only Local Labels (`OLL`), a novel algorithm designed explicitly for PFL. `OLL` effectively eliminates the interference that arises from predicting classes outside the local distribution. As a result, it simplifies local training for clients, allowing them to concentrate fully on their local distribution, thereby speeding up training and enhancing final accuracy.

Here is an outline of the sections in this chapter:

- Section 4.1 introduces the PFL setting, from the general motivations and existing literature to the formal mathematical framework.
- Section 4.2 presents the `OLL` algorithm.
- Section 4.3 shows the empirical experiments to analyze the impact of the initialization in PFL, also including `Fed3R` and `OLL` experiments.
- Section 4.4 discusses the impact of this analysis and possible future directions for future work.

4.1 Introduction to Personalized Federated Learning

4.1.1 Motivation and prior works

Traditional FL aims to train a shared global model across various clients. However, this approach often overlooks the diversity in client data, leading to a one-size-fits-all model that may not perform optimally for every participant. Due to the challenges of learning global models in heterogeneous environments, some researchers [69, 19, 14] suggest that tailoring models to meet the specific needs of individual clients while still

benefiting from shared knowledge across the network, could be a viable alternative, leading to the introduction of the PFL framework.

Several strategies for PFL have been proposed, including model personalization techniques for multi-task learning [158, 159] and meta-learning [19, 160]. Other methods include clustered FL (CFL) approaches [161–165], where the objective is to partition clients into non-overlapping groups based on similarities in their data distributions, with each group having its own model, and regularization strategies [166–169]. Notably, `Ditto` [167] incorporates a regularization term similar to the one used in [15] to align local model updates with the global model. In contrast, `pFedMe` [166] utilizes Moreau envelopes to decouple local and global model updates, enabling client-specific customization without sacrificing the advantages of collaborative learning.

Other PFL approaches involve dividing model parameters into two groups. Using the terminology from [170], some *shared parameters* can be aggregated globally in a traditional FL fashion, while other *personal parameters* can be considered as client-specific and remain local, capturing individual data distributions. Specifically, [171–173] propose to train both groups simultaneously, whereas [174, 175] suggest learning the two groups separately, updating personal parameters before the shared ones. The authors of [170] refer to these two strategies as `FedSim` and `FedAlt`, respectively.

4.1.2 Formal introduction to Personalized Federated Learning

In contrast to FL, PFL focuses on finding distinct, optimized local models for each client. The objective in PFL is, for each client $k \in \mathcal{K}$, to determine the optimal local parameters $\theta_k^* \in \Theta$ that minimize the following expected risk:

$$\{\theta_k^*\}_{k\in\mathcal{K}} = \underset{\{\theta_k\}_{k\in\mathcal{K}}}{\arg\min} \sum_{k\in\mathcal{K}} q_k \mathbb{E}_{(x,y)\sim p_k}[\mathcal{L}(f(x;\theta_k), y)], \tag{4.1}$$

which can equivalently be formulated as:

$$\theta_k^* = \underset{\theta_k\in\Theta}{\arg\min}\, \mathbb{E}_{(x,y)\sim p_k}[\mathcal{L}(f(x;\theta_k), y)],\ \forall k \in \mathcal{K}, \tag{4.2}$$

where the constant value $q_k \in [0, 1]$ has been dropped as it does not affect the final optimal solution. In practice, each client $k \in \mathcal{K}$ has to individually solve the following local ERM problem:

$$\theta_k^* = \arg\min_{\theta_k \in \Theta} \sum_{(x,y)\in\mathcal{D}_k} \mathcal{L}(f(x;\theta_k), y). \tag{4.3}$$

In other words, each client seeks to determine the optimal parameters for its own local data distribution. This approach provides greater flexibility, even admitting different models $f_k : \mathcal{X} \to \mathcal{Y}$ for each client, and allows for adaptation to the specific characteristics of each client's data. However, clients often have only limited data, especially in cross-device scenarios. Therefore, it can be beneficial to leverage the knowledge gained from other clients.

Following the same framework of [170], in this manuscript, two PFL distinct strategies are considered: *partial personalization* and *full personalization*.

In partial personalization (PP), the local parameters θ_k of each client $k \in \mathcal{K}$ are divided into two sets: *shared parameters* $u_k \in \mathcal{U}$ and *personal parameters* $v_k \in \mathcal{V}$, where $u_k \cup v_k = \theta_k$ and $u_k \neq \emptyset$, $v_k \neq \emptyset$. Shared parameters are designed to capture global knowledge across all clients, whereas personal parameters are tailored to learn and adapt to individual client's specific local data distributions. At the beginning of each round $t \in [T]$, the server sends the current global shared parameters u^t to the sampled clients $k \in \mathcal{K}'$, which initialize their local shared parameters with $u_k^t = u^t$. The clients then locally optimize both their shared and personal parameters, resulting in updated parameters $\theta_k^{t+1} = u_k^{t+1} \cup v_k^{t+1}$. However, only the updated shared parameters u_k^{t+1} are communicated back to the server, while the personal parameters v_k^{t+1} are retained locally. The server then aggregates the received u_k^{t+1} to obtain the updated global shared parameters u^{t+1} for the next round.

In contrast, full personalization (FP) represents the scenario where $u_k = \emptyset$ and $v_k \subseteq \theta_k$ for all $k \in \mathcal{K}$. The case where $v_k = \theta_k$ can be viewed as a special case of PP where all parameters are personal. This strategy requires no communication between the clients and the server, as each client trains its model independently on its local data. Consequently, no rounds are necessary with this strategy.

In PFL evaluation, it is usually assumed that each client $k \in \mathcal{K}$ has access to an additional dataset $\mathcal{D}_k^{\text{test}}$, not available during the training phase, representative of the

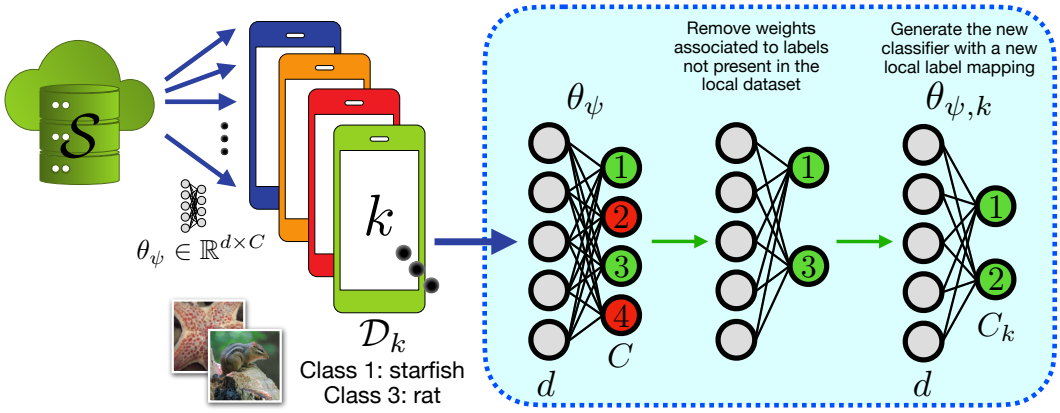


Figure 4.1: The OLL algorithm. The global mapping associates the labels 1 and 3 with the categories "starfish" and "rat", respectively. For each client k, OLL filters out the columns of the classifier weights θ_ψ that correspond to the classes not present in the local dataset $\mathcal{D}_k$ (specifically classes 2 and 4 in this example). This process generates a new classifier, parameterized by $\theta_{\psi,k}$, which utilizes a local mapping. This classifier is already effective in enhancing performance and can be further fine-tuned to better align with the local data distribution.

local data distribution of client k. This manuscript evaluates the final performance for image classification using the *weighted mean accuracy (WMA)* over the test datasets $\mathcal{D}_k^{\text{test}}$, defined as:

$$\texttt{WMA} = \sum_{k\in\mathcal{K}} \frac{n_k^{\text{test}}}{n^{\text{test}}} \sum_{(x,y)\in\mathcal{D}_k^{\text{test}}} \frac{\mathbb{1}\left[\arg\max_{c\in[C]} \hat{y}_{k,c} = \arg\max_{c\in[C]} y_c\right]}{\left|\mathcal{D}_k^{\text{test}}\right|}, \tag{4.4}$$

where $|\mathcal{D}_k^{\text{test}}| = n_k^{\text{test}}$, $n^{\text{test}} = \sum_{k\in\mathcal{K}} n_k^{\text{test}}$ and $\hat{y}_k = f_k(x;\theta_k)$.

4.2 Full personalization with Only Local Labels (OLL)

Fine-tuning the original classifier $\theta_\psi \in \mathbb{R}^{d\times C}$ can be inefficient, especially in highly heterogeneous settings where the local class set $\mathcal{C}_k$ is significantly smaller than the global class set $\mathcal{C}$. In these cases, the classifier should ideally avoid predicting classes that are outside the client's local distribution ($\mathcal{C} \setminus \mathcal{C}_k$), since here it is assumed that clients operate with closed-set local datasets, as is common in many FL and PFL scenarios [13, 62, 16, 77, 69]. However, due to its $d \times C$ dimension, the original classifier retains the ability to predict these irrelevant classes, which can degrade

Algorithm 7 - The Only Local Labels (OLL) algorithm.

Require:
Classifier parameters $\theta_\psi \in \mathbb{R}^{d \times C}$
Set of global classes $\mathcal{C}$, set of local classes $\mathcal{C}_k \subseteq \mathcal{C}$
Initialize $\theta_{\psi,k} = 0_{d \times C_k}$
for each $c \in \mathcal{C}$ **do**
  if $c \in \mathcal{C}_k$ **then**
    $\theta^c_{\psi,k} = \theta^c_\psi$
  end if
end for
Return $\theta_{\psi,k}$

performance and slow down training. This issue becomes particularly pronounced during the initial fine-tuning iterations, when the predicted probabilities may not be sufficiently biased toward the local classes.

Only Local Labels (OLL) is designed to enhance FP by simplifying the local classifier. As illustrated in Figure 4.1, OLL prunes the columns associated with irrelevant classes ($\mathcal{C} \setminus \mathcal{C}_k$) from the classifier weights θ_ψ. This pruning process produces a new, smaller classifier $\theta_{\psi,k} \in \mathbb{R}^{d \times C_k}$, which focuses exclusively on the local classes $\mathcal{C}_k$ through a local label mapping.

This simplification provides several advantages. By preventing the prediction of irrelevant classes, the OLL method enhances the efficiency and performance of the local classifier. Additionally, it speeds up and simplifies the subsequent local fine-tuning by streamlining the local problem. Reducing the classifier's dimensionality also helps prevent interference from irrelevant classes. Importantly, OLL is a versatile technique that can be applied to any classifier, including those developed using Fed3R, introduced in Chapter 3.

The complete OLL algorithm is shown in Algorithm 7. A possible training strategy for PFL, which includes both Fed3R and OLL, is shown in Figure 4.2.

4.3 Experiments

This section introduces the experiments to evaluate the performance, communication costs, and computation costs of several proposed training strategies, with the ultimate

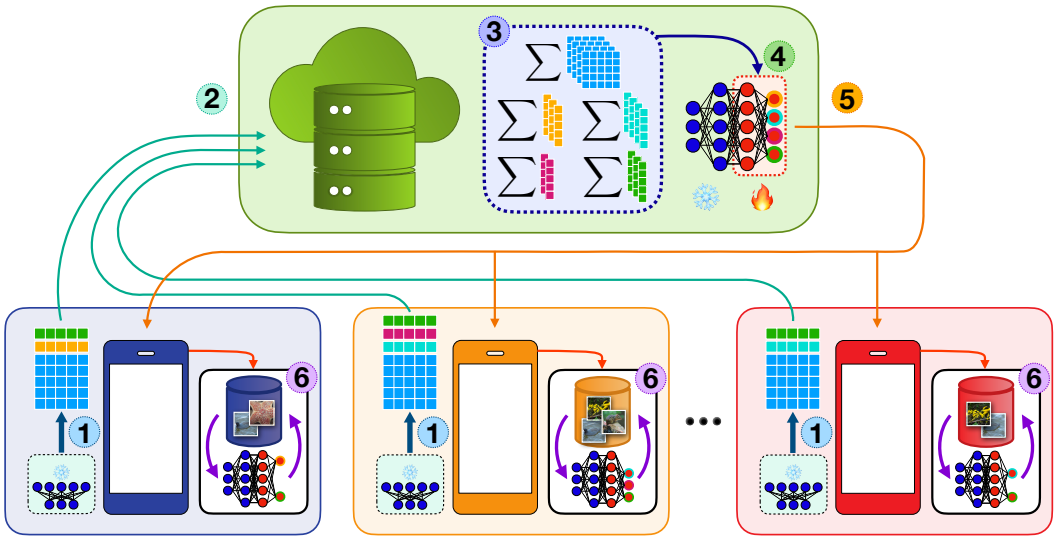


Figure 4.2: A possible PFL strategy which includes both the Fed3R and OLL methods. The learnable parameters of the model are pictorially represented by the edges between the red neurons of the model, while the other parameters are frozen. Classical PFL training may require multiple rounds of communication. Conversely, this strategy requires only one upstream communication from the clients to the server. Steps ①-④ are associated with Fed3R, while steps ⑤-⑥ are associated with OLL. ① Each client extracts the local Fed3R statistics using the pre-trained feature extractor. ② Each client shares its Fed3R statistics with the server. ③ The server aggregates the received statistics. ④ The server uses the aggregated statistics to compute the optimal parameters for the Fed3R classifier. ⑤ The server shares the Fed3R classifier with the clients. ⑥ The clients filter the parameters of the classifier, keeping only the ones associated with the local classes, and fine-tune the classifier locally.

goal of providing each client with the best possible personalized model. Specifically, this section is organized as follows:

- Section 4.3.1 presents the datasets, the taxonomy, and the implementation details on how the experiments have been performed.
- Section 4.3.2 complements Section 3.9.2, describing how the communication and computation costs have been calculated for the additional PFL methods considered in this chapter.
- Section 4.3.3 presents the experiments focused on the evaluation of the PP strategies.

Table 4.1: Summary of the information of the image classification datasets for PFL used in the experiments presented in this chapter.

Dataset	Split	Avg. samples per client	K	C	Tot. number of samples
Google Landmarks [150, 74, 170]	Pers-Users-160k	80.2	823	2028	83255
iNaturalist [73, 74, 170]	Pers-Geo-100	22.7	2714	1203	66165

- Section 4.3.4 presents the experiments focused on the evaluation of the `FP` strategies.
- Section 4.3.5 shows how `OLL` guarantees impressive performance and that combining `OLL` with `Fed3R` classifier initialization allows for completely avoiding the `PP` phase.

4.3.1 Datasets, taxonomy and implementation details

The implementation details of the PFL experiments in this chapter are the same as the implementation details of Section 3.9.1, with some differences and additions that are presented in this section.

The model is trained on a modified version of the Users-160k split of Google Landmarks, referred to in this manuscript as Pers-Users-160k. This split has been generated following the approach of [170], which excludes clients with fewer than 50 images from the original Users-160k split and the original global test set for this dataset. Instead, each remaining client is divided into a local training set and a test dataset, with approximately 45% of the total local samples allocated to the test dataset. The Pers-Users-160k split comprises a total of 823 clients.

In addition, in this manuscript, a new Pers-Geo-100 split has also been introduced for the iNaturalist dataset. This split is based on the Geo-100 split proposed by [74] and follows a similar protocol as described by [170]. In this case, clients with 10 or more local images are retained rather than discarding those with fewer than 50 images, thereby simulating a more heterogeneous and challenging scenario. The Geo-100 split has been preferred to Users-120k because most clients in Users-120k count less than 10 images. The proposed Pers-Geo-100 split results in a total of 2714 clients. Additional details of the two proposed splits for PFL are presented in Table 4.1 and in Figure 4.3.

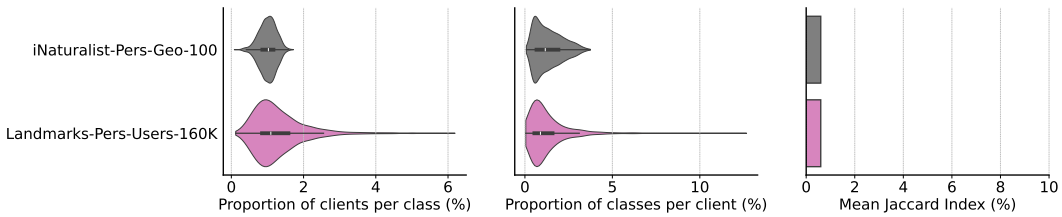


Figure 4.3: Statistical heterogeneity for the two splits of iNaturalist and Google Landmarks employed for the PFL experiments, using three different metrics. On the left: distribution of the proportion of clients per class (relative to the total number of classes). In the middle: distribution of the proportion of classes per client (relative to the total number of clients). On the right: The Mean Jaccard Index. All values are presented as percentages.

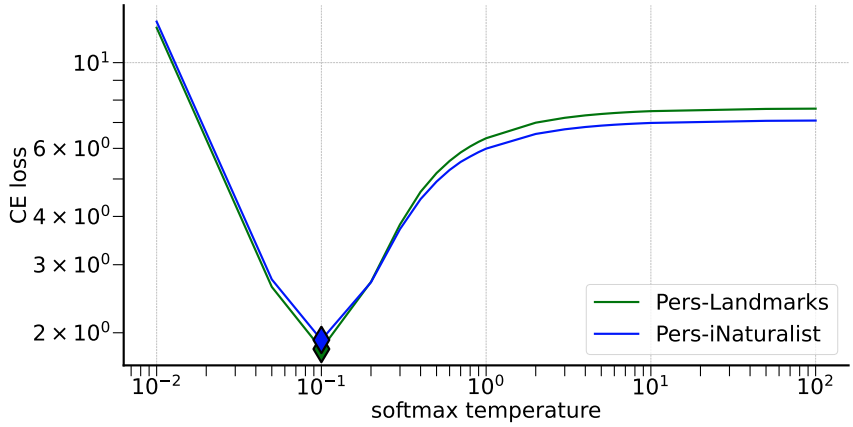


Figure 4.4: Tuning of the best temperature τ for the softmax classifier, after initializing its parameters with the `Fed3R` parameters, on the personalized Google Landmarks and iNaturalist datasets. The y-axis shows the average CE loss on the training set. The temperature value that guarantees the lowest CE loss has been chosen for all the experiments. The best value is consistently 0.1.

To systematically analyze the efficacy of the initialization in PFL, as well as the optimal strategy for leveraging global knowledge from all clients (*i.e.*,, the trade-off between learning from others and focusing on local training) with a given communication and/or computation budget, the experiments in this chapter are divided into **at most** four distinct phases:

- **Phase 1** and **Phase 2** are the same as Phase 1 and Phase 2 from Section 3.9.1.

  - Similarly to the FL experiments presented in Section 3.9, the chosen temperature for the softmax classifier is 0.1, as it provided the lowest training loss, as shown in Figure 4.4.

- **Phase 3:** Partial personalization (PP). This phase involves partial personalization training, as described in Section 4.1.2, where a subset of the model parameters is shared among clients while the remaining parameters are kept private.

  - Experiments are conducted with three PFL algorithms: `FedSim` [170], `FedAlt` [170], and `pFedMe` [166], running each for 1000 rounds. The selected hyper-parameters for these algorithms are equivalent to the best hyper-parameters found for `FedAvg` in Section 3.9.1.
  - Depending on how the personal and shared parameters are defined, in this manuscript are considered the following two `PP` strategies:
    * Shared feature extractor parameters θ_φ, personal classifier parameters θ_ψ. This strategy is denoted as `Name_Alg(`ψ`)`.
    * Shared classifier parameters θ_ψ, personal feature extractor parameters θ_φ. This strategy is denoted as `Name_Alg(`φ`)`.

    This focus is motivated by the observation that the classifier is generally more susceptible to bias in federated settings [92], as discussed in Chapter 3. By comparing these two strategies, the aim is to understand how the personalization of different components affects performance in the presence of statistical heterogeneity.

- **Phase 4:** Full personalization (FP). This phase encompasses `FP` training, as introduced in Section 4.1.2, where all, or a subset, of the model parameters are locally fine-tuned and there is no sharing across clients.

  - In all the experiments that include this phase, all the clients are locally fine-tuned for 25 epochs.
  - By default, the model is fine-tuned using the SGD optimizer. As an alternative, the `Ditto` [167] regularization term has been tested, which penalizes personalized updates that diverge from the global model.
  - The experiments performing `FP` on all the model parameters are indicated with `FP(`f`)`. Instead, the experiments that fine-tune only the parameters

of the feature extractor or of the classifier are denoted by `FP(`φ`)` and `FP(`ψ`)`, respectively.

Since the experiments can include at most four phases, explicitly detailing all four phases in the text can be somewhat challenging for the reader and redundant. Therefore, rather than describing a specific experiment as one where the classifier is initialized with `Fed3R` parameters, fine-tuned with `FedAvg`, using `FedAlt` as the `PP` strategy, and `FT(`ψ`)` as the `FP` strategy, it is simply expressed as `Fed3R + FedAvg + FedAlt + FT(`ψ`)`.

Additionally, to indicate that the columns of the classifier parameters have been pruned using `OLL`, the notation `OLL(`$\cdot$`)` is used, where $\cdot$ can represent any combination of phases. For instance, if we want to specify a strategy that includes `Fed3R` in Phase 1 and `FedAvg` in Phase 2, followed by pruning the classifier parameters with `OLL` and then personalizing the feature extractors of each client based on their local distribution, we would write it as `OLL(Fed3R + FedAvg) + FT(`φ`)`.

If the classifier is pruned using `OLL`, then the `FP` learning rate is multiplied by 0.1. Therefore, if the classifier is initialized using the `Fed3R` parameters and then pruned using `OLL`, the default learning rate is multiplied by 0.1 two times, resulting in a learning rate of 0.001. For all the PFL experiments, the chosen batch size is 64.

In the following of this section, iNaturalist and Google Landmarks indicate the iNaturalist-Pers-Geo-100 and Google Landmarks-Pers-Users-160k, respectively.

4.3.2 Communication and computation costs estimation

This section complements the total communication costs and average computational costs per client for the FL methods presented in Section 3.9.2, including the costs for the PFL methods considered for the experiments of this section. The communication costs are shown in Table 4.2, while the computation costs are shown in Table 4.3. The $\bar{F}_{\hat{\psi}}$ symbol represents the average forward cost though the `OLL` classifiers of all the clients, which has a dimension $d \times \bar{C}$, with $\bar{C} \ll C$. Therefore, $\bar{F}_{\hat{\psi}} \ll F_{\psi}$.

Table 4.2: Estimation of communication costs of the PFL methods per each sampled client in one round included in the experiments of this section. The FP methods are not included, as they all have no communication costs.

Method	**Downstream**	**Upstream**	**Total**
FedSim(φ)	dC	dC	$2dC$
FedSim(ψ)	b	b	$2b$
FedAlt(φ)	dC	dC	$2dC$
FedAlt(ψ)	b	b	$2b$
pFedMe(φ)	dC	dC	$2dC$
pFedMe(ψ)	b	b	$2b$

Table 4.3: Estimation of average computation costs per client per round per client of all the methods included in the experiments of this section, expressed in number of parameters.

Method	$\mathcal{T}/n_k$
FedSim(φ)	$3EF_f$
FedSim(ψ)	$3E(F_\varphi + 3F_\psi)$
FedAlt(φ)	$3EF_f$
FedAlt(ψ)	$3E(F_\varphi + 3F_\psi)$
pFedMe(φ)	$6EF_f$
pFedMe(ψ)	$6E(F_\varphi + 3F_\psi)$
FP(f)	$3EF_f$
FP(φ)	$3EF_f$
FP(ψ)	$E(F_\varphi + 3F_\psi)$
FP(f) + Ditto	$3EF_f$
FP(φ) + Ditto	$3EF_f$
FP(ψ) + Ditto	$E(F_\varphi + 3F_\psi)$
OLL(f)	$3E(F_\varphi + \bar{F}_{\hat{\psi}})$
OLL(φ)	$3E(F_\varphi + \bar{F}_{\hat{\psi}})$
OLL(ψ)	$E(F_\varphi + 3\bar{F}_{\hat{\psi}})$

4.3.3 Experiments with partial personalization

This section shows how the initialization of the classifier parameters using Fed3R optimal parameters W^* yields benefits in PFL settings comparable to those shown in Section 3.9 for the FL setting. Since pFedMe consistently provides lower WMA compared to FedSim and FedAlt, which always perform similarly in all the experiments,

Table 4.4: Comparison among personal and shared parameters selection for the PP experiments using FedSim as PP algorithm. ψ indicates the classifier; FedSim(ψ) indicates training with FedSim, with the classifier parameters as the personal parameters and the feature extractor parameters as the shared parameters; FedSim(φ) indicates training with FedSim, with the feature extractor parameters as the personal parameters and the classifier parameters as the shared parameters. Results are expressed in WMA (%).

Dataset	ψ init	FT alg	FT acc	FedSim(ψ)	FedSim(φ)
iNaturalist	✗	FedAvg	46.3 ± 0.4	70.9 ± 0.2	67.4 ± 0.2
	✗	FedAvg-1k	30.0 ± 0.3	40.7 ± 0.1	52.1 ± 0.1
	Fed3R	✗	58.0 ± 0.0	75.1 ± 0.0	72.9 ± 0.1
	Fed3R	FedAvg	67.3 ± 0.2	**78.5 ± 0.1**	76.9 ± 0.1
	Fed3R	FedAvg-1k	63.5 ± 0.1	77.3 ± 0.1	75.5 ± 0.1
	Fed3R	FedAvg(φ)	63.1 ± 0.1	77.5 ± 0.0	77.2 ± 0.1
Google Landmarks	✗	FedAvg	61.2 ± 1.0	79.4 ± 0.0	79.7 ± 0.1
	✗	FedAvg-1k	47.6 ± 1.0	66.5 ± 0.2	74.1 ± 0.0
	Fed3R	✗	49.9 ± 0.0	72.9 ± 0.1	74.1 ± 0.1
	Fed3R	FedAvg	68.1 ± 0.4	79.9 ± 0.1	**80.4 ± 0.0**
	Fed3R	FedAvg-1k	63.0 ± 0.3	77.0 ± 0.3	78.6 ± 0.1
	Fed3R	FedAvg(φ)	68.2 ± 0.1	79.9 ± 0.1	**80.4 ± 0.0**

here the focus is solely on the differences between FedSim(φ) and FedSim(ψ), to emphasize the difference between the two PP strategies.

Table 4.4 shows the WMA after the PP phase with FedSim(ψ) of different strategies for Phases 1 and 2, on both the Google Landmarks and iNaturalist dataset splits.

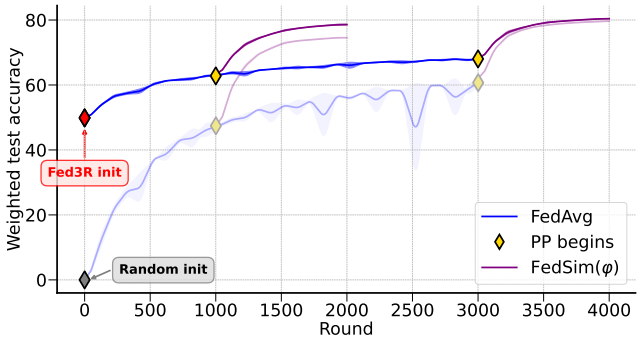
PP improves the final WMA of the FT experiments. Overall, PP significantly improves the original WMA. For example, the WMA of the FedAvg experiment without the Fed3R initialization on Google Landmarks increases from 61.2% to 79.7% after the PP phase using FedSim(φ).

Comparison between PP(φ) and PP(ψ) strategies. On Google Landmarks, FedSim(φ) consistently outperforms FedSim(ψ). Conversely, on most of the iNaturalist experiments, FedSim(ψ) outperforms FedSim(φ), with the sole exception of the FedAvg-1k experiment without Fed3R initialization, where FedSim(φ) WMA

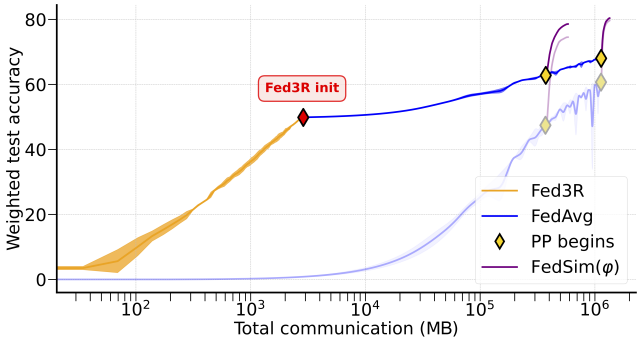
is 12% points above the WMA of FedSim(ψ). The difference between the results of Google Landmarks and iNaturalist can be attributed to the data recency bias problem. This issue causes the local classifiers of the clients to diverge as they are optimized for their specific local distributions. In the more challenging and heterogeneous iNaturalist split, maintaining the classifier locally while aggregating the parameters of the feature extractor helps prevent destructive interference. Conversely, in the less heterogeneous Google Landmarks scenario, specializing the feature extractor on the target task while keeping the classifier global enhances the quality of the feature extractor. This approach allows it to benefit from the knowledge obtained from all the clients.

PP(ψ) is robust to the quality of the FT classifier. The experiments reveal an interesting trend in the PP phase. Indeed, the WMA difference between adopting FT(f) or FT(φ) as the strategy for Phase 2 becomes negligible. For instance, on iNaturalist, the WMA difference between Fed3R + FedAvg + FedSim(ψ) and Fed3R + FedAvg(φ) + FedSim(ψ) is only 1%, whereas the difference between Fed3R + FedAvg and Fed3R + FedAvg(φ) (without PP) is almost 4%. This suggests that when the classifier parameters are the sole local parameters of Phase 3 (FedSim(ψ)), the local classifiers can effectively specialize regardless of the initial classifier state. Whether the initial classifier is the robust Fed3R classifier or a fine-tuned version potentially affected by data recency bias and destructive interference appears to have minimal impact on the final personalized performance. This finding highlights the ability of PFL methods to overcome the limitations of previous training phases and achieve effective personalization.

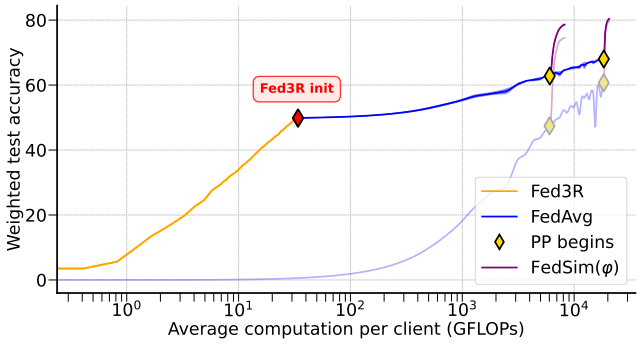
When statistical heterogeneity is high, Fed3R initialization makes FT Phase unnecessary. Results in Table 4.4 show that in highly heterogeneous settings such as iNaturalist, the FT phase can be avoided if the classifier is initialized with Fed3R parameters. Indeed, FedSim(ψ) with Fed3R initialization and no fine-tuning achieves 75.1% WMA, compared with the same PP algorithm but with FedAvg as FT algorithm and no Fed3R initialization, which reaches only 70.9%. This result underscores that only 1k rounds of PP with Fed3R initialization are sufficient to surpass the WMA with 3k rounds of FT and 1k rounds of PP, therefore saving 75% of the total training rounds. Additionally, with enough time and resources available, the full pipeline with Fed3R initialization, FT Phase, and PP Phase is the



(a)



(b)



(c)

Figure 4.5: (a) PP performance, (b) communication, and (c) computation costs with and without Fed3R classifier initialization, and with 1k or 3k rounds for the FT phase (FedAvg), in the Google Landmarks scenario. For this visualization, it has been chosen the FedSim(φ) algorithm for the PP phase. Less intense colors are associated with the baseline algorithms without Fed3R initialization.

one that provides the best results, higher than the results of FT + PP without Fed3R initialization, in both the settings. Once again, these results stress the importance of having a robust classifier initialization using Fed3R in scenarios with high statistical heterogeneity.

Table 4.5: Final WMA for the Google Landmarks PP experiments.

ψ init	FT alg	FT acc	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	61.2 ± 1.0	79.4 ± 0.0	79.5 ± 0.0	66.7 ± 0.0	79.7 ± 0.1	79.7 ± 0.0	67.0 ± 0.2
✗	FedAvg-1k	47.6 ± 1.0	66.5 ± 0.2	66.6 ± 0.2	57.2 ± 0.1	74.1 ± 0.0	74.2 ± 0.0	56.1 ± 0.2
Fed3R	✗	49.9 ± 0.0	72.9 ± 0.1	72.2 ± 0.3	55.0 ± 0.2	74.1 ± 0.1	73.2 ± 0.2	53.9 ± 0.4
Fed3R	FedAvg	68.1 ± 0.4	79.9 ± 0.1	79.4 ± 0.1	69.7 ± 0.4	**80.4 ± 0.0**	80.3 ± 0.1	69.7 ± 0.0
Fed3R	FedAvg-1k	63.0 ± 0.3	77.0 ± 0.3	76.2 ± 1.0	45.5 ± 0.0	78.6 ± 0.1	78.4 ± 0.1	41.2 ± 0.3
Fed3R	FedAvg(φ)	68.2 ± 0.1	79.9 ± 0.1	79.5 ± 0.1	70.0 ± 0.0	**80.4 ± 0.0**	**80.4 ± 0.0**	69.6 ± 0.0

Table 4.6: Final WMA for the iNaturalist PP experiments.

ψ init	FT alg	FT acc	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	46.3 ± 0.4	70.9 ± 0.2	71.0 ± 0.1	52.3 ± 0.1	67.4 ± 0.2	67.9 ± 0.1	53.0 ± 0.0
✗	FedAvg-1k	30.0 ± 0.3	40.7 ± 0.1	40.7 ± 0.1	38.3 ± 0.1	52.1 ± 0.1	53.1 ± 0.4	37.9 ± 0.0
Fed3R	✗	58.0 ± 0.0	75.1 ± 0.0	73.8 ± 0.1	59.8 ± 0.1	72.9 ± 0.1	74.4 ± 0.1	59.0 ± 0.0
Fed3R	FedAvg	67.3 ± 0.2	**78.5 ± 0.1**	77.9 ± 0.1	68.8 ± 0.0	76.9 ± 0.1	78.0 ± 0.0	68.3 ± 0.0
Fed3R	FedAvg-1k	63.5 ± 0.1	77.3 ± 0.1	76.3 ± 0.1	42.1 ± 0.1	75.5 ± 0.1	76.7 ± 0.1	41.7 ± 0.1
Fed3R	FedAvg(φ)	63.1 ± 0.1	77.5 ± 0.0	76.6 ± 0.0	64.5 ± 0.0	77.2 ± 0.1	76.9 ± 0.1	64.6 ± 0.0

Table 4.7: Google Landmarks PP total communication costs to achieve 70% WMA. N/A indicates that the experiment never reaches the target WMA.

ψ init	FT alg	FT costs	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	N/A	1.2TB ± 720.7MB	1.2TB ± 720.7MB	N/A	1.2TB ± 98.1MB	1.2TB ± 169.8MB	N/A
✗	FedAvg-1k	N/A	413.3GB ± 1.4GB	411.7GB ± 1.3GB	N/A	448.2GB ± 2.4GB	426.6GB ± 4.9GB	N/A
Fed3R	✗	N/A	**39.6GB ± 3.2GB**	48.5GB ± 2.6GB	N/A	100.3GB ± 2.7GB	77.4GB ± 1.2GB	N/A
Fed3R	FedAvg	N/A	1.1TB ± 851.4MB	1.1TB ± 819.9MB	N/A	1.1TB ± 196.1MB	1.1TB ± 196.1MB	N/A
Fed3R	FedAvg-1k	N/A	392.2GB ± 280.9MB	392.6GB ± 280.9MB	N/A	407.0GB ± 1.1GB	397.1GB ± 588.3MB	N/A
Fed3R	FedAvg(φ)	N/A	519.8GB ± 265.8MB	519.9GB ± 283.8MB	669.3GB ± 4.4GB	523.6GB ± 107.5MB	521.8GB ± 107.5MB	N/A

Fed3R initialization reduces the necessary rounds for the FT Phase in mildly heterogeneous scenarios. Google Landmarks experiments do not exhibit the same desirable results exposed in the previous paragraph for iNaturalist, where Fed3R initialization allows avoiding Phase 2 phase at all. Nevertheless, Figure 4.5 shows that running the FT phase for only 1k rounds starting from Fed3R initialization and then performing the PP phase is sufficient to achieve performances comparable to the best results that can be obtained with the same strategy but with 3k rounds in Phase 2, therefore saving two-thirds of the FT rounds and, consequently, communication and computational costs. Specifically, Fed3R + FedAvg-1k + FedSim(φ) reaches 78.6% WMA, only 1 point below FedAvg + FedSim(φ) without Fed3R initialization and 2 points below Fed3R + FedAvg + FedSim(φ), but saving 2k rounds compared to these two strategies.

Table 4.8: iNaturalist PP total communication costs to achieve 70% WMA. N/A indicates that the experiment never reaches the target WMA.

ψ init	FT alg	FT costs	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	N/A	1.0TB ± 3.8GB	1.0TB ± 3.5GB	N/A	N/A	N/A	N/A
✗	FedAvg-1k	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Fed3R	✗	N/A	54.0GB ± 996.3MB	66.5GB ± 81.1MB	N/A	78.0GB ± 1.7GB	**53.0GB ± 303.7MB**	N/A
Fed3R	FedAvg	N/A	900.4GB ± 720.9MB	902.6GB ± 401.9MB	N/A	904.1GB ± 623.3MB	898.9GB ± 406.6MB	N/A
Fed3R	FedAvg-1k	N/A	324.6GB ± 292.3MB	329.8GB ± 405.4MB	N/A	336.3GB ± 2.0GB	323.3GB ± 773.3MB	N/A
Fed3R	FedAvg(φ)	N/A	545.6GB ± 243.2MB	548.9GB ± 1.5GB	N/A	542.2GB ± 861.1MB	540.2GB ± 998.9MB	N/A

Table 4.9: Google Landmarks PP average computation per client to achieve 70% WMA. N/A indicates that the experiment never reaches the target WMA.

ψ init	FT alg	FT costs	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	N/A	19.2 TFLOPs ± 25.9 GFLOPs	19.1 TFLOPs ± 25.9 GFLOPs	N/A	18.7 TFLOPs ± 985.6 MFLOPs	18.7 TFLOPs ± 1.7 GFLOPs	N/A
✗	FedAvg-1k	N/A	7.4 TFLOPs ± 50.5 GFLOPs	7.3 TFLOPs ± 46.9 GFLOPs	N/A	6.9 TFLOPs ± 23.9 GFLOPs	6.6 TFLOPs ± 49.0 GFLOPs	N/A
Fed3R	✗	N/A	1.4 TFLOPs ± 113.6 GFLOPs	1.7 TFLOPs ± 93.7 GFLOPs	N/A	**1.0 TFLOPs ± 27.6 GFLOPs**	782.8 GFLOPs ± 12.0 GFLOPs	N/A
Fed3R	FedAvg	N/A	18.7 TFLOPs ± 30.6 GFLOPs	18.7 TFLOPs ± 29.4 GFLOPs	N/A	18.6 TFLOPs ± 2.0 GFLOPs	18.6 TFLOPs ± 2.0 GFLOPs	N/A
Fed3R	FedAvg-1k	N/A	6.6 TFLOPs ± 10.1 GFLOPs	6.6 TFLOPs ± 10.1 GFLOPs	N/A	6.4 TFLOPs ± 10.7 GFLOPs	6.3 TFLOPs ± 5.9 GFLOPs	N/A
Fed3R	FedAvg(φ)	N/A	18.7 TFLOPs ± 9.5 GFLOPs	18.7 TFLOPs ± 10.2 GFLOPs	29.5 TFLOPs ± 317.8 GFLOPs	18.6 TFLOPs ± 1.7 GFLOPs	18.6 TFLOPs ± 1.7 GFLOPs	N/A

Table 4.10: iNaturalist PP average computation per client to achieve 70% WMA. N/A indicates that the experiment never reaches the target WMA.

ψ init	FT alg	FT costs	FedSim(ψ)	FedAlt(ψ)	pFedMe(ψ)	FedSim(φ)	FedAlt(φ)	pFedMe(φ)
✗	FedAvg	N/A	1.7 TFLOPs ± 9.7 GFLOPs	1.7 TFLOPs ± 8.9 GFLOPs	N/A	N/A	N/A	N/A
✗	FedAvg-1k	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Fed3R	✗	N/A	137.7 GFLOPs ± 2.6 GFLOPs	170.0 GFLOPs ± 209.2 MFLOPs	N/A	93.6 GFLOPs ± 2.1 GFLOPs	**63.5 GFLOPs ± 365.6 MFLOPs**	N/A
Fed3R	FedAvg	N/A	1.4 TFLOPs ± 1.7 GFLOPs	1.4 TFLOPs ± 911.7 MFLOPs	N/A	1.4 TFLOPs ± 760.5 MFLOPs	1.3 TFLOPs ± 487.5 MFLOPs	N/A
Fed3R	FedAvg-1k	N/A	517.5 GFLOPs ± 754.1 MFLOPs	530.8 GFLOPs ± 1.0 GFLOPs	N/A	492.2 GFLOPs ± 2.4 GFLOPs	476.6 GFLOPs ± 930.8 MFLOPs	N/A
Fed3R	FedAvg(φ)	N/A	1.4 TFLOPs ± 627.5 MFLOPs	1.4 TFLOPs ± 4.0 GFLOPs	N/A	1.4 TFLOPs ± 1.0 GFLOPs	1.4 TFLOPs ± 1.2 GFLOPs	N/A

Complete results. Tables 4.5 and 4.6 provide the complete results for the PP experiments. Moreover, Tables 4.7 and 4.9 present the communication and computation costs to achieve 70% WMA for the same experiments in the Google Landmarks setting, while Tables 4.8 and 4.10 show the communication and computation costs to achieve 70% WMA for the same experiments in the iNaturalist setting. As it is possible to observe, Fed3R + PP experiments are the cheapest in terms of costs, as the expensive FT phase is not performed.

4.3.4 Experiments with full personalization

This section presents the results of the FP experiments, focusing on the impact of Fed3R initialization and the trade-offs of different intermediate phases. At first, the Ditto experiments are omitted for clarity, as they generally show negligible improvements over standard SGD optimization. Complete results are presented at the end of this section.

Table 4.11: Comparison between FP strategies with different choices for the initialization, FT, and PP phase, on both iNaturalist and Google Landmarks. ψ indicates the classifier; φ indicates the feature extractor; f indicates the whole model; FP$(\cdot)$ indicates full personalization of the parameters associated with $\cdot$, while the other parameters (if any) are fixed.

ψ init	FT alg	PP alg	iNaturalist				Google Landmarks			
			✗	FP(ψ)	FP(φ)	FP(f)	✗	FP(ψ)	FP(φ)	FP(f)
Fed3R	✗	✗	58.0 ±0.0	77.4 ±0.1	78.6 ±0.2	77.1 ±0.1	49.9 ±0.0	68.4 ±0.2	68.9 ±0.1	69.7 ±0.1
✗	FedAvg	✗	46.3 ±0.4	75.2 ±0.1	75.5 ±0.2	78.1 ±0.1	61.2 ±1.0	78.3 ±0.1	78.3 ±0.2	80.0 ±0.0
Fed3R	FedAvg	✗	67.3 ±0.2	83.8 ±0.0	84.1 ±0.2	83.3 ±0.0	68.1 ±0.4	80.0 ±0.0	79.5 ±0.1	80.0 ±0.1
Fed3R	✗	FedSim(φ)	72.9 ±0.1	83.4 ±0.1	84.8 ±0.1	82.8 ±0.1	74.1 ±0.1	73.1 ±0.2	74.5 ±0.1	73.0 ±0.1
✗	FedAvg	FedSim(φ)	67.4 ±0.2	82.1 ±0.1	79.3 ±0.1	82.4 ±0.1	79.7 ±0.1	80.2 ±0.0	80.0 ±0.0	80.3 ±0.1
Fed3R	FedAvg	FedSim(φ)	76.9 ±0.1	86.8 ±0.1	**87.4** ±0.1	86.4 ±0.1	80.4 ±0.0	80.3 ±0.1	**80.7** ±0.1	80.2 ±0.1

Table 4.11 summarizes the FP experiments, with FedSim(φ) selected as the PP algorithm due to its slight advantage in WMA compared to other PP strategies like FedSim(ψ), FedAlt(φ), and FedAlt(ψ).

Robust Fed3R initialization improves final FP WMA. The results in Table 4.11 clearly demonstrate the positive impact of Fed3R initialization on FP performance. Across both the Google Landmarks and iNaturalist scenarios, all the experiments with Fed3R initialization consistently achieve higher WMA than their counterpart without Fed3R initialization. For example, Fed3R + FedAvg + FedSim(φ) + FT(φ) achieves 87.4% WMA on iNaturalist and 80.7% WMA on Google Landmarks, compared with FedAvg + FedSim(φ) + FT(φ) that achieves only 79.3% on iNaturalist and 80.0% on Google Landmarks. Similarly, Fed3R + FedAvg + FT(φ) achieves 84.1% WMA on iNaturalist and 79.5% WMA on Google Landmarks, compared with FedAvg + FT(φ) that achieves only 75.5% on iNaturalist and 78.3% on Google Landmarks.

PP and FP phases are not both necessary in mildly heterogeneous settings. Interestingly, the necessity of combining PP and FP appears to depend on the severity of data heterogeneity. In the mildly heterogeneous Google Landmarks scenario, the performance gains from combining both phases are minimal. For instance, Fed3R + FedAvg + FedSim(φ) + FT(φ) achieves 80.7% WMA, only slightly higher than the 79.5% WMA of Fed3R + FedAvg + FT(φ). Conversely, on the more heterogeneous iNaturalist dataset, combining PP and FP consistently leads to significant improvements. This suggests that in highly heterogeneous settings, both

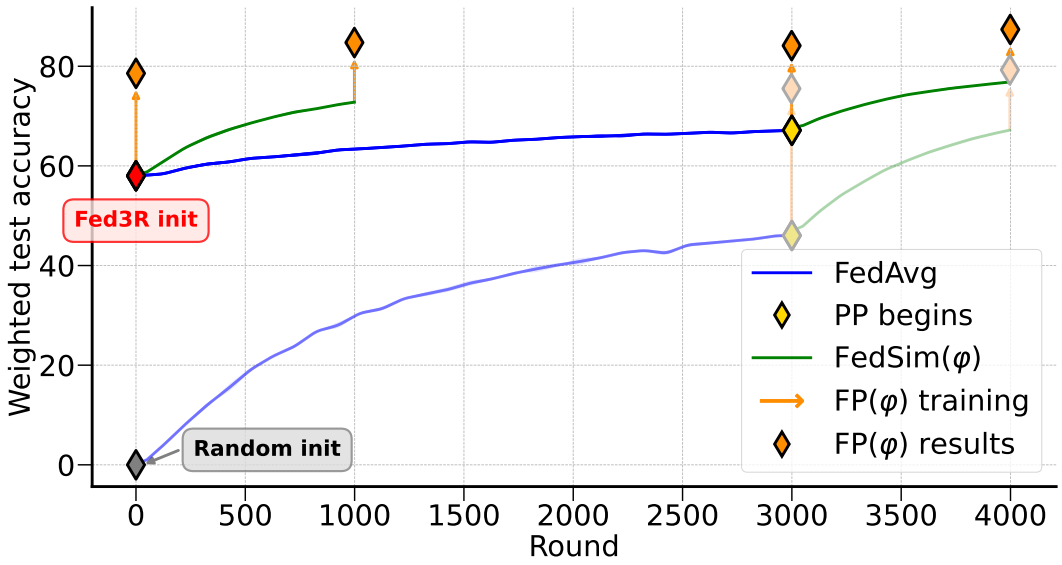


Figure 4.6: Comparison of FP results after different possibilities for the first three phases, for the iNaturalist setting, using FedSim(φ) and FP(φ). Less intense colors are associated with the baseline algorithms without Fed3R initialization.

forms of personalization are necessary to fully exploit the potential of personalized models. Given that the PP phase is generally more resource-intensive than FP, it may be more efficient to focus solely on FP in mildly heterogeneous scenarios like Google Landmarks as it enables comparable performance with reduced computational and communication overhead.

In highly heterogeneous scenarios, Fed3R initialization + PP + FP is the strategy with the highest performance-costs ratio. Section 4.3.3 shows that Fed3R initialization allows avoiding the FT phase in highly statistically heterogeneous scenarios, saving 75% of the total rounds and costs. As Figure 4.6 shows for the iNaturalist dataset, this finding holds also when including the final FP phase In particular, Fed3R + FedSim(φ) + FP(φ) achieves only 2.6 percentage points of WMA less than Fed3R + FedAvg + FedSim(φ) + FP(φ), **but saves 3000 rounds of training.** This substantial reduction in training rounds translates to significant savings in communication and computational costs, making Fed3R a highly efficient and scalable approach for achieving personalization in challenging federated settings. These results reinforce the value of Fed3R as a robust initialization that not only accelerates convergence but also enables substantial cost reduction by eliminating the need for expensive federated fine-tuning.

Table 4.12: Google Landmarks FP results without PP phase.

ψ init	FT alg	FT acc	FP(ψ)		FP(φ)		FP(f)	
			SGD	Ditto	SGD	Ditto	SGD	Ditto
✗	FedAvg	61.2 ± 1.0	78.3 ± 0.1	78.8 ± 0.0	78.3 ± 0.2	78.4 ± 0.1	80.0 ± 0.0	80.0 ± 0.0
✗	FedAvg-1k	47.6 ± 1.0	72.6 ± 0.1	73.0 ± 0.2	72.5 ± 0.1	71.3 ± 0.3	71.7 ± 0.3	71.7 ± 0.0
Fed3R	✗	49.9 ± 0.0	68.4 ± 0.2	68.5 ± 0.1	68.9 ± 0.1	68.7 ± 0.1	69.7 ± 0.1	69.6 ± 0.2
Fed3R	FedAvg	68.1 ± 0.4	80.0 ± 0.0	79.4 ± 0.1	79.5 ± 0.1	79.4 ± 0.1	80.0 ± 0.1	**80.2 ± 0.1**
Fed3R	FedAvg-1k	63.0 ± 0.3	77.6 ± 0.2	77.7 ± 0.1	76.9 ± 0.1	76.8 ± 0.1	77.4 ± 0.2	77.6 ± 0.2
Fed3R	FedAvg(φ)	68.2 ± 0.1	80.1 ± 0.1	**80.2 ± 0.0**	79.5 ± 0.0	79.4 ± 0.0	80.0 ± 0.1	**80.2 ± 0.1**

Table 4.13: iNaturalist FP results without PP phase.

ψ init	FT alg	FT acc	FP(ψ)		FP(φ)		FP(f)	
			SGD	Ditto	SGD	Ditto	SGD	Ditto
✗	FedAvg	46.3 ± 0.4	75.2 ± 0.1	76.7 ± 0.1	75.5 ± 0.2	76.8 ± 0.1	78.1 ± 0.1	83.5 ± 0.1
✗	FedAvg-1k	30.0 ± 0.3	68.0 ± 0.0	69.1 ± 0.1	67.1 ± 0.0	66.8 ± 0.0	66.8 ± 0.0	68.1 ± 0.1
Fed3R	✗	58.0 ± 0.0	77.4 ± 0.1	77.5 ± 0.1	78.6 ± 0.2	78.6 ± 0.2	77.1 ± 0.1	77.3 ± 0.1
Fed3R	FedAvg	67.3 ± 0.2	83.8 ± 0.0	84.0 ± 0.1	84.1 ± 0.2	**84.2 ± 0.1**	83.3 ± 0.0	83.5 ± 0.1
Fed3R	FedAvg-1k	63.5 ± 0.1	81.8 ± 0.1	82.0 ± 0.1	82.3 ± 0.1	82.3 ± 0.2	81.4 ± 0.1	81.5 ± 0.1
Fed3R	FedAvg(φ)	63.1 ± 0.1	81.6 ± 0.1	81.7 ± 0.0	80.5 ± 0.0	80.6 ± 0.0	79.5 ± 0.1	79.6 ± 0.0

Fed3R initialization + FP final WMA is comparable to the federated FT WMA but drastically reduces the required communication. As Figure 4.6 and Table 4.11 show, the Fed3R + FP(φ) strategy reaches 78.6% final WMA on iNaturalist, and the Fed3R + FP(f) strategy achieves 69.7% final WMA on Google Landmarks, with no communication rounds required, necessitating only the limited communication costs of Fed3R. In contrast, federated FT without Fed3R initialization achieves only 46.3% and 61.2% on iNaturalist and Google Landmarks, respectively, while FT with Fed3R initialization reaches 67.3% on iNaturalist and 68.1% on Google Landmarks. Hence, Fed3R + FP demonstrates to be the best strategy when the allowed communication budget is particularly scarce.

Complete results. Tables 4.12 and 4.13 provide all the results for FP without PP phase. Moreover,Tables 4.14 to 4.17 present the results for the full pipeline, with all the 4 phases. The best result is achieved by the Fed3R + FedAvg + FedSim(ψ) + FP(φ) experiment, with a final WMA of 88.1%. As already observed in other contexts, the best results are achieved with Fed3R initialization. Interestingly, the best strategy is to keep the classifier parameters local first during the PP phase, and

Table 4.14: Google Landmarks FP results with PP(ψ) phase.

ψ init	FT alg	FedSim(ψ)				FedAlt(ψ)			
		✗	FP(ψ)	FP(φ)	FP(f)	✗	FP(ψ)	FP(φ)	FP(f)
✗	FedAvg	79.4 ±0.0	79.7 ±0.0	**80.4** ±0.0	**80.4** ±0.0	79.5 ±0.0	79.7 ±0.0	80.2 ±0.0	80.2 ±0.0
✗	FedAvg-1k	72.1 ±0.1	72.9 ±0.0	72.9 ±0.0	73.0 ±0.1	72.4 ±0.1	72.7 ±0.1	72.7 ±0.2	72.7 ±0.2
Fed3R	✗	72.9 ±0.1	72.9 ±0.1	73.0 ±0.1	73.5 ±0.1	72.2 ±0.3	72.4 ±0.2	72.9 ±0.0	72.3 ±0.1
Fed3R	FedAvg	79.9 ±0.1	79.8 ±0.1	80.1 ±0.0	79.5 ±0.2	79.4 ±0.1	79.5 ±0.1	79.8 ±0.2	79.2 ±0.0
Fed3R	FedAvg-1k	77.0 ±0.3	77.0 ±0.2	77.5 ±0.1	76.9 ±0.2	76.2 ±1.0	76.4 ±0.6	77.2 ±0.2	76.5 ±0.1
Fed3R	FedAvg(φ)	79.9 ±0.1	79.8 ±0.1	80.1 ±0.0	79.5 ±0.2	79.5 ±0.1	79.5 ±0.1	79.8 ±0.2	79.2 ±0.0

Table 4.15: iNaturalist FP results with PP(ψ) phase.

ψ init	FT alg	FedSim(ψ)				FedAlt(ψ)			
		✗	FP(ψ)	FP(φ)	FP(f)	✗	FP(ψ)	FP(φ)	FP(f)
✗	FedAvg	70.9 ±0.2	82.5 ±0.2	81.8 ±0.2	80.7 ±3.7	71.0 ±0.1	82.4 ±0.0	81.7 ±0.3	83.3 ±0.1
✗	FedAvg-1k	66.2 ±0.1	79.8 ±0.2	73.0 ±0.1	80.9 ±0.2	65.6 ±0.1	79.8 ±0.0	72.7 ±0.1	80.9 ±0.2
Fed3R	✗	75.1 ±0.0	85.6 ±0.2	86.4 ±0.1	85.3 ±0.1	73.8 ±0.1	85.0 ±0.1	85.7 ±0.2	84.6 ±0.1
Fed3R	FedAvg	78.5 ±0.1	87.7 ±0.1	**88.1** ±0.1	87.4 ±0.1	77.9 ±0.1	87.3 ±0.1	88.0 ±0.1	87.0 ±0.1
Fed3R	FedAvg-1k	77.3 ±0.1	86.9 ±0.1	87.4 ±0.1	86.9 ±0.1	76.3 ±0.1	86.4 ±0.1	87.1 ±0.1	86.1 ±0.1
Fed3R	FedAvg(φ)	77.5 ±0.0	87.0 ±0.2	87.5 ±0.1	86.4 ±0.2	76.6 ±0.0	86.6 ±0.1	87.1 ±0.1	85.8 ±0.1

Table 4.16: Google Landmarks FP results with PP(φ) phase.

ψ init	FT alg	FedSim(φ)				FedAlt(φ)			
		✗	FP(ψ)	FP(φ)	FP(f)	✗	FP(ψ)	FP(φ)	FP(f)
✗	FedAvg	79.7 ±0.1	80.2 ±0.0	80.0 ±0.0	80.3 ±0.1	79.7 ±0.0	80.2 ±0.0	80.0 ±0.0	80.3 ±0.0
✗	FedAvg-1k	74.5 ±0.1	75.3 ±0.0	75.6 ±0.1	75.6 ±0.1	74.0 ±0.2	74.5 ±0.1	74.9 ±0.1	74.8 ±0.1
Fed3R	✗	74.1 ±0.1	73.1 ±0.2	74.5 ±0.1	73.0 ±0.1	73.2 ±0.2	71.6 ±0.1	73.7 ±0.1	72.2 ±0.1
Fed3R	FedAvg	80.4 ±0.0	80.3 ±0.1	**80.7** ±0.1	80.2 ±0.1	80.3 ±0.1	79.8 ±0.1	80.5 ±0.1	79.8 ±0.1
Fed3R	FedAvg-1k	78.6 ±0.1	78.1 ±0.0	78.9 ±0.1	78.0 ±0.1	78.4 ±0.1	77.4 ±0.0	78.6 ±0.1	77.5 ±0.1
Fed3R	FedAvg(φ)	80.4 ±0.0	80.3 ±0.1	80.6 ±0.0	80.1 ±0.1	80.4 ±0.0	79.9 ±0.1	80.6 ±0.1	79.8 ±0.1

Table 4.17: iNaturalist FP results with PP(φ) phase.

ψ init	FT alg	FedSim(φ)				FedAlt(φ)			
		✗	FP(ψ)	FP(φ)	FP(f)	✗	FP(ψ)	FP(φ)	FP(f)
✗	FedAvg	67.4 ±0.2	82.1 ±0.1	79.3 ±0.1	82.4 ±0.1	67.9 ±0.1	82.1 ±0.1	79.4 ±0.1	82.5 ±0.2
✗	FedAvg-1k	63.6 ±0.2	79.2 ±0.1	76.4 ±0.2	80.2 ±0.1	66.7 ±0.3	79.5 ±0.1	78.1 ±0.1	81.1 ±0.0
Fed3R	✗	72.9 ±0.1	83.4 ±0.1	84.8 ±0.1	82.8 ±0.1	74.4 ±0.1	83.5 ±0.1	85.2 ±0.2	83.1 ±0.2
Fed3R	FedAvg	76.9 ±0.1	86.8 ±0.1	87.4 ±0.1	86.4 ±0.1	78.0 ±0.0	87.1 ±0.1	**87.8** ±0.1	86.7 ±0.1
Fed3R	FedAvg-1k	75.5 ±0.1	85.8 ±0.0	86.5 ±0.1	85.3 ±0.1	76.7 ±0.1	86.0 ±0.1	86.9 ±0.1	85.7 ±0.1
Fed3R	FedAvg(φ)	77.2 ±0.1	86.5 ±0.0	86.1 ±0.0	84.9 ±0.1	76.9 ±0.1	86.1 ±0.1	86.1 ±0.0	84.6 ±0.1

then locally fine-tune the feature extractor only during the FP phase while keeping the classifier fixed.

Table 4.18: FP results after OLL, compared with the FP results without OLL, on both the iNaturalist and Google Landmarks datasets. φ indicates the feature extractor; ψ indicates the classifier; f indicates the whole model; FP($\cdot$) indicates full personalization of the parameters associated with $\cdot$, while the other parameters (if any) are fixed. The symbol † means that the displayed results are the best among the following 4 PP strategies (Phase 3): FedSim(ψ), FedSim(φ), FedAlt(ψ), and FedSim(φ). Conversely, the results using OLL do not include Phase 3, because OLL applied on local classifiers from Phase 1 and 2 is already sufficient to surpass the baselines by a large margin, even without Phase 3.

Dataset	ψ init	FT alg	With OLL				Without OLL			
			✗	FP(ψ)	FP(φ)	FP(f)	✗†	FP(ψ)†	FP(φ)†	FP(f)†
iNaturalist	✗	FedAvg	76.9 ±0.1	89.5 ±0.1	88.5 ±0.1	88.6 ±0.2	71.0 ±0.1	82.5 ±0.2	81.8 ±0.2	83.3 ±0.1
	✗	FedAvg-1k	74.9 ±0.1	76.6 ±0.2	75.4 ±0.1	72.7 ±0.1	66.7 ±0.3	79.8 ±0.2	78.1 ±0.1	81.1 ±0.0
	Fed3R	✗	75.4 ±0.0	88.6 ±0.1	88.3 ±0.1	87.9 ±0.1	75.1 ±0.0	85.6 ±0.2	86.4 ±0.1	85.3 ±0.1
	Fed3R	FedAvg	80.5 ±0.1	**92.5 ±0.1**	92.3 ±0.1	91.1 ±0.1	78.5 ±0.1	87.7 ±0.1	88.1 ±0.1	87.4 ±0.1
	Fed3R	FedAvg-1k	78.9 ±0.1	88.2 ±0.0	88.5 ±0.1	91.5 ±0.1	77.3 ±0.1	86.9 ±0.1	87.4 ±0.1	86.9 ±0.1
	Fed3R	FedAvg(φ)	79.4 ±0.2	86.4 ±0.1	88.3 ±0.1	88.5 ±0.1	77.5 ±0.0	87.0 ±0.2	87.5 ±0.1	86.4 ±0.2
Google Landmarks	✗	FedAvg	82.6 ±0.3	84.7 ±0.1	84.0 ±0.2	84.2 ±0.1	79.7 ±0.0	80.2 ±0.0	80.4 ±0.0	80.4 ±0.0
	✗	FedAvg-1k	75.1 ±0.3	79.2 ±0.1	78.4 ±0.2	77.8 ±0.1	74.5 ±0.1	75.3 ±0.0	75.6 ±0.1	75.6 ±0.1
	Fed3R	✗	73.0 ±0.0	75.7 ±0.0	75.4 ±0.1	75.5 ±0.0	74.1 ±0.1	73.1 ±0.2	74.5 ±0.1	73.5 ±0.1
	Fed3R	FedAvg	83.9 ±0.1	**85.5 ±0.0**	85.4 ±0.0	85.2 ±0.1	80.4 ±0.0	80.3 ±0.1	80.7 ±0.1	80.2 ±0.1
	Fed3R	FedAvg-1k	81.1 ±0.1	81.4 ±0.1	81.5 ±0.1	83.3 ±0.1	78.6 ±0.1	78.1 ±0.0	78.9 ±0.1	78.0 ±0.1
	Fed3R	FedAvg(φ)	84.0 ±0.0	84.0 ±0.0	84.2 ±0.1	85.4 ±0.1	80.4 ±0.0	80.3 ±0.1	80.6 ±0.1	80.1 ±0.1

4.3.5 Experiments with OLL

This section presents the results of the OLL algorithm. The results demonstrate that OLL alone without further local fine-tuning of Phase 4 significantly improves the results by simply removing sources of potential classification errors from classes outside the local distribution. Additionally, fine-tuning the OLL classifiers locally further improves the final WMA, as the new OLL classifier is streamlined for the local class distribution. Finally, the effects of FP with and without OLL are compared to each other, with a closer look at the clients.

Table 4.18 presents the final WMA of OLL applied to the classifier parameters at the end of the training of several Phases 1 and 2 strategies, comparing it with the best results without OLL among 4 possible algorithms for Phase 3. The column with the ✗ symbol for the results with OLL indicates that OLL is directly employed on the federated classifier from Phase 1 or Phase 2 to adapt it to each local clients' data distribution. In other words, the columns associated with the classes in $\mathcal{C} \setminus \mathcal{C}_k$ are removed from the latest classifier, which is then used it to perform predictions.

OLL consistently improves final WMA. OLL improves the final WMA of any FP strategy. Notably, OLL without further fine-tuning in the FP phase already guarantees substantial performance improvements in both the Google Landmarks and iNaturalist scenarios, reaching a final WMA that is always near or above 80% for all the experiments. Remarkably, in the Google Landmarks dataset, OLL(Fed3R + FedAvg(φ)) reaches 84.0% WMA, 3.3 percentage points of WMA higher than the best strategy without OLL, *i.e.,* Fed3R + FedAvg + FedSim(φ) + FP(φ) which achieves only 80.7% WMA. Moreover, OLL(Fed3R + FedAvg-1k) is also sufficient to improve results over the best method without OLL, achieving 81.1% WMA and saving training time, communication, and computational costs. Notably, WMA is even higher than the ones with $FP(f)$ as the strategy for the FP phase. Indeed, OLL essentially imposes a strong bias towards locally present classes, effectively eliminating incorrect predictions for classes not represented locally — a result that the naïve FP strategies also achieve, but with more time and costs.

OLL dramatically simplifies the FP phases, further improving final WMA. OLL significantly improves predictive performance without the need for further local fine-tuning. Indeed, the results of OLL applied to any strategy including Phase 1 and/or Phase 2 and no Phase 4 are consistently higher than the results with Phase 1 and/or Phase 2 and Phase 4. However, in the iNaturalist scenario, strategies including Phase 3 and 4 without using OLL still outperform the strategies with Phase 1, 2 and OLL. For instance, Table 4.11 shows that Fed3R + FedAvg + FP(f) achieves a WMA of 83.3%, compared to 80.5% for OLL(Fed3R + FedAvg). Still, the simplified OLL classifier is designed to enhance subsequent FP optimization. Indeed, it prevents predictions for classes in $\mathcal{C} \setminus \mathcal{C}_k$, thereby eliminating many potential sources of incorrect predictions. Indeed, with OLL, the maximum WMA is 92.5% on iNaturalist, compared to the best result of 88.1% without OLL, representing an improvement of 4.4%. On the Google Landmarks dataset, OLL yields a maximum WMA of 85.5%, whereas the best result without OLL is 80.7%, corresponding to an improvement of 4.8%. Furthermore, the FedAvg-1k algorithm, when used as federated FT method with Fed3R initialization and OLL, achieves a WMA of 85.4% on Google Landmarks and 86.4% on iNaturalist, saving two-thirds of the total FT rounds and thus reducing communication and computational costs.

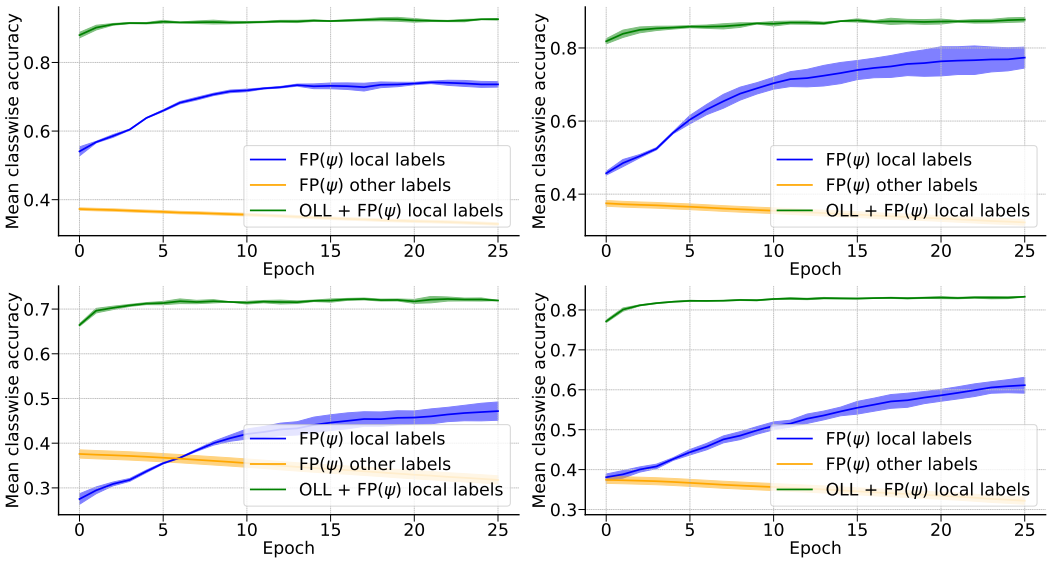


Figure 4.7: Comparison of 1) the average classwise WMA of the local labels for the `FedAvg + FP(`ψ`)` strategy (blue), 2) the average classwise WMA of classes not present in the local client for the `FedAvg + FP(`ψ`)` strategy (orange), and 3) the average classwise WMA of the local labels for the `OLL(FedAvg) + FP(`ψ`)` strategy (green), on four random clients of the iNaturalist dataset. The average classwise WMA of classes not present in the local client for the `OLL(FedAvg) + FP(`ψ`)` strategy is not included in the visualization, as it is consistently zero because of how the `OLL` classifier is constructed. All classwise accuracies are evaluated on the original test set of the iNaturalist dataset (the FL test set presented in Section 3.9.1 and Table 3.1).

FP inherently biases the classifier towards local classes. Figure 4.7 shows that by using the `FP` strategy, the original classifier becomes increasingly biased towards local classes over time (see the orange vs. blue curves). For static client distributions, as the ones considered in this manuscript, this is a positive outcome, as the goal is to maximize WMA only on those local classes. In other words, the deterioration of the performance for classes outside the local distribution is not important. While this is naturally happening with gradient-based `FP`, Figure 4.7 shows that it still requires several epochs to converge. `OLL` dramatically accelerates this process by simplifying the local classification problem, explicitly dropping classes outside the local distribution. By eliminating classes outside the local distribution, `OLL` leads to more efficient training by removing confounding factors for the classifier, such as the interference of potentially wrong-predicting classes outside the local distribution, saving computational costs, and improving final WMA.

4.4 Impact and future works

This chapter presents a thorough analysis of the impact of effective initialization in PFL. It highlights that the optimal strategy depends on several factors, including the available communication and computational budget. At the beginning of the training, it may be beneficial to utilize standard FL training to leverage knowledge extracted from all clients. The analyses of this chapter demonstrate that `Fed3R` can provide a robust initialization for the classifier's parameters, enabling the subsequent use of `PP` or `FP` techniques. This approach accelerates the training process, reduces communication overhead, and minimizes data recency bias and client drift.

The novel `OLL` algorithm has been proposed to enhance personalization further, simplifying the local classifier by focusing exclusively on locally relevant classes. `OLL` effectively minimizes misclassifications to classes outside of the local distribution, resulting in improved efficiency and performance in personalized learning.

A comprehensive empirical evaluation of real-world cross-device datasets demonstrates the significant advantages of `Fed3R` and `OLL`. These algorithms accelerate training, lower communication and computation costs, and enhance performance in FL and PFL scenarios. These contributions pave the way for more efficient and robust FL systems, especially in challenging and heterogeneous settings.

Future work may explore extending these methods to other learning paradigms, such as Continual Learning, where addressing catastrophic forgetting is a critical challenge.

For example, `Fed3R` could be adapted to the Federated Continual Learning (FCL) setting [176], where clients access streams of data, and the same data points available in a given training round may not be accessible in future rounds. By allowing clients to send their statistics to the server multiple times, once for each new stream of data, the server can consistently reconstruct the optimal centralized RR solution without encountering catastrophic forgetting. Furthermore, `OLL` can also be utilized in a personalized FCL scenario. When a client receives a new sample belonging to a novel class c, it can ask the server to share the global parameters of the global classifier associated with that class, specifically the column θ_ψ^c of the global classifier. If this class is part of the global classifier, the client will receive θ_ψ^c and add it as a new column to its local classifier $\theta_{\psi,k}$, which can then be further fine-tuned. Conversely, if the class is not included in the global classifier, the client can stack a

new column initialized with random parameters and subsequently fine-tune it using the new samples.

Finally, investigating the theoretical properties of `Fed3R` and `OLL` in these contexts could further enhance their applicability and impact.

Chapter 5

Federated Learning in computer vision

Part of the content of this chapter, including the methodologies and the results, is adapted, with permission, from the following papers:

- ***E. Fani**, M. Ciccone, B. Caputo. "**FedDrive v2: an Analysis of the Impact of Label Skewness in Federated Semantic Segmentation for Autonomous Driving.**" 5th Italian Conference on Robotics and Intelligent Machines, 2023. (**IRIM23**)* [25].
- L. Fantauzzo*, ***E. Fani****, *D. Caldarola, A. Tavera, F. Cermelli, M. Ciccone, B. Caputo. "**FedDrive: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving.**" IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022. (**IROS22**)* [27].
- D. Shenaj*, ***E. Fani****, *M. Toldo, D. Caldarola, A. Tavera, U. Micheli, M. Ciccone, P. Zanuttigh, B. Caputo. "**Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning.**" IEEE/CVF Winter Conference on Applications of Computer Vision, 2023 (**WACV23**)* [26].

* equal contribution.

The previous chapters explored the intrinsic challenges of FL, mainly focusing on statistical heterogeneity and proposing possible general solutions to client drift. There,

the main source of statistical heterogeneity was class imbalance. The discussion mainly took an algorithmic approach, using image classification as a means to analyze the FL challenges. The claims made in those chapters are intended to be applicable across various tasks.

In contrast, this chapter explores FL within a very practical scenario. Specifically, it studies the challenges of FL in the context of autonomous driving, with semantic segmentation (SS) as the predictive task. SS is the task of assigning a category to each pixel in an image, effectively capturing the complex semantic structure of a scene. SS is critical for various applications, including medical image analysis, satellite image interpretation, and robotics, and plays a vital role in autonomous driving.

The upcoming sections will discuss the importance of SS for autonomous driving and investigate how FL can be applied in this area, addressing its necessity, applicability, and potential challenges. Instead of focusing solely on class imbalance as a source of statistical heterogeneity, this chapter will focus on domain imbalance, where different clients display distinct visual characteristics. The proposed analyses will highlight the unique challenges present in this specific setting, which may not apply to all FL scenarios but are particularly relevant in the context of autonomous driving.

The following sections are structured as follows:

- Section 5.1 introduces the background and the existing works related to the SS task in a FL scenario for autonomous driving.
- Section 5.2 presents `FedDrive`, the first benchmark that studies the efficacy of the adoption of style transfer and domain generalization (DG) techniques in this scenario.
- Finally, Section 5.3 introduces a new realistic problem, Federated Source-Free Domain Adaptation (`FFreeDA`), which takes into account real necessities and constraints of real-world federated methods for SS in the context of self-driving vehicles. Additionally, it introduces Learning Across Domains and Devices (`LADD`), an algorithm to solve `FFreeDA`.

5.1 Background and related works

This section provides the fundamental background and related literature to fully understand the `FedDrive` benchmark and the `LADD` algorithm discussed later in this chapter. Specifically:

- Section 5.1.1 describes the motivations and functioning of the batch normalization (BN) layers.
- Section 5.1.2 introduces the SS task.
- Section 5.1.3 presents the domain adaptation (DA) problem for SS.
- Section 5.1.4 expands Section 5.1.3 by briefly presenting the DA literature for FL.
- Section 5.1.5 describes the DG problem focusing on FL applications.
- Section 5.1.6 presents some existing approaches tackling domain imbalance in FL.
- Finally, Section 5.1.7 shows some style translation techniques from DG and DA that will be used later on in this chapter.

5.1.1 The batch normalization layer

Batch normalization [177] is a technique that enhances the training stability and speed of convergence in deep neural networks. It tackles the problem of internal covariate shift, which happens when the distribution of activations within a network changes during training, complicating the optimization process. By normalizing the activations within a mini-batch, BN layers help maintain a consistent mean and variance, leading to faster convergence and improved model generalization.

In the context of convolutional neural networks (CNNs), an activation is the output generated by a convolutional layer. When processing an input image or feature map, the convolutional layer applies a set of learned filters (kernels), producing new feature maps emphasizing important spatial patterns, such as edges and textures. Formally, the activation at spatial location (i, j) in the $\mathbf{k}$-th feature map is computed as:

$$x_{i,j}^{\mathrm{k}} = \sigma\left(\sum_{c}\sum_{p}\sum_{q} w_{p,q,c}^{\mathrm{k}} \cdot x_{i+p,j+q}^{c} + b^{\mathrm{k}}\right), \tag{5.1}$$

where $w_{p,q,c}^{\mathrm{k}}$ are the filter weights for channel c at the spatial positions p and q, b^{k} is the bias term, and $\sigma(\cdot)$ is a non-linear activation function, *e.g.,* a Rectified Linear Unit (ReLU) [178]. These activations are then propagated to the next layer.

A BN layer initializes a running mean $\mu_{\mathrm{running}}^{\mathrm{k}} = 0$ and running variance $\left(\sigma_{\mathrm{running}}^{\mathrm{k}}\right)^2 = 0$. Then, during training, given a mini-batch of activations $B = \{x_1^{\mathrm{k}}, x_2^{\mathrm{k}}, \ldots, x_m^{\mathrm{k}}\}$ from a convolutional layer, it applies the following transformations:

1. compute the mini-batch mean and variance for each feature map:

$$\mu_B^{\mathrm{k}} = \frac{1}{m}\sum_{i=1}^{m} x_i^{\mathrm{k}}, \quad \left(\sigma_B^{\mathrm{k}}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i^{\mathrm{k}} - \mu_B^{\mathrm{k}})^2. \tag{5.2}$$

2. update the running mean and variance using an hyper-parameter $\alpha \in [0, 1]$ (typically very close to 1, *e.g.,* 0.99):

$$\begin{aligned} \mu_{\mathrm{running}}^{\mathrm{k}} &\leftarrow \alpha\mu_{\mathrm{running}}^{\mathrm{k}} + (1-\alpha)\mu_B^{\mathrm{k}} \\ \left(\sigma_{\mathrm{running}}^{\mathrm{k}}\right)^2 &\leftarrow \left(\alpha\sigma_{\mathrm{running}}^{\mathrm{k}} + (1-\alpha)\sigma_B^{\mathrm{k}}\right)^2 \end{aligned} \tag{5.3}$$

3. normalize the activations:

$$\hat{x}_i^{\mathrm{k}} = \frac{x_i^{\mathrm{k}} - \mu_B^{\mathrm{k}}}{\sqrt{\left(\sigma_B^{\mathrm{k}}\right)^2 + \varepsilon}} \tag{5.4}$$

   where ε is a small constant added for numerical stability.

4. scale and shift using learnable parameters:

$$x_i^{\mathrm{k}+1} = \gamma^{\mathrm{k}}\hat{x}_i^{\mathrm{k}} + \beta^{\mathrm{k}}, \tag{5.5}$$

   where γ^{k} and β^{k} are trainable parameters that allow the network to adapt the normalized activations to an appropriate distribution.

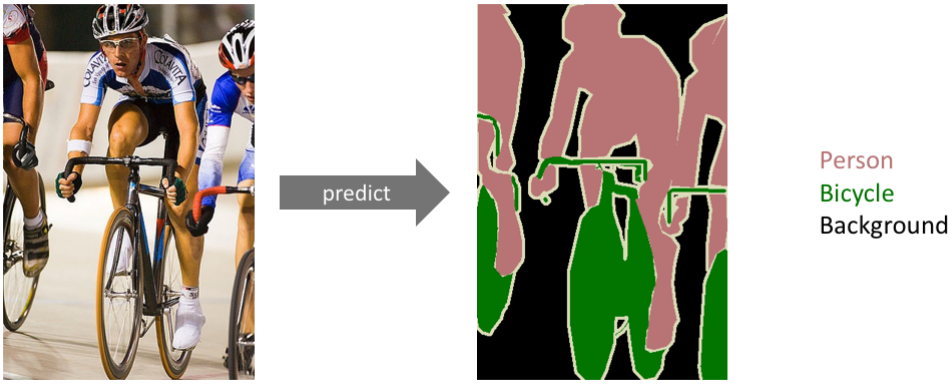


Figure 5.1: Illustration of SS input and corresponding target. The left displays the original image, while the right shows each pixel's corresponding ground truth label. Source: [179].

For convolutional layers, BN is typically applied channel-wise, meaning the normalization is performed separately for each feature map across the spatial dimensions.

At inference time, the BN layer uses the running mean and variance to normalize the activations consistently:

$$\hat{x}_i^{\mathrm{k}} = \frac{x_i^{\mathrm{k}} - \mu_{\mathrm{running}}^{\mathrm{k}}}{\sqrt{\left(\sigma_{\mathrm{running}}^{\mathrm{k}}\right)^2 + \varepsilon}}, \qquad x_i^{\mathrm{k}+1} = \gamma^{\mathrm{k}} \hat{x}_i^{\mathrm{k}} + \beta^{\mathrm{k}}. \tag{5.6}$$

BN reduces sensitivity to weight initialization, enables the use of higher learning rates, and can act as a form of regularization, reducing the need for dropout in some cases. Its widespread adoption in CNN highlights its effectiveness in improving both convergence and generalization.

5.1.2 The semantic segmentation task

The goal of semantic image segmentation (or, shortly, semantic segmentation) is to predict the correct semantic class of every pixel of an image. Therefore, SS can be seen as a generalization of the image classification task described in Section 3.1. This

task is pictorially described in Figure 5.1. Applications of SS range from autonomous driving vehicles to visual image search, handwriting recognition, and medical image diagnostics.

Similarly to the image classification task, the input space $\mathcal{X}$ for a model $f : \mathcal{X} \to \mathcal{Y}$ is assumed to be $\mathcal{X} \subseteq \mathbb{R}^{3\times W\times H}$, and the output space $\mathcal{Y}$ is the natural extension of the image classification output space, this time to predict a label for each single pixel: $\mathcal{Y} \subseteq \left(\Delta^{C-1}\right)^{W\times H}$. Given the similarities with the image classification task, one possibility is to use as the loss function the mean cross-entropy loss computed over the cross-entropy loss defined in Equation 3.1 of every single pixel of the image:

$$\mathcal{L}_{\text{MCE}}(\hat{y}, y) = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\mathcal{L}_{\text{CE}}(\hat{y}_{ij}, y_{ij}). \tag{5.7}$$

One typical metric to evaluate the performance of a SS model is the *mean Intersection over Union* (`mIoU`). Let $\mathcal{C}(c, y)$ be the set of all the pairs (i, j) such that $\arg\max_{c'\in\mathcal{C}} y_{ijc'} = c,\ i \in [W], j \in [H], c \in \mathcal{C}$. In other words, $\mathcal{C}(c, y)$ is the set of pixels of a ground truth label belonging to class c, and $\mathcal{C}(c, \hat{y})$ is the set of pixels predicted by f as class c. For any target y and prediction $\hat{y}$, the Intersection over Union for a given class $c \in \mathcal{C}$ is defined as:

$$\texttt{IoU}(\hat{y}, y; c) = \frac{|\mathcal{C}(c, y) \cap \mathcal{C}(c, \hat{y})|}{|\mathcal{C}(c, y) \cup \mathcal{C}(c, \hat{y})|}, \tag{5.8}$$

and the mean Intersection over Union for any target y and prediction $Haty$ is the mean of the Intersection over Union over all the possible classes $c \in \mathcal{C}$:

$$\texttt{mIoU}(\hat{y}, y) = \frac{1}{C}\sum_{c\in\mathcal{C}}\texttt{IoU}(\hat{y}, y; c). \tag{5.9}$$

finally, for an evaluation dataset $\mathcal{D}$, the mean Intersection over Union is defined as follows:

$$\texttt{mIoU} = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\texttt{mIoU}(\hat{y}, y). \tag{5.10}$$

A family of centralized methods for SS employ an encoder-decoder architecture based on CNN [180–183] or Transformer modules [184–188]. Such encoders are

impractical for real-world applications due to their large number of parameters. Therefore, the viability of more lightweight architectures with smaller model sizes and lower computational complexity has been explored [189, 71, 190].

In FL, the SS task has been primarily studied in the context of medical images [191–194]. Recently, [195] proposed `FedProto`[1], a general method for object segmentation in FL. This approach computes client deviation from prototypical representations learned on distributed data and employs them to drive federated optimization via an attention mechanism.

5.1.3 Domain adaptation for semantic segmentation

SS is a complex, structured prediction task. As such, it generally requires expensive dense annotations. Recently, an increasing number of methods [196, 197] tackle this problem by training on synthetic data generated in virtual environments [198–201] where the labels can be generated automatically. Nonetheless, models trained on these data fail to generalize to the real world because of the inherent domain shift between the simulated and real distributions.

DA [202] aims to reduce the performance gap between a *source* dataset and a *target* dataset that has a different distribution. The source dataset is typically larger than the target dataset. As a result, even though both datasets are available during training, the model tends to rely more on the source data, which limits its ability to generalize effectively to the target domain.

When the source and the target data are unavailable simultaneously, but the model first has access to the source, and then to the target dataset, the problem is referred to as Source-Free Domain Adaptation (SFDA) [203]. If the target data is unlabeled, the problem is known as Unsupervised Domain Adaptation (UDA) [204].

Initially, DA methods attempted to close the gap by measuring domain divergence [205–207]. Another popular direction is adversarial training [208–210], which includes the segmentation network and a domain discriminator competing in a *minimax* game. Other applications attempt to reduce domain shift by employing image-to-image translation algorithms to generate images modified with the style of the other domain [211, 212, 196]. Since this is a time-consuming technique,

[1]Please note that `FedProto` of [195] is not the same algorithm as the `FedProto` proposed by [134] and discussed in Section 3.6.

some non-trainable style translation algorithms, such as Fourier Domain Adaptation (`FDA`) [213], have been introduced. Modern approaches [214–217] use self-learning techniques to create pseudo-labels from the target data.

5.1.4 Domain adaptation in Federated Learning

The study of DA in FL, including both UDA and SFDA, is still in its early stages. Existing works have explored different aspects of this problem: [218] applies unsupervised DA techniques to face recognition, [219] addresses domain shift using adversarial approaches, and [220] models each client as a distinct target domain.

In [221], the authors investigate unsupervised FL under strong theoretical assumptions but focus only on classification tasks with relatively simple datasets such as MNIST [222] and Cifar10 [151].

Instead, [220] focuses on SS, introducing the Federated Multi-Target Domain Adaptation (FMTDA) problem. FMTDA is a more realistic approach where a few clients possess unlabeled target datasets from diverse distributions while a labeled source dataset remains accessible on the server.

5.1.5 Domain generalization in Federated Learning

Domain Generalization [223] focuses on creating a robust model that can perform well across multiple domains. Unlike DA, where access to the target dataset is permitted, DG does not allow interaction with the target dataset. Instead, the training is solely conducted using the source dataset. Some strategies focus on minimizing domain shifts between different source domains [224, 225], while others utilize meta-learning techniques [226–228]. However, both approaches raise concerns about user privacy if naïvely applied to FL, as they require access to all the data involved.

The authors of [230] present a solution to the DG problem in FL. They propose a Continuous Frequency Space Interpolation (`CFSI`) to exchange data distribution information among clients while protecting their privacy. In addition, [231] proposes a novel gradient alignment loss based on the Maximum Mean Discrepancy (`MMD`). Following the footsteps of [230] and [231], `FedDrive` analyzes how image-to-image translation methods such as the frequency exchange in [230] or the `LAB` color space [232] transformation proposed in [233], can be used to address the problem of

Figure 5.2: Example of DG techniques to transfer the style from a source to a target image. `CFSI` may produce artifacts if the amplitude interpolation aligns closely with the target; however, the difference from the original image is less noticeable if it does not. In contrast, `LAB` seems to perform well under all conditions, as it effectively transfers the style of the target image to the source. The original images are taken from the Cityscapes [229] and IDDA [200] datasets.

generalization to unknown domains in FL. Figure 5.2 shows the effects of the `CFSI` and `LAB` DG techniques to two datasets, Cityscapes [229], and Italdesign dataset (IDDA) [200], that will be introduced in the next section.

5.1.6 Domain imbalance in Federated Learning

As presented in Section 2.4.1, domain imbalance is one of the potential sources of statistical heterogeneity. For example, in a scenario where autonomous driving cars collaborate to solve a vision task, domain imbalance may arise from various factors, such as differing weather conditions or variations in architectural styles encountered by the cars, or even the types of cameras used to record scenes.

Most research in FL primarily addresses class imbalance, with only a few notable exceptions that focus on domain imbalance. One such exception is `FedRobust` [234], which tackles the issue of domain imbalance by learning affine transformations.

Another approach involves learning domain-specific BN statistics [235, 236]. A significant drawback of BN layers is the assumption that the training and test data are drawn from the same distribution [237]. This assumption makes BN layers unsuitable for FL scenarios.

To address this issue, [236] propose `FedBN`, which utilizes BN layers to mitigate the feature shift during server-side parameter aggregation. In this approach, local

BN layers are updated on each client but are never shared or aggregated on the server. This results in personalized parameters for each client. However, a key challenge with this method is managing unseen domains or clients during testing.

`SiloBN` [235] clarifies the distinct roles of BN statistics $\{\mu, \sigma\}$ and the learnable parameters $\{\gamma, \beta\}$. The statistics capture local domain information, while the learnable parameters are transferable across different domains. Consequently, clients only share the learnable parameters, keeping the statistics local to address domain heterogeneity. New clients and domains at evaluation time are handled using `AdaBN` [238], which recomputes the BN statistics for the new testing domain while freezing all other parameters of the server model.

5.1.7 Style translation techniques

Fourier Domain Adaptation (`FDA`)

Initially introduced in the context of unsupervised DA, Fourier Domain Adaptation [213] is a technique that allows to capture the style from a source image $x^{\mathrm{S}} \in \mathbb{R}^{3\times W\times H}$ and transfer it to a target image $x^{\mathrm{T}} \in \mathbb{R}^{3\times W\times H}$. The core idea is based on the finding that the amplitudes associated with the low-level distributions encode the visual style of the image, while the high-level semantics can be captured by the phase spectrum in frequency space, as demonstrated by visual psychophysics [239, 240, 213, 230].

First, `FDA` computes the Discrete Fourier Transforms (DFTs) $\mathcal{F}(x^{\mathrm{S}})$ and $\mathcal{F}(x^{\mathrm{T}})$ channelwise:

$$\mathcal{F}(x_c)(m,n) = \sum_{w=0}^{W-1}\sum_{h=0}^{H-1} x_c(w,h)e^{-j2\pi\left(\frac{w}{W}m+\frac{h}{H}n\right)}, \tag{5.11}$$

where $j^2 = -1$ and c indicates a channel of the image x, which could be x^{S} or x^{T}. This can be efficiently computed using the Fast Fourier Transform (FFT) algorithm [241].

Then, for each channel, `FDA` extract the corresponding amplitude $\mathcal{A}(\mathcal{F}(x_c))$:

$$\mathcal{A}(\mathcal{F}(x_c))(m,n) = |\mathcal{F}(x_c)(m,n)| = \\ = \sqrt{\Re\left(\mathcal{F}(x_c)(m,n)\right)^2 + \Im\left(\mathcal{F}(x_x)(m,n)\right)^2}, \tag{5.12}$$

and phase $\Phi(\mathcal{F}(x_c))$:

$$\Phi(\mathcal{F}(x_c))(m,n) = \arg\left(\mathcal{F}(x_c)(m,n)\right) = \tan^{-1}\left(\frac{\Im\left(\mathcal{F}(x_c)(m,n)\right)}{\Re\left(\mathcal{F}(x_c)(m,n)\right)}\right), \tag{5.13}$$

where the symbols $\Re$ and $\Im$ indicate the real and imaginary parts, respectively.

Let $M_\beta : \mathbb{R}^{W\times H} \to \mathbb{R}^{\lfloor \beta W \rfloor \times \lfloor \beta H \rfloor}$ be a masking function that outputs the center crop of a $W \times H$ matrix, with each side scaled by a coefficient $\beta \in (0,1)$:

$$M_\beta(w,h) = \mathbb{1}_{w\in\left[-\frac{\beta}{2}W,\frac{\beta}{2}W\right]\wedge h\in\left[-\frac{\beta}{2}H,\frac{\beta}{2}H\right]}, \tag{5.14}$$

and let $(0,0)$ be the center of the image by definition.

`FDA` computes the Hadamard product between the mask and each channel of the amplitude spectrum. This operation can be seen as applying a low-pass filter to each image channel, returning only the components associated with the low frequency part of the amplitude spectrum. The filtered amplitude matrix of the target image is then substituted with the filtered amplitude matrix of the source image. Finally, the target image is reconstructed through the Inverse DFT (IDFT). The `FDA` process can be formalized as:

$$\begin{aligned}\texttt{FDA}\left(x_c^{\mathrm{S}}, x_c^{\mathrm{T}}; \lambda\right) = \mathcal{F}^{-1}\Big(&\lambda M_\beta \odot \mathcal{A}\left(\mathcal{F}\left(x_c^{\mathrm{S}}\right)\right) + \\ &(1-\lambda) M_\beta \odot \mathcal{A}\left(\mathcal{F}\left(x_c^{\mathrm{T}}\right)\right) + \\ &(1 - M_\beta) \odot \mathcal{A}\left(\mathcal{F}\left(x_c^{\mathrm{T}}\right)\right), \Phi\left(\mathcal{F}\left(x_c^{\mathrm{T}}\right)\right)\Big),\end{aligned}$$

$$\texttt{FDA}\left(x^{\mathrm{S}}, x^{\mathrm{T}}\right) = \begin{bmatrix} \texttt{FDA}\left(x_1^{\mathrm{S}}, x_1^{\mathrm{T}}\right) \\ \texttt{FDA}\left(x_2^{\mathrm{S}}, x_2^{\mathrm{T}}\right) \\ \texttt{FDA}\left(x_3^{\mathrm{S}}, x_3^{\mathrm{T}}\right) \end{bmatrix}, \tag{5.15}$$

where $\lambda \in (0, 1]$ is a hyper-parameter that in [213] is set to 1.

Continuous Frequency Space Interpolation (CFSI)

Inspired by [213], [230] proposes Continuous Frequency Space Interpolation (CFSI), a style translation technique designed explicitly for FL to address domain imbalance.

Given an image from client $k \in \mathcal{K}$, the frequency space signal can be derived using Equations (5.12) and (5.13). The low-level information (*i.e.,* color, brightness, etc.) is reflected in the amplitude spectrum, while the semantics is reflected in the phase spectrum. To share the distribution information among local clients, a distribution bank $\mathtt{A} = \bigcup_{k \in \mathcal{K}} \mathtt{A}^k$ is firstly created, where each $\mathtt{A}^k = \bigcup_{(x,y) \in \mathcal{D}^k} \{\mathcal{A}(\mathcal{F}(x))\}$ contains all amplitude spectrum of images of client k[2]. Then, during training, CFSI applies FDA to all the local images, by randomly picking an image amplitude from $\mathtt{A} \setminus \mathtt{A}^k$ as the source amplitude for every local image. Therefore, FDA is used by CFSI as a data augmentation technique to expose the model to different client distributions during local training while preserving the clients' privacy.

LAB-based image translation

The authors of [233] propose a LAB-based image translation technique for style translation. The core idea is similar to CFSI, but instead of transferring the style in the frequency spaces, the image style is modified in the LAB color space [232].

To share the distribution information across local clients, a shared bank $\mathtt{L} = \bigcup_{k \in \mathcal{K}} \mathtt{L}^k$ is created, where each $\mathtt{L}^k = \bigcup_{(x,y) \in \mathcal{D}_k} \{(\mu_x, \sigma_x)\}$ is the set of means and standard deviations for each image of client k, transformed in the LAB color space. More specifically, an RGB image $x^{\mathrm{S}}_{\mathrm{RGB}}$ of a client k is firstly converted to the LAB color space $x^{\mathrm{S}}_{\mathrm{LAB}}$. Then, the mean μ^{S} and the standard deviation σ^{S} are computed for each $x^{\mathrm{S}}_{\mathrm{LAB}}$ channel. A random pair of $\left(\mu^{\mathrm{T}}, \sigma^{\mathrm{T}}\right)$ is sampled from the distribution bank L. The $x^{\mathrm{S}}_{\mathrm{LAB}}$ image style is then translated as follows:

$$\hat{x}^{\mathrm{S}}_{\mathrm{LAB}} = \frac{\sigma^{\mathrm{T}}}{\sigma^{\mathrm{S}}} \left(x^{\mathrm{S}}_{\mathrm{LAB}} - \mu^{\mathrm{S}}\right) + \mu^{\mathrm{T}}. \tag{5.16}$$

[2] $\mathcal{A}(\mathcal{F}(x))$ indicates the stack of the amplitude spectrum of x_c for all the three RGB channels.

Finally, the translated LAB image $\hat{x}^{S}_{LAB}$ is converted back to the RGB color space as $\hat{x}^{S}_{RGB}$, for the subsequent training phase.

5.2 FedDrive, the first Federated Learning benchmark for semantic segmentation applied to autonomous driving

Research in autonomous driving aims to improve our safety and driving experience. To derive and execute trustworthy actions autonomously, vehicles must be able to perceive and understand their surroundings [242–246]. An autonomous vehicle needs to precisely determine whether there is a danger, such as an obstruction or a pedestrian, and consequently decide whether to slow down or accelerate to ensure the safety of the passengers. To this aim, vehicles utilize the SS task to provide a semantic prediction for each pixel in an image, thereby enhancing understanding of the surrounding environment.

However, training robust SS models requires having access to large-scale datasets that possibly represent all the conditions that can be encountered in the real world, which could be a challenging problem as data generation and collection are costly processes (see Section 2.1). Indeed, one practical solution is to collect images from customers' vehicles, which are already on the road and encountering various situations across multiple geographic locations, weather conditions, and viewpoints. Collecting images from vehicles can lessen the workload for a company; however, sending these images to a central server for model training may compromise user privacy, as personal and privacy-protected information could potentially be directly or indirectly violated, such as revealing the Global Positioning System (GPS) coordinates or the habits or personal routines of the users. Moreover, the privacy risks associated with data collection can also lead to violations of existing regulations [56, 247, 248].

Therefore, FL presents a promising solution for training models using data from multiple clients while ensuring their privacy is protected. However, FL is still a relatively new field, and current methods primarily focus on straightforward vision tasks, such as image classification [13, 74, 235, 236]. Hence, using FL for SS in realistic urban environments could introduce challenges that had never been explored before related to the diverse scenes encountered.

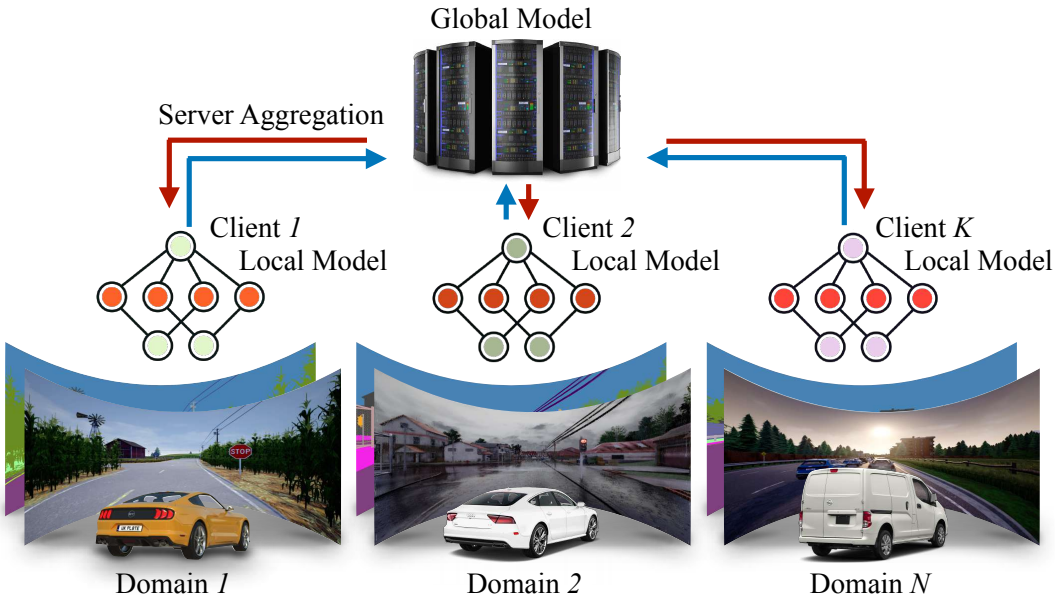


Figure 5.3: Multiple K vehicles (*clients*), driving in different N domains, interact with a central server to learn a global model while keeping data private.

Statistical heterogeneity is a key challenge of FL, as already thoroughly discussed in Chapters 3 and 4. Indeed, FL algorithms usually suffer in terms of convergence speed, increasing training time and hampering the final model performance. In the scenario of FL for the SS task for self-driving vehicles, there are several potential sources of heterogeneity.

For instance, images taken outdoors by private autonomous vehicles can vary significantly due to differences in lighting, reflections, viewpoints, atmospheric conditions, geographical location, viewpoint, weather, and illumination, leading to various *visual domains* which contribute to the non-i.i.d-ness on the clients' data distribution in the form of domain imbalance and shown in Figure 5.3. Given the critical nature of SS in autonomous driving, models need to be robust enough to ensure the safety of passengers, therefore avoiding making incorrect decisions based on inaccurate environmental semantic predictions.

Furthermore, the model should be resilient to unexpected and unpredictable situations. As a result, DG becomes a crucial aspect, requiring the model to perform effectively in environments that differ from its training conditions. New clients may emerge anytime in real-world scenarios, exposing the model to unknown distributions. For instance, in an autonomous driving context, new vehicles may operate in locations

the model has never seen at training time, such as different cities or countries, or introduce entirely new vehicle types with different characteristics. A robust model must be able to generalize and adapt to these novel, unseen conditions to function effectively in the real world. This capability is crucial for real-world applications since not all potential scenarios can be anticipated during training. Therefore, a model must ensure safe operation under those unforeseen conditions.

Another critical source of statistical heterogeneity is class imbalance across the clients. Indeed, certain clients may access environments with different categories, leading to an uneven observation of these categories. For example, a fleet of autonomous vehicles deployed in various locations may encounter a limited or unbalanced set of classes. They might record differing numbers of riders, vehicles, traffic signs, or pedestrians and may not come across some classes at all.

To analyze domain and class imbalance in the context of FL for the SS task for autonomous driving, this manuscript introduces `FedDrive`, the first SS benchmark in a FL scenario applied to autonomous driving. `FedDrive` benchmarks FL algorithms, eventually combined with style-transfer methods [230, 233], to assess if current methods can deal with these real-world challenges and generalize well on unseen domains.

The main contributions of `FedDrive` are the following:

- `FedDrive` is the first benchmark for FL in SS applied to autonomous driving, focusing on the real-world challenges of statistical heterogeneity and DG.
- `FedDrive` compares federate learning methods and combines them with style transfer techniques to improve their generalization.
- `FedDrive` underscores the importance of using the correct clients' statistics for the batch normalization layers when dealing with different domains.
- `FedDrive` evaluate how style transfer techniques can improve the performance on unseen domains.
- `FedDrive` introduces 12 novel FL splits with different levels of statistical heterogeneity.

Table 5.1: Summary of the FedDrive federated datasets.

Dataset	Setting	Distribution	K	n_k	Test sets
Cityscapes [229]	-	Uniform, Heterogeneous, Class Imbalance	146	10 - 45	unseen cities
IDDA [200]	Country	Uniform, Heterogeneous, Class Imbalance	90	48	seen + unseen (country) domains
	Rainy	Uniform, Heterogeneous, Class Imbalance	69	48	seen + unseen (rainy) domains
	Bus	Uniform, Heterogeneous, Class Imbalance	83	48	seen + unseen (bus) domains

5.2.1 Proposed semantic segmentation datasets for autonomous driving in FL

The FedDrive federated splits are constructed from two centralized datasets for SS in the autonomous driving scenario: Cityscapes [229] and IDDA [200]. Cityscapes comprises 2975 annotated images for training and 500 for testing, all gathered from similar cities in Central Europe in optimal weather conditions. Instead, IDDA is a synthetic dataset with 105 different *domains* of 60 images each, uniquely characterized by the triad (*weather*, *viewpoint*, *town*), where *weather*, *viewpoint* and *town* are one among three weather conditions, five different points of view for the camera recording the scenes simulating various vehicles, and seven distinct locations, which are composed of six cities and one bucolic country, respectively.

Table 5.1 summarizes the proposed FL splits of the FedDrive benchmark. The FL datasets proposed by FedDrive are identified by a *setting* and a *distribution*.

The *setting* indicates how the test sets are composed. There is only one possible test set for Cityscapes, which is the evaluation set from the original Cityscapes dataset, composed of 500 images from cities not present in the training set. In contrast, the IDDA has three distinct settings: *country*, *rainy*, and *bus*. In the *country* setting, the *unseen* test set consists of the union of all domains located in rural areas. For the *rainy* setting, the *unseen* test set includes the union of all domains characterized by rainy weather. Lastly, in the *bus* setting, the *unseen* test set comprises the union of all domains where the images are captured from a bus viewpoint. For IDDA, an additional *seen* test set is provided, composed of one-fifth of the images of each of the remaining domains. The remaining four-fifths of the images per each domain

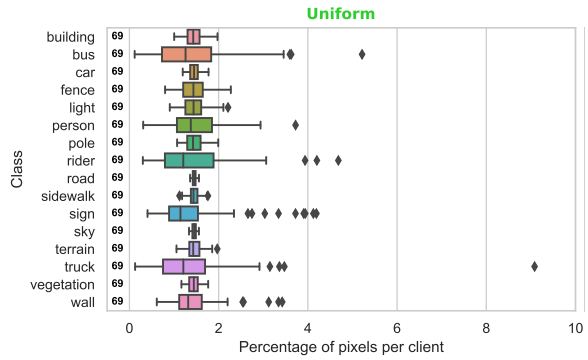
Algorithm 8 - Class Imbalance client distribution generation

Require:
$\mathcal{D}$ = set of all the available training images
Empty client datasets $\mathcal{D}_k \; \forall k \in \mathcal{K}$
List $S \in \mathbb{N}^K : \|S\|_1 = n$, S_k = desired number of samples (x, y) for client k
$\mathcal{D}^c \subseteq \mathcal{D} \; \forall c \in \mathcal{C}$: set of all $(x, y) \in \mathcal{D}$ such that class c appears in y
for each $k \in \mathcal{K}$ **do**
  $c^* = \arg\min_{c \in \mathcal{C}} |\mathcal{D}^c|$
  while $|\mathcal{D}_k| < S_k$ **do**
    $\mathcal{E}$ = set of $min(S_k - n_k, |\mathcal{D}^{c^*}|)$ randomly sampled pairs (x, y) from $\mathcal{D}^{c^*}$
    Assign samples $(x, y) \in \mathcal{E}$ to $\mathcal{D}_k$: $\mathcal{D}_k = \mathcal{D}_k \cup \mathcal{E}$
    Update $\mathcal{D}^c \; \forall c \in \mathcal{C}$: $\mathcal{D}^c = \mathcal{D}^c \setminus \mathcal{E} \; \forall c \in \mathcal{C}$
  end while
end for
Return $\mathcal{D}_k \; \forall k \in \mathcal{K}$

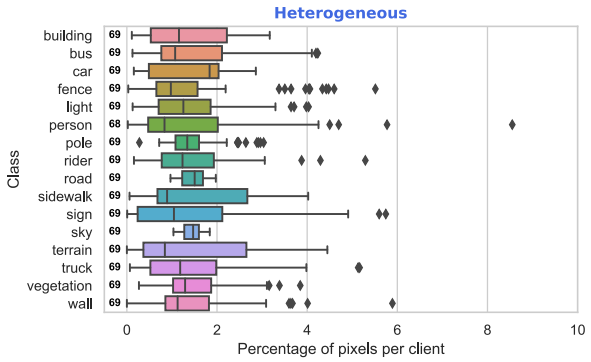
left constitute the training set. The distinction among three different settings and the evaluation of the methods on *seen* and *unseen* test sets allow for a fair evaluation of various methods for DG, considering different types of domain shifts between the training and test sets.

The *distribution* indicates how the training images are distributed across the different clients for each setting. For both the Cityscapes and IDDA datasets, there are three possible settings: *uniform*, *heterogeneous*, and *class imbalance*. In the *uniform* distribution for the Cityscapes dataset, each client has access to the same number of images from all the cities. Similarly, in IDDA each client has access to the same number of images from all the domains. Instead, in the *heterogeneous* setting, each client has access to the images of only one city for the Cityscapes datasets and only one domain for the IDDA dataset. The purpose of this setting is to maximize the domain imbalance across the clients, as various cities/domains may exhibit different building styles, weather conditions, or viewpoints of the camera capturing the images.

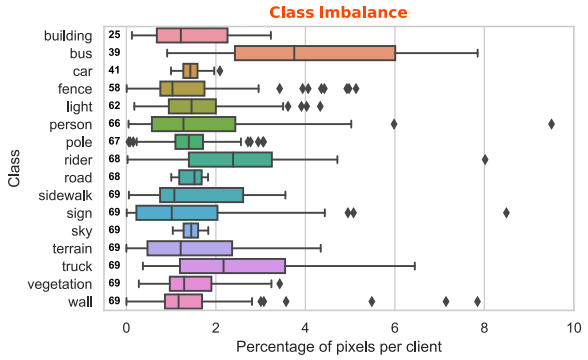
Finally, the *class imbalance* distribution is meant to maximize the class imbalance across the clients. Given any training dataset $\mathcal{D}$ and the list of desired number of images per client S, Algorithm 8 details the process adopted in FedDrive to allocate the training images to enhance class imbalance. The class imbalance distribution represents another realistic source of statistical heterogeneity, since some autonomous driving cars may be primarily exposed to environments where specific categories are more frequent than others, while some clients may not have access to some categories



(a)



(b)



(c)

Figure 5.4: Distribution of the percentage of pixels belonging to each class, per client, on the *rainy* IDDA setting for (a) the *uniform*, (b) the *heterogeneous*, and (c) *class imbalance* clients' distribution. The numbers between the class labels and the box plots are the number of clients having at least one image with that class, where 69 is the total number of clients. The *class imbalance* distribution has wider boxes, indicating more variation in the quantity of pixels of each class among the clients.

at all. The different class distributions for the *uniform*, *heterogeneous*, and *class imbalance* scenarios for the *rainy* setting are shown in Figure 5.4.

5.2.2 Inference strategies for `SiloBN`

One of the selected baselines for domain imbalance is `SiloBN` [235]. At inference time, `SiloBN` has been evaluated with two different strategies:

- *standard* strategy. This strategy directly computes the batch normalization statistics (μ, σ) from the test set.
- *by domain* strategy. Similarly to a personalized FL scenario, assuming that each training client may have its local test set is reasonable. In this case, the model is evaluated over all the local test sets by using the corresponding local statistics (μ, σ) computed at training time. Differently from the personalized FL scenario of Chapter 4, the evaluation metric is the accuracy (instead of the weighted mean accuracy) on the union of all the test sets.

5.2.3 Implementation details

Because in real-world cross-device FL scenarios, devices are assumed to have limited computational capabilities, simulations have been executed using a BiSeNet V2 architecture [249]. This lightweight network has two principal branches that capture spatial features and high-level semantic context. In all the experiments except for the ones in Section 5.2.6, the server optimizer is stochastic gradient descent with learning rate 1.0.

On each client, the cross-entropy loss is minimized using SGD as the local optimizer, with an initial learning rate of 0.05 for Cityscapes and 0.1 for IDDA, momentum of 0.9, and weight decay of 0.0005 for Cityscapes but no weight decay for IDDA. The batch size is set to 16, and when clients do not have sufficient samples, their dataset is virtually doubled by applying horizontal flipping.

A polynomial learning rate schedule is implemented for each client, similarly to [249], and online Hard-Negative Mining [250] is employed to improve further optimization, where only the hardest 25% of the pixels are used to compute gradients and update the model parameters. Additionally, a standard data augmentation pipeline is utilized. This includes randomly scaling images within 0.5 to 1.5 for the Cityscapes dataset and 0.5 to 2.0 for the IDDA dataset. After scaling, the images are cropped to 512×1024 for Cityscapes and 512×928 for IDDA.

Table 5.2: FedDrive results for the Cityscapes dataset.

Distribution	Method	CFSI	LAB	mIoU (%)
Uniform	FedAvg	✗	✗	45.6 ± 1.3
Heterogeneous	FedAvg	✗	✗	**43.3** ± **1.6**
		✓	✗	40.6 ± 2.2
		✗	✓	42.7 ± 2.1
	SiloBN	✗	✗	**52.9** ± **1.3**
		✓	✗	52.1 ± 1.8
		✗	✓	53.4 ± 1.7
Class Imbalance	FedAvg	✗	✗	44.5 ± 1.7
		✓	✗	**48.3** ± **1.8**
		✗	✓	44.3 ± 1.7
	SiloBN	✗	✗	**51.8** ± **1.2**
		✓	✗	50.8 ± 1.1
		✗	✓	51.5 ± 1.2

In these experiments, 5 clients are sampled in each round, with each client training its model for 2 local epochs. The total number of rounds for all experiments is set to 1600. Results are presented using the mean and standard deviation of the last mIoU scores from 20 evaluations on the test set, which are computed every 5 rounds over the final 100 rounds.

In the CFSI experiments, unlike the approach taken in [230], which interpolates each client image across all domain distributions, the FedDrive experiments populate the amplitude bank A with only half of the clients' images. This adjustment is necessary because the style transfer methods require significant computational resources, making the original strategy unscalable and impractical in a realistic scenario with multiple domains.

5.2.4 Cityscapes results

Table 5.2 shows the mIoU of the Cityscapes experiments. The first line shows the performance of FedAvg with the *uniform* distribution (45.6 ± 1.3%) as a reference for the performance in the absence of domain shift.

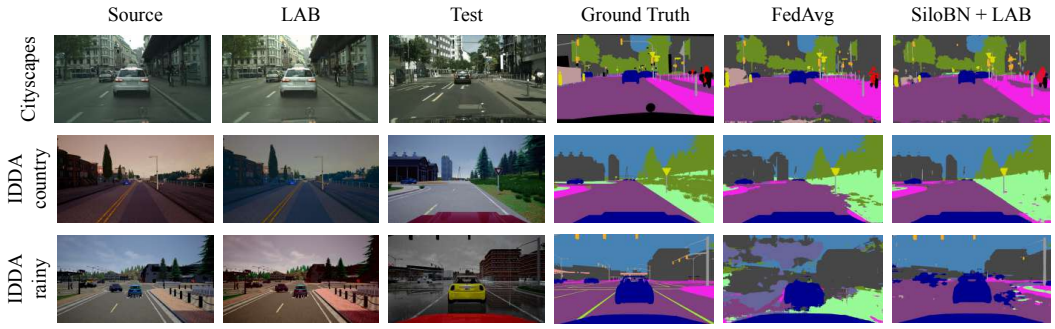


Figure 5.5: Qualitative results in the *heterogeneous* distribution for Cityscapes and for the *country* and *rainy* IDDA settings with the *heterogeneous* distribution.

In the Cityscapes dataset, the improvements achieved with `SiloBN` are consistent, showing an increase of about 9 to 10 percentage points in mIoU compared to the other experiments. This is noteworthy, given that the domain shift across cities is primarily related to semantic differences rather than stylistic ones. Indeed, the test images are taken from cities not seen during training but in the same country and under similar weather conditions.

On the contrary, the style translation methods do not contribute to performance improvement: in the *heterogeneous* distribution, there are performance drops of about 3 and 1 percentage points of mIoU between the `FedAvg` experiment and the `FedAvg + CFSI` ($40.6 \pm 2.2\%$ vs. $43.3 \pm 1.6\%$) and `FedAvg + LAB` experiments ($42.7 \pm 2.1\%$ vs. $43.3 \pm 1.6\%$), respectively, while the performances are similar between no style translation, `CFSI`, and `LAB`, when applying `SiloBN`.

Regarding the *class imbalance* experiments, `SiloBN` still improves performance; however, the gains are less significant compared to the ones of the *heterogeneous* distribution. This is because `SiloBN` primarily focuses on overcoming domain shifts rather than addressing label skewness. Similarly, `CFSI` and `LAB` do not appear to provide considerable benefits when used with `SiloBN`. Nevertheless, with the given client distribution, `CFSI` does improve performance by four percentage points compared to using `FedAvg` alone in experiments that do not include `SiloBN`.

5.2.5 IDDA results

This section presents the IDDA results for the `FedDrive` benchmark. First, note that *country*, *rainy* and *bus* settings present different challenges, as illustrated in

Table 5.3: FedAvg results on IDDA. Values are expressed in mIoU (%).

Distribution	Setting	CFSI	LAB	Test set	
				unseen	seen
Uniform	country	✗	✗	49.7 ± 0.8	63.6 ± 0.6
	rainy	✗	✗	27.6 ± 2.8	62.7 ± 3.7
	bus	✗	✗	58.5 ± 1.3	64.9 ± 0.7
Heterogeneous	country	✗	✗	40.0 ± 1.3	42.4 ± 1.8
		✓	✗	**45.7 ± 1.7**	54.7 ± 1.1
		✗	✓	**45.7 ± 1.0**	**56.6 ± 0.9**
	rainy	✗	✗	26.8 ± 2.3	38.2 ± 1.4
		✓	✗	**31.1 ± 2.7**	55.2 ± 1.7
		✗	✓	26.8 ± 1.8	**58.9 ± 0.9**
	bus	✗	✗	38.1 ± 2.0	45.7 ± 1.7
		✓	✗	48.9 ± 1.5	56.9 ± 1.4
		✗	✓	**50.5 ± 1.1**	**58.8 ± 1.0**
Class Imbalance	country	✗	✗	47.6 ± 0.7	58.1 ± 0.8
		✓	✗	48.7 ± 0.8	59.7 ± 0.9
		✗	✓	**48.9 ± 0.8**	**60.1 ± 1.2**
	rainy	✗	✗	**29.5 ± 1.9**	58.8 ± 1.5
		✓	✗	25.7 ± 2.8	60.4 ± 0.6
		✗	✓	29.1 ± 1.7	**61.2 ± 1.0**
	bus	✗	✗	53.3 ± 2.1	60.5 ± 1.8
		✓	✗	52.9 ± 1.3	61.2 ± 1.2
		✗	✓	**54.0 ± 1.2**	**62.1 ± 0.6**

Figure 5.5 for *country* and *rainy*. Indeed, the *country* domain shares a style similar to that of the training clients since the images are captured under varying weather conditions. However, it possesses distinct semantic characteristics; for instance, it rarely has sidewalks adjacent to the road, which is commonly found in training clients. In contrast, the *rainy* and *bus* domains have similar semantic features but a fundamentally different appearance. Indeed, no images capture *rainy* conditions in the training set for the former, and the position of the camera capturing the scenes is in a different point of view in the test images of the training images for the latter.

Table 5.4: `SiloBN` results on the *heterogeneous* distribution of IDDA. Values are expressed in mIoU (%). "standard" and "by domain" refer to the evaluation strategies introduced in Section 5.2.2.

Setting	CFSI	LAB	Test set		
			unseen	seen	
			standard	standard	by domain
Country	✗	✗	45.3 ± 0.9	54.5 ± 0.7	58.8 ± 2.9
	✓	✗	49.2 ± 1.0	63.4 ± 0.6	61.2 ± 3.4
	✗	✓	**50.2 ± 0.6**	**64.6 ± 0.5**	64.3 ± 0.8
Rainy	✗	✗	50.0 ± 0.8	54.4 ± 0.8	62.5 ± 1.4
	✓	✗	50.5 ± 0.9	64.9 ± 0.2	63.0 ± 0.3
	✗	✓	**54.0 ± 0.8**	**65.9 ± 0.6**	**65.9 ± 0.9**
Bus	✗	✗	47.4 ± 0.8	57.8 ± 0.9	61.6 ± 1.4
	✓	✗	55.8 ± 1.0	65.8 ± 0.8	64.0 ± 2.7
	✗	✓	**56.2 ± 0.6**	**67.0 ± 0.3**	66.2 ± 0.8

Table 5.5: `SiloBN` results on the *class imbalance* distribution of IDDA. Values are expressed in mIoU (%).

Setting	CFSI	LAB	Test set	
			unseen	seen
Country	✗	✗	51.2 ± 0.6	66.5 ± 0.4
	✓	✗	51.7 ± 0.7	67.2 ± 0.4
	✗	✓	**53.1 ± 0.6**	**67.3 ± 0.4**
Rainy	✗	✗	**54.7 ± 0.6**	66.4 ± 0.5
	✓	✗	54.4 ± 1.0	67.1 ± 0.4
	✗	✓	52.9 ± 0.8	**67.6 ± 0.3**
Bus	✗	✗	57.6 ± 0.6	67.5 ± 0.4
	✓	✗	57.7 ± 0.7	**67.9 ± 0.6**
	✗	✓	**58.0 ± 0.6**	67.8 ± 0.4

Table 5.3 shows the experiments on the IDDA dataset using `FedAvg`, while Table 5.4 and Table 5.5 show the results for the `SiloBN` algorithm on the *heterogeneous* and *class imbalance* settings, respectively.

Similar to Table 5.2 for Cityscapes, the first three rows of Table 5.3 show the mIoU of the `FedAvg` method under a uniform distribution across all three settings,

which serves as a reference. The results demonstrate strong performance when the same domain is used for the test client (*seen*). However, when evaluating FedAvg on the *unseen* test client, performance declines significantly, particularly in the *rainy* setting, where the mIoU drops by an average of 35% compared to the *seen* test client. This highlights the substantial domain shift caused by the *unseen* test client.

Introducing domain imbalance in *heterogeneous* experiments significantly affects the performance of FedAvg. On average, the performance of seen test clients decreases by nearly 23% mIoU, reaching 42.4% in the *country* setting, 38.2% in the *rainy* setting, and 45.7% in the *bus* setting. These results highlight the domain imbalance among training clients and the high statistical heterogeneity of this scenario. Similarly, the performance drop for unseen test clients is considerable, with a decrease of about 10% mIoU in the *country* setting and approximately 20% mIoU in the *bus* setting. The only exception is the *rainy* setting, where the mIoU drop is less than 1%. This is due to the particularly challenging domain shift between the training and test sets in this scenario, which has already severely impacted performance.

The *class imbalance* FedAvg experiments perform significantly better than the corresponding *heterogeneous* experiments in both the test sets, indicating that the maximum class imbalance achievable in this scenario only marginally increases statistical heterogeneity. This shows that domain imbalance is a more relevant source of heterogeneity for the SS task in FL.

As demonstrated in Tables 5.3 to 5.5, applying the style transfer methods CFSI and LAB significantly improves the mIoU. Notably, on the seen test client, LAB provides a substantial performance boost when utilized with both FedAvg and SiloBN. In the *country* setting, LAB enhances FedAvg by approximately 22% and SiloBN by around 6%. In the *rainy* setting, it achieves an impressive improvement of about 27% on FedAvg and 4% on SiloBN. It is important to highlight that while LAB yields significant performance improvements for FedAvg, its effectiveness on SiloBN is comparatively lower; this is because it transfers the style of the training domains across clients' images, much like SiloBN does.

Table 5.4 shows the results for the seen test set comparing both the *standard* and the *by domain* inference strategy for the *heterogeneous* experiments. The *by domain* strategy provides better performance without style translation techniques, guaranteeing about 4% mIoU increase compared to the *standard* strategy. However, when combined with domain translation techniques, the *standard* strategy provides

Table 5.6: Server optimizers comparison on Cityscapes.

Distribution	SGD	FedAvgM	Adam	AdaGrad
Uniform	45.6 ± 1.3	40.0 ± 4.3	**45.9 ± 1.3**	44.3 ± 3.7
Heterogeneous	43.3 ± 1.7	37.8 ± 4.6	**45.1 ± 1.6**	42.3 ± 3.1
Class Imbalance	44.5 ± 1.7	36.1 ± 4.5	**45.2 ± 1.8**	39.8 ± 4.1

Table 5.7: Server optimizers comparison on IDDA without SiloBN.

Distribution	Setting	Test set							
		unseen				seen			
		SGD	FedAvgM	Adam	AdaGrad	SGD	FedAvgM	Adam	AdaGrad
Uniform	Country	49.7 ± 0.8	**55.5 ± 1.1**	50.2 ± 0.4	46.1 ± 0.8	63.6 ± 0.6	**71.3 ± 0.9**	63.3 ± 0.4	59.4 ± 1.0
	Rainy	27.6 ± 2.8	29.8 ± 2.0	**31.7 ± 1.4**	27.8 ± 1.3	62.7 ± 3.7	**71.0 ± 0.7**	65.4 ± 0.5	59.5 ± 1.3
	Bus	58.5 ± 1.3	**62.3 ± 1.3**	58.6 ± 0.4	54.0 ± 1.2	64.9 ± 0.7	**71.8 ± 0.4**	64.8 ± 0.4	61.2 ± 1.0
Heterogeneous	Country	40.0 ± 1.3	**42.4 ± 2.2**	38.2 ± 1.9	38.0 ± 1.7	42.4 ± 1.9	**44.4 ± 2.0**	39.9 ± 2.4	40.9 ± 2.1
	Rainy	26.8 ± 2.3	**31.9 ± 3.8**	28.5 ± 2.6	27.1 ± 2.9	38.2 ± 1.4	**41.2 ± 2.0**	38.0 ± 2.0	39.0 ± 1.7
	Bus	38.1 ± 2.0	**40.4 ± 1.6**	38.9 ± 1.4	37.2 ± 1.6	45.7 ± 1.7	**48.9 ± 1.5**	45.2 ± 1.1	44.5 ± 1.3
Class Imbalance	Country	47.6 ± 0.7	**50.2 ± 1.2**	47.4 ± 0.6	44.6 ± 1.3	48.1 ± 0.8	**63.0 ± 1.2**	59.0 ± 0.7	55.5 ± 1.0
	Rainy	29.5 ± 1.9	32.3 ± 1.9	**35.8 ± 1.9**	29.3 ± 1.2	58.8 ± 1.5	**64.0 ± 1.1**	57.5 ± 1.2	55.9 ± 1.9
	Bus	53.3 ± 2.1	**54.7 ± 1.7**	53.1 ± 0.7	49.0 ± 1.5	60.5 ± 1.8	**64.5 ± 1.0**	61.0 ± 0.7	57.6 ± 1.4

slightly better results. Therefore, the *by domain* inference strategy is a solid choice when style translation is not available for computational constraints.

5.2.6 Ablation on server optimizers

Table 5.6 show the results using different server optimizers on Cityscapes. Similarly, Tables 5.7 and 5.8 show the results using different server optimizers without and with SiloBN, respectively, on the various distributions and settings of IDDA. On Cityscapes, Adam consistently performs better than the other server optimizers, and SGD is always the second best. On the contrary, on IDDA, FedAvgM consistently provides the best results across all the distributions and settings. Moreover, Table 5.8 shows that the *by domain* inference strategy for SiloBN is particularly effective when combined with FedAvgM, especially in the *country* setting, improving the final mIoU by 12%.

Table 5.8: Server optimizers comparison using SiloBN on the *heterogeneous* distribution.

Setting	Server Optimizer	Test set		
		unseen	seen	
		standard	standard	by domain
Country	SGD	45.3 ± 0.9	54.5 ± 0.7	58.8 ± 2.9
	FedAvgM	**46.2** ± **1.2**	54.6 ± 1.3	**62.0** ± **1.5**
	Adam	42.3 ± 0.8	52.5 ± 0.8	58.4 ± 1.3
	AdamGrad	41.7 ± 1.3	51.4 ± 1.0	49.0 ± 1.3
Rainy	SGD	**50.0** ± **0.8**	50.4 ± 0.8	62.5 ± 1.4
	FedAvgM	48.5 ± 1.0	51.7 ± 0.7	**63.7** ± **1.3**
	Adam	47.2 ± 0.9	50.9 ± 0.5	61.5 ± 0.9
	AdamGrad	45.8 ± 0.9	48.5 ± 0.8	54.1 ± 0.3
Bus	SGD	**47.4** ± **0.8**	57.8 ± 0.9	61.6 ± 1.4
	FedAvgM	46.9 ± 1.0	56.0 ± 0.8	**63.4** ± **1.4**

5.2.7 Impact and future works

This manuscript introduces FedDrive, a benchmark for SS in FL designed for autonomous driving. FedDrive offers a comprehensive set of real-world scenarios to analyze the effects of heterogeneous domain distributions across clients, mainly focusing on domain and class imbalance.

Evaluations of methods from the FL and DG literature demonstrate that SiloBN consistently enhances performance, especially in the presence of severe domain imbalance. In contrast, DG techniques such as LAB and CFSI exhibit inconsistent effectiveness; they sometimes fail to improve performance or even degrade it, as seen in the Cityscapes experiments. Furthermore, although initially designed for DA and DG, SiloBN, LAB, and CFSI are also effective in scenarios with class imbalance. Findings suggest domain shifts present more significant challenges than label skewness in federated SS for autonomous vehicles.

FedDrive establishes a foundation for advancing research in privacy-preserving autonomous driving. Future work may focus on developing specialized algorithms to address class imbalance and label skewness in federated settings and expand FedDrive with additional large-scale, real-world datasets. Moreover, another

possible future direction is to analyze real-world scenarios where autonomous driving cars do not have access to ground-truth labels, therefore requiring the adoption of unsupervised or semi-supervised learning techniques; this is the exact direction explored in the next section of this chapter.

5.3 Style-driven source-free domain adaptation in Clustered Federated Learning

Most FL research assumes that clients possess labeled data. However, this assumption is often unrealistic, particularly in SS, where dense pixel-level annotations require extensive manual effort and high costs [220]. This section introduces Federated Source-Free Domain Adaptation (`FFreeDA`), a novel, real-world, cross-device FL system where clients' data is entirely unlabeled, and the server has access to a labeled source dataset only for pre-training.

This section additionally introduces a new method, Learning Across Domains and Devices (`LADD`), designed to solve `FFreeDA` for the SS task in the autonomous driving scenario. `LADD` leverages the knowledge of a pre-trained model by employing self-supervision with ad-hoc regularization techniques for local training and introducing a novel federated clustered aggregation scheme based on the styles of the clients. Experiments show that `LADD` can efficiently tackle the new problem, outperforming existing approaches.

This section is structured as follows:

- Section 5.3.1 details how the `FFreeDA` problem is defined.
- Section 5.3.2 briefly introduces the `LADD` algorithm.
- Section 5.3.3 describes how `LADD` pre-trains the model on a separate source dataset.
- Section 5.3.4 presents the `LADD` clustering algorithm, which groups clients together based on their average visual style.
- Section 5.3.5 describes how `LADD` aggregates the updates received from the clients in every round.

- Section 5.3.6 describes how the clients train the model locally in a semi-supervised learning manner.
- Section 5.3.7 describes how the teacher models are updated.
- Section 5.3.8 explains the LADD evaluation strategy for the test images.
- Section 5.3.9 presents the datasets used for the LADD experiments and the implementation details.
- Section 5.3.10 compares LADD with other baselines, empirically demonstrating its efficacy.
- Section 5.3.11 provides an analysis on the impact of the LADD components on the final performance.
- Section 5.3.12 shows the results of LADD when using different window sizes to extract the styles from the clients.
- Section 5.3.13 confronts different choices for the cluster-specific and global parameters when aggregating using LADD.
- Section 5.3.14 investigates the hallmarks of the different clusters of clients found by LADD.
- Section 5.3.15 shows how LADD provides good performances even in a cross-silo scenario, despite it being designed for cross-device FL.
- Section 5.3.16 shows qualitative LADD results.
- Finally, Section 5.3.17 discusses the impact of FFreeDA and LADD and possible future lines of research.

5.3.1 Federated Source-Free Domain Adaptation (FFreeDA)

Federated Source-Free Domain Adaptation (FFreeDA) is a novel FL scenario where clients only have access to unlabeled data. Before the start of the federated training, the server can pre-train the model using a separate labeled source dataset. The assumption that the server can access a separate source dataset is reasonable because pre-training on an external dataset does not compromise client privacy and, since

supervised training on labeled data is typically more effective than unsupervised training, it can enhance learning and set up for the application of domain adaptation techniques to align the model trained on the source domain to the clients' target domains. In `FFreeDA`, the source dataset is available only before the beginning of the federated training and cannot be used afterward, similar to the source-free domain adaptation setting. In other words, once the FL procedure begins, the source dataset is no longer accessible.

Crucially, this assumption is both practical and strictly necessary in real-world applications for several reasons:

- Limited-time access to source data. The server may have access to a proprietary labeled dataset available only for a limited period (*e.g.,* due to licensing agreements, data retention policies, or privacy constraints). Once this period expires, the dataset can no longer be accessed, making it impossible to use for more training.
- Resource constraints in cross-device FL. Real-world edge devices (*e.g.,* mobile phones, IoT devices) have limited computational power and storage, making it impractical to assume they can store or process large labeled datasets.
- Communication overhead and privacy concerns. Transmitting the entire source dataset to all clients would introduce significant communication costs, contradicting the efficiency goals of FL. In many scenarios, the source dataset may contain sensitive or proprietary information, making its distribution to clients infeasible due to privacy regulations.

Given these constraints, `FFreeDA` represents a realistic and necessary setting for enabling FL in resource-limited, privacy-sensitive, and large-scale deployment scenarios.

Formally, the clients have access to unsupervised, target datasets $\mathcal{D}_k^{\mathrm{T}} = \{x_i \sim p_k(\boldsymbol{x})\}_{i=1}^{n_k},\ \forall k \in \mathcal{K}$, where $x \in \mathcal{X}$. Additionally, the server is assumed to have access to a source, labeled dataset $\mathcal{D}^{\mathrm{S}} = \{(x_j, y_j) \sim p^{\mathrm{S}}(\boldsymbol{x}, \boldsymbol{y})\}_{j=1}^{n_{\mathrm{S}}}$ before the federated training, where $n_{\mathrm{S}} \in \mathbb{N}, x \in \mathcal{X}, y \in \mathcal{Y}$.

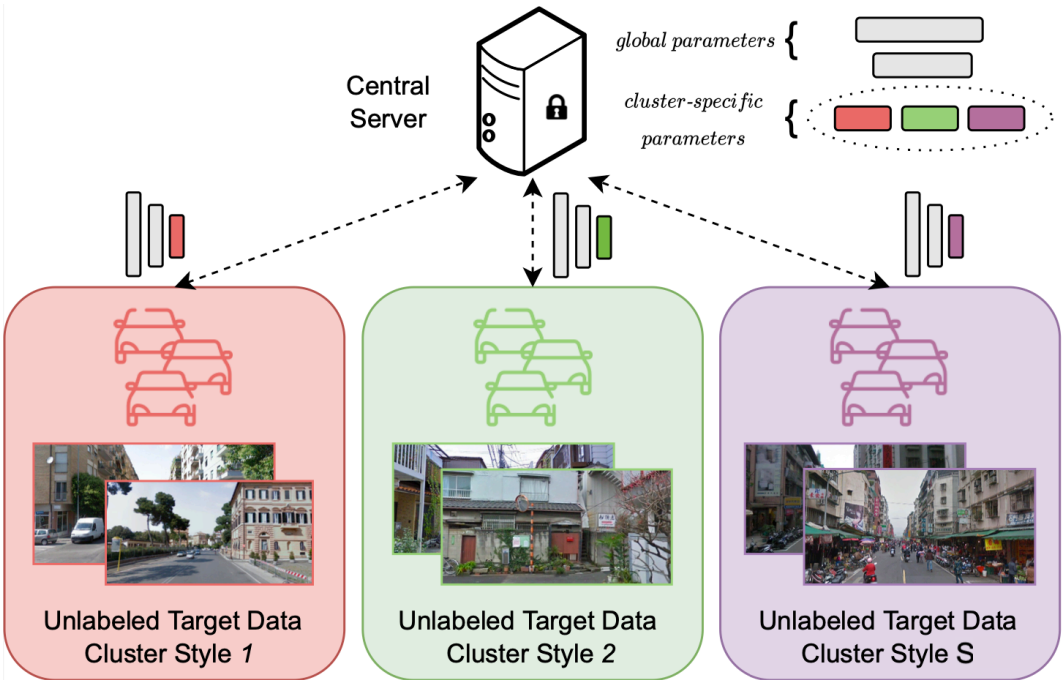


Figure 5.6: Sketch of the LADD algorrithm.

5.3.2 Introduction to the Learning Across Domains and Devices (LADD) algorithm

Learning Across Domains and Devices algorithm (LADD) is a semi-supervised learning algorithm designed to address the FFreeDA setting. A brief overview of the algorithm is illustrated in Figure 5.6.

LADD consists of two phases. In the first phase, the server pre-trains the model using a distinct source dataset. Inspired by FDA and CFSI, this dataset is augmented with styles from the clients, which are clustered based on their styles. Next, each cluster fine-tunes a portion of its model in a semi-supervised learning manner, utilizing a teacher model to generate pseudo-labels. The next sections separately describe every LADD component.

5.3.3 Pre-training on the source dataset

At first, the server pre-trains the model $f(\theta)$ on the labeled source dataset $\mathcal{D}^{\mathrm{S}}$. Parameters θ are assumed to be randomly initialized. Before the pre-training

starts, each client $k \in \mathcal{K}$ extracts its local style $a_k \in \mathbb{R}^{3\times\lfloor\beta W\rfloor\times\lfloor\beta H\rfloor}$ such that $a_{k,c} = \frac{1}{n_k}\sum_{x\in\mathcal{D}_k}\mathcal{A}(\mathcal{F}(x_c))\ \forall c \in \{1,2,3\}$. Then, the clients' styles are collected in a style bank $\mathtt{A} = \{a_k\}_{k\in\mathcal{K}}$. Importantly, these coefficients do not contain any relevant information about the scene, ensuring the user's privacy is preserved. Then, the model f is trained on the source dataset $\mathcal{D}^S$, which is augmented using `FDA` with the random styles extracted from `A`, obtaining the final pre-trained parameters θ^0.

5.3.4 Style-based clients clustering

In a realistic FL scenario, different clients may observe similar scenes. For example, self-driving cars operating in the same region likely collect similar images. In contrast, self-driving cars in distant geographical areas will probably encounter different scenes. Leveraging on this premise, `LADD` clusters the clients using the styles in `A` to find the latent visual domains.

The complete `LADD` clustering algorithm is described in Algorithm 9, which returns the clustering $\mathcal{S}$ of $\mathcal{K}$, composed of S clusters of clients such that $\bigcup_{s\in\mathcal{S}} s = \mathcal{S}$ and $s' \cap s'' = \emptyset\ \forall s', s'' \in \mathcal{S}, s' \neq s''$. Given a set of integers $\{m, m+1, \ldots, n-1\}$, the *k-means* algorithm [251] is applied using the styles of the clients in the bank `A` as the features associated to each client. The values in the set represent the desired number of clusters. For each specified number of clusters, the best clustering is determined by selecting the one with the smallest intra-cluster distance among the clustering obtained using different random seeds. Finally, the clustering with the highest *Silhouette score* [252] is selected as the overall best.

5.3.5 Aggregation policy

Unlike classical FL algorithms, `LADD` does not aggregate all the updates received in one round into one global model. After the server has computed the clusters using Algorithm 9, it partitions the model parameters θ into *global* parameters u and *cluster-specific* parameters v_s for some cluster $s \in \mathcal{S}$ such that $\theta = u \cup v_s$ and $u \cap v_s = \emptyset$. Additionally, the server defines S *student models* $f(\{u, v_s\})$ and S associated *teacher models* $f(\hat{\theta}_s)$, where u, v_s and $\hat{\theta}_s$ are initialized as the pre-trained parameters u^0, v^0 and $\theta^0 = \{u^0, v^0\}$, $\forall s \in \mathcal{S}$, and $S = |\mathcal{S}|$. At the beginning of each training round t, the server shares the global and cluster-specific parameters

Algorithm 9 - **LADD clustering.**

Require:
Clients $k \in \mathcal{K}$
Styles bank $\mathtt{A} = \{a_k\}_{k\in\mathcal{K}}$
Function $\Gamma(\mathcal{S}, k)$ that, for any clustering $\mathcal{S}$ of $\mathcal{K}$ and client $k \in \mathcal{K}$, returns the cluster $s \in \mathcal{S}$ such that $k \in s$
Hyper-parameters $n, m \in \mathbb{N}$, $m < n$
Set of random seeds $\mathcal{R}$
for each $o \in \{m, m+1, ..., n-1\}$ **do**
  for each $r \in \mathcal{R}$ **do**
    $\mathcal{S}_r^o = \mathtt{k\text{-}means}(\text{clients\_features} = \mathtt{A}, \text{num\_clusters} = o, \text{random\_seed} = r)$
    $s_k := \Gamma(\mathcal{S}_r^o, k)$
    Compute the *intra-cluster distance* of each client with its cluster:
      $\delta_k(\mathcal{S}_r^o) = \frac{1}{|s_k|-1} \sum_{h\in s_k, h\neq k} \|a_k - a_h\|_2 , \forall k \in \mathcal{K}$
  end for
  $r^* = \arg\min_{r\in\{m,m+1,...,n-1\}} \sum_{k\in\mathcal{K}} \delta_k(\mathcal{S}_r^o)$
  $\mathcal{S}^o := \mathcal{S}_{r^*}^o$, $s_k := \Gamma(\mathcal{S}^o, k)$
  Compute the *inter-cluster distance* of each client with all the other clusters:
    $\Delta_k(\mathcal{S}^o) = \min_{s\in\mathcal{S}^o\setminus s_k} \frac{1}{|s|} \sum_{h\in s} \|a_k - a_h\|_2$
  $\sigma_k^o = \frac{\Delta_k(\mathcal{S}^o)-\delta_k(\mathcal{S}^o)}{max(\Delta_k(\mathcal{S}^o),\delta_k(\mathcal{S}^o))}$ if $|s_k| > 1$, 0 otherwise, $\forall k \in \mathcal{K}$
  Compute the *Silhouette score* of $\mathcal{S}^o$:
  $\sigma^o = \frac{1}{K} \sum_{k\in\mathcal{K}} \sigma_k^o$
end for
$o^* = \arg\max_{o\in\{m,m+1,...,n-1\}} \sigma^o$
$\mathcal{S} := \mathcal{S}^{o^*}$
Return $\mathcal{S}$

u^t and v_s^t, and the teacher parameters $\hat{\theta}_s^t$, with the clients $k \in \mathcal{K}^t$. Then, each client computes the updates u_k^{t+1} and v_k^{t+1} as explained in the next section, which are then communicated to the server. At this point, the server computes the updated global parameters u^{t+1} using Equation 2.6, and the updated cluster-specific parameters v_s^{t+1} as follows:

$$v_s^{t+1} = \sum_{k\in s} \frac{n_k}{\sum_{k'\in s} n_{k'}} v_k^{t+1}, \ \forall s \in \mathcal{S}. \tag{5.17}$$

Finally, the server updates the teacher models as explained in Section 5.3.7.

5.3.6 Local training

This section describes how the clients in LADD compute their final loss function, denoted as $\mathcal{L}$. This loss function is the sum of several distinct losses, each calculated pixel-wise and then averaged. For better clarity, here it is assumed, without loss of generality, that the images consist of only one pixel. In this way, it is possible to avoid redundant subscripts for the pixels of the images and improve readability and clarity.

Firstly, the client computes the (*unsupervised*) entropy loss $\mathcal{L}_E$ to reduce the uncertainty of the predictions:

$$\mathcal{L}_{\mathrm{E}}(f(x;\theta_k^t)) = \mathcal{L}_{\mathrm{CE}}(f(x;\theta_k^t), f(x;\theta_k^t)). \tag{5.18}$$

Given any local image x, (*hard*) one-hot pseudo-labels $\tilde{y}$ are computed using the predictions $f(x;\theta_k^t)$ of the local model following the same thresholding mechanism proposed by [213]. Moreover, the teacher model computes *soft* pseudo-labels $\hat{y} = f(x;\theta_s^t)$, where s is the cluster to which the client k belongs.

Using Equation 3.1, the client computes the cross-entropy loss $\mathcal{L}_{\mathrm{CE}}(f(x;\theta_k^t), \tilde{y})$ using the one-hot pseudo-labels $\tilde{y}$. Moreover, the soft pseudo-labels $\hat{y}$ are used to compute the teacher-student loss $\mathcal{L}_{\mathrm{TS}}(f(x;\theta_k^t), \hat{y}) = \mathtt{KL}(f(x;\theta_k^t) \parallel \hat{y})$, where KL represents the Kullback–Leibler divergence [253]. The sum of the two losses is collectively referred to as $\mathcal{L}_{\mathrm{PSEUDO}}$:

$$\mathcal{L}_{\mathrm{PSEUDO}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{TS}}. \tag{5.19}$$

Finally, the server computes the *Knowledge Distillation* loss $\mathcal{L}_{\mathrm{KD}}$ [254] between the pre-trained model prediction and the local model prediction:

$$\mathcal{L}_{\mathrm{KD}} = \mathtt{KL}(f(x;\theta_k^t) \parallel f(x;\theta^0)). \tag{5.20}$$

The complete LADD loss function is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{E}} + \mathcal{L}_{\mathrm{PSEUDO}} + \lambda\mathcal{L}_{\mathrm{KD}}, \tag{5.21}$$

where $\lambda \in \mathbb{R}^+$ is a hyper-parameter.

5.3.7 Teacher models update

Every ω rounds, the server updates the cluster models. If $t < t_{\text{START}}$, where $t_{\text{START}} \in [T]$ is a hyper-parameter, the new cluster model parameters are updated as $\hat{\theta}_s^{t+1} = \theta_s^{t+1}, \forall s \in \mathcal{S}$. After the round t_{START}, inspired by the Stochastic Weight Averaging (`SWA`) method [255], they are updated using a moving average:

$$\hat{\theta}_s^{t+1} = \frac{\hat{\theta}_s^t \tau^t + \theta_s^{t+1}}{\tau^t + 1} \; \forall s \in \mathcal{S}, \tag{5.22}$$

where $\tau^t = \frac{t - t_{\text{START}}}{\omega}$. In all the rounds where $t \mod \omega \neq 0$, the teacher models are not updated, *i.e.,* $\hat{\theta}_s^{t+1} = \hat{\theta}_s^t, \forall s \in \mathcal{S}$.

5.3.8 Evaluation strategy

Before the evaluation on the test set starts, the server computes the cluster centroids μ_s:

$$\mu_s = \frac{1}{|s|} \sum_{k \in s} a_k, \; \forall s \in \mathcal{S}. \tag{5.23}$$

Then, each image x is augmented using `FDA` with the style of the nearest cluster s^*, which is the one with the smallest L2 distance between its style a_k and all the cluster centroids.

Finally, the model is initialized with the cluster parameters of s^*, *i.e.,* θ_{s^*}.

The complete `LADD` algorithm is shown in Algorithm 10.

Algorithm 10 - The LADD algorithm.

Require:
Source dataset $\mathcal{D}$
Clients $k \in \mathcal{K}$ with local target datasets $\mathcal{D}_k^{\mathrm{T}}$
Model $f(\theta)$
Hyper-parameters $\lambda, \omega, t_{\mathrm{START}}$

Server:

Extract A from the clients as described in Section 5.3.3
Calculate the clustering $\mathcal{S}$ using Algorithm 9
Pre-train $f(\theta)$ as described in Section 5.3.3, obtaining the pre-trained parameters $\theta^0 = \{u^0, v^0\}$
For each cluster $s \in \mathcal{S}$, initialize the teacher model $f(\hat{\theta}_s^0)$ and the student model $f(\{u^0, v_s^0\})$: $u^0 = u^0, v_s^0 = v^0, \hat{\theta}_s^0 = \theta^0$
for each round $T \in [T]$ **do**
  Uniformly sample $\mathcal{K}^t \subseteq \mathcal{K}$
  for each client $k \in \mathcal{K}^t$, *in parallel* **do**
    $s_k = \Gamma(S, k)$
    Send $\hat{\theta}_{s_k}^t, u^t, v_{s_k}^t$ to client k: $\mathcal{M}_{\mathcal{S}\to k}^t = \{\hat{\theta}_{s_k}^t, u^t, v_{s_k}^t\}$
    $\mathcal{M}_{k\to\mathcal{S}}^{t+1} = \{u_k^{t+1}, v_k^{t+1}\} = \texttt{ClientUpdate}(k, E, B, \eta, \hat{\theta}_{s_k}^t, u^t, v_{s_k}^t, \lambda)$
  end for
  $u^{t+1} = \sum_{k\in\mathcal{K}} \frac{n_k}{n} u_k^{t+1}$
  $v_s^{t+1} = \sum_{k\in s} \frac{n_k}{\sum_{k\in s} n_k} v_k^{t+1},\ \forall s \in \mathcal{S}$
  if $t \mod \omega \equiv 0$ **then**
    if $t \geq t_{\mathrm{START}}$ **then**
      $\tau^t = \frac{t - t_{\mathrm{START}}}{\omega}$
      $\hat{\theta}_s^{t+1} = \frac{\hat{\theta}_s^t \tau^t + \theta_s^{t+1}}{\tau^t + 1}\ \forall s \in \mathcal{S}$
    else
      $\hat{\theta}_s^{t+1} = \theta_s^{t+1}, \forall s \in \mathcal{S}$
    end if
  else
    $\hat{\theta}_s^{t+1} = \hat{\theta}_s^t, \forall s \in \mathcal{S}$
  end if
end for

Client:

ClientUpdate$(k, E, B, \eta, \hat{\theta}_{s_k}^t, u^t, v_{s_k}^t, \lambda)$ { *// run on client k*
Initialize local parameters: $\theta_k = \{u^t, v_{s_k}^t\}$
$\mathcal{B} = \{$split $\mathcal{D}_k^{\mathrm{T}}$ into batches of size $B\}$
for each $e \in [E]$ **do**
  for each $b \in \mathcal{B}$ **do**
    Compute $\mathcal{L}$ according to Equation 5.21
    $\theta_k \leftarrow \theta_k - \eta \nabla \mathcal{L}(b; \theta_k)$
  end for
end for
Return $\mathcal{M}_{k\to\mathcal{S}}^{t+1} = \{u_k^{t+1}, v_k^{t+1}\}$
}

Table 5.9: Federated datasets employed for the experiments of Section 5.3.

Dataset	C	$\lvert\mathcal{D}^T\rvert$	$\lvert\mathcal{D}^T_{test}\rvert$	K	**# Img/Client**
Cityscapes [229]	19	2975	500	144	$[10, 45]$
CrossCity [256]	13	12800	400	476	$[17, 37]$
Mapillary [257]	19	17969	2000	357	$[16, 100]$
CrossCity (split of [220])	13	12800	400	4	3200

5.3.9 Datasets and implementation details

The proposed framework is evaluated in synthetic-to-real experimental setups for autonomous driving applications, commonly used as benchmarks for domain adaptation methods. For the source domain, the synthetic GTA5 dataset [198] is selected, which contains 24966 highly realistic road scenes from typical US-like urban and suburban environments. For the real (target) domain, experiments are conducted using three different datasets: Cityscapes [229], CrossCity [256], and Mapillary Vistas [257], which consist of unlabeled training samples. Results are reported on the original validation split for Cityscapes and Mapillary and the original test split for CrossCity.

CrossCity includes more diverse locations and appearances than Cityscapes, featuring driving scenes from multiple cities worldwide, such as Rome, Rio, Tokyo, and Taipei. The Mapillary Vistas dataset contains geo-localized street-view images from around the world. The largest overlapping classes between the GTA5 and real datasets are considered, *i.e.,* 19 for Cityscapes and Mapillary and 13 for CrossCity. In the GTA5 to Cityscapes/Mapillary adaptations, 19 semantic classes are used, while in the GTA5 to CrossCity scenario, the class set is reduced to the 13 common categories.

This section proposes different federated splits for each target dataset, as summarized in Table 5.9. For Cityscapes, the heterogeneous split from Section 5.2.1 is employed, with 144 clients each observing images from a single city.

A similar split is applied to the CrossCity dataset, where each client observes images from one of the four cities in the dataset. Specifically, 27 ± 10 images taken from the same city are assigned to each client, where the number of samples per client is uniformly sampled. The final distributions of the number of images per client are shown in Figure 5.7.

For the Mapillary Vistas dataset, a novel partitioning strategy is introduced using a clustering procedure based on GPS coordinates. Thirty-one images with missing GPS coordinates are discarded from the original training set of 18000 images. The `k-means` algorithm is then applied separately to the GPS coordinates of images from each continent, forming 357 clients, each assigned samples exclusively from a single continent. The final distributions of the number of images per client are presented in Figure 5.8. Unlike the other scenarios, substantial variability is observed across continents due to the highly imbalanced nature of the dataset.

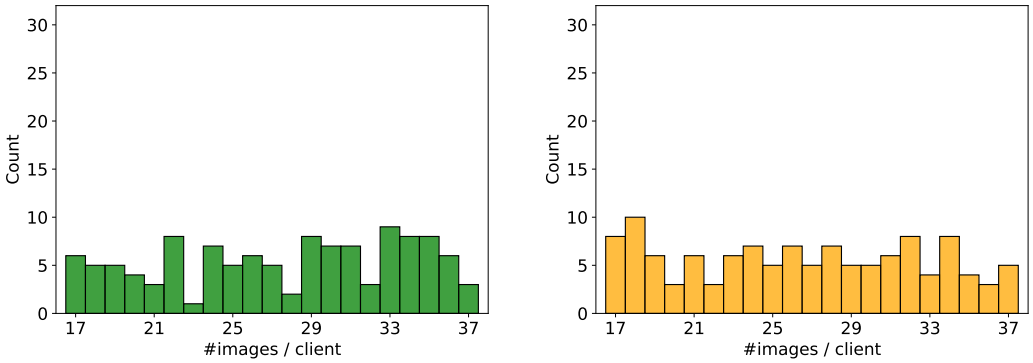
The SS model used is DeepLab-V3 [182] with a MobileNet-V2 [71] backbone and a width multiplier of 1, offering a balance between performance and efficiency—critical factors for real-world applications such as self-driving cars. During each communication round, selected clients are trained sequentially, enabling the entire simulation to be executed and reproduced on a single GPU with 32GB of VRAM (NVIDIA RTX 3090).

Comparisons are made with multiple methods in centralized and federated scenarios to assess the effectiveness of the proposed approach in the unexplored `FFreeDA` setup. The naive `source-only` approach is considered as a lower bound, where training relies solely on labeled source data.

On the other hand, Fine-Tuning Domain Adaptation (`FTDA`) and `Oracle` comparisons serve as upper bounds. Both methods assume access to supervised target data on the server side (centralized framework) and the client side (federated framework). The `FTDA` method fine-tunes the model using target data after an initial pre-training on source data. In contrast, the `Oracle` baseline consists of supervised `FedAvg` training on a labeled version of the target dataset.

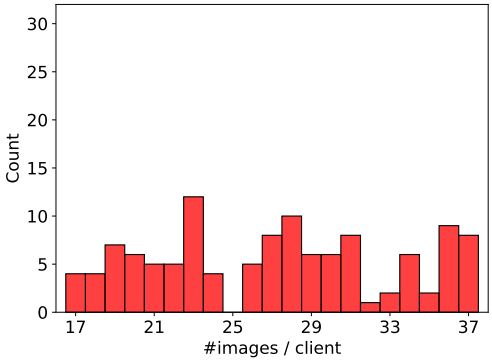
Additionally, the Maximum Classifier Discrepancy (`MCD`) method [207] is implemented and adapted to the federated setting following the approach in [220]. Comparisons are also made with DAFormer [217], which is considered the current state-of-the-art in unsupervised domain adaptation. Notably, both baselines are evaluated in an unsupervised domain adaptation setting, where source and target datasets are jointly available, making it a less challenging scenario.

The model is pre-trained on GTA5 using a power-law decreasing learning rate η, initialized at 5.0×10^{-3} with a decay power of 0.9. The optimization uses SGD with a momentum of 0.9 and no weight decay. Pre-training is conducted for 15000 steps.

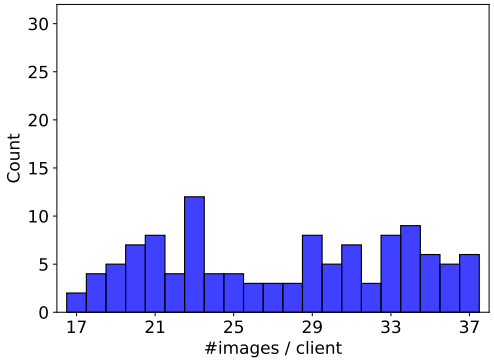


(a) Rio

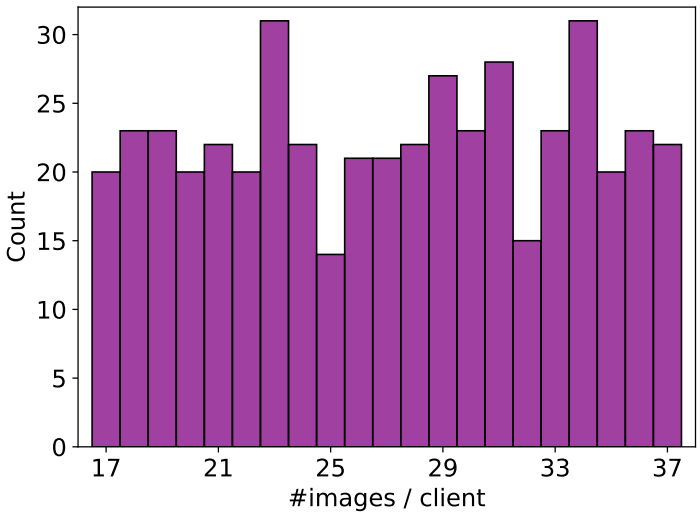
(b) Rome



(c) Taipei

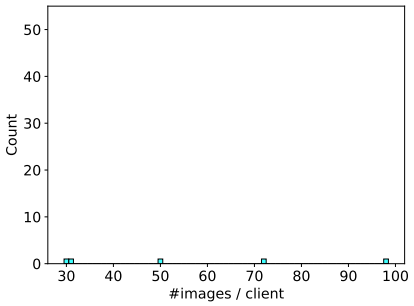


(d) Tokyo

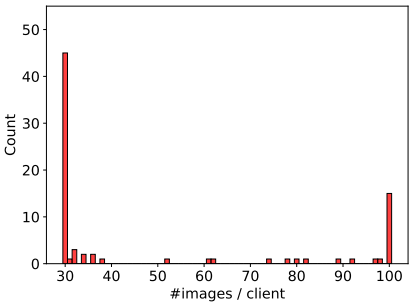


(e) Oceania

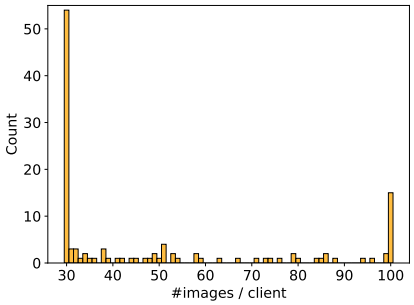
Figure 5.7: Histogram of images per clients in the proposed federated CrossCity split.



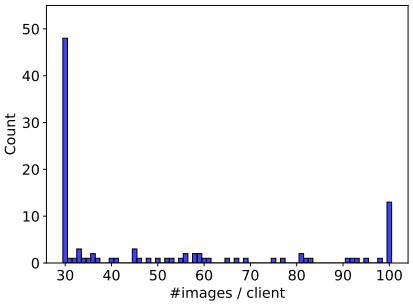
(a) Africa



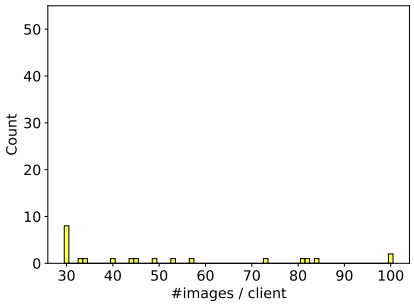
(b) Asia



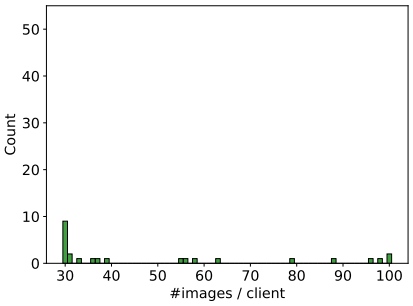
(c) Europe



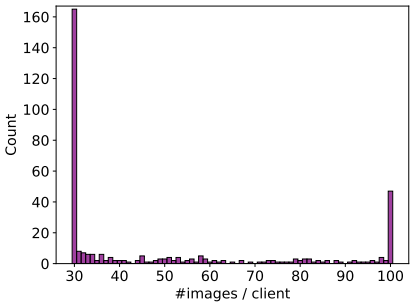
(d) North America



(e) Oceania



(f) South America



(g) Cumulative

Figure 5.8: Histogram of images per clients in the proposed federated Mapillary Vistas split.

Before the pre-training phase begins, each client computes the style of its images using a 3×3 window and transmits the mean style to the server.

For CrossCity, experiments are conducted with a fixed learning rate of $\eta = 1.0 \times 10^{-2}$, training on four clients per round for T = 1k rounds, with $\lambda = 20$. The pseudo-label teacher model is updated every round ($\omega = 1$), and $t_{\text{START}} = 400$.

For the Cityscapes experiments, training is performed on five clients per round with $T = 300$, a fixed learning rate of $\eta = 5 \times 10^{-5}$, $\lambda = 10$, $\omega = 5$, and $t_{\text{START}} = 200$.

The Mapillary experiments' learning rate is set to $\eta = 1.0 \times 10^{-2}$, with $\lambda = 10$, six clients per round, and $T = 100$. The pseudo-labeling policy follows the same approach as in Cityscapes, with $t_{\text{START}} = 50$.

Across all datasets, the batch size is set to 16. Data augmentation includes random scaling $(0.7, 2)$, random cropping to 1024×512, color jittering with brightness, contrast, and saturation set to 0.5, and image normalization. For Mapillary, images are resized to a fixed width of 1024 instead of random scaling.

5.3.10 Main experiments

GTA5 → Cityscapes

The first experimental setup focuses on adapting the GTA5 dataset to Cityscapes, with the results presented in Table 5.10. Although the GTA5 dataset provides high-quality and realistic images, there is a noticeable domain gap compared to the real-world images in Cityscapes. When training solely on the supervised source data (`Source Only`), there is a significant performance gap with the results from the experiment with full target supervision (`Oracle`). Even when using `DAFormer` in an unsupervised domain adaptation setting, where both source-labeled and target-unlabeled data are available, there is still an approximate 25% decrease in mIoU relative to the performance achieved with supervised oracle training.

The increased complexity of tasks introduced by the federated scenario is evident in the performance degradation of the `Oracle` and `FTDA`, despite their access to target supervision. In the `FFreeDA` setting, the proposed `LADD` method achieves strong results in this challenging scenario, attaining a mIoU of approximately 36.5%. The combination of the pre-training with the `FDA`-inspired augmentation and the

Table 5.10: Comparison between LADD and the baselines on the Cityscapes dataset.

Setting	Method	mIoU (%)
centralized	Oracle	66.6 ± 0.3
centralized	Source Only	24.1 ± 1.1
centralized	FTDA	65.7 ± 0.5
centralized	MCD	20.6 ± 2.7
centralized	DAFormer	42.3 ± 0.2
federated	Oracle	58.2 ± 1.0
federated	FTDA	59.4 ± 0.6
FL-UDA	MCD	10.9 ± 0.7
FFreeDA	FedAvg + self-training	35.1 ± 0.7
FFreeDA	**LADD (cls)**	**36.5** ± **0.1**
FFreeDA	**LADD (all)**	**36.5** ± **0.1**

Table 5.11: Comparison between LADD and the baselines on the CrossCity dataset.

Setting	Method	mIoU (%)
centralized	Source Only	26.5 ± 1.5
centralized	MCD	27.2 ± 0.9
FL-UDA	MCD	24.8 ± 1.6
FFreeDA	FedAvg + self-training	33.6 ± 1.3
FFreeDA	**LADD (cls)**	39.9 ± 0.1
FFreeDA	**LADD (all)**	**40.1** ± **0.2**

self-training optimization scheme effectively addresses the absence of source data at the client level. Moreover, LADD maintains a comparable performance gap to the target oracle as DAFormer does in a centralized unsupervised domain adaptation setting. Competitive results are also achieved by different variations of the proposed style-based clustering, where either only the decoder (LADD (cls)) or the entire network (LADD (all)) is treated as cluster-specific parameters during aggregation.

GTA5 → CrossCity

The performance of the proposed approach is further evaluated in the GTA5 → CrossCity adaptation, with quantitative results presented in Table 5.11. Comparisons are made against the naïve Source Only baseline and MCD. Since no target supervision

Table 5.12: Comparison between LADD and the baselines on the Mapillary dataset.

Setting	Method	mIoU (%)
centralized	Oracle	61.5 ± 0.2
centralized	Source Only	32.4 ± 0.7
centralized	MCD	31.9 ± 1.9
federated	Oracle	49.9 ± 0.5
FL-UDA	MCD	19.2 ± 0.8
FFreeDA	FedAvg + Self-Training	39.0 ± 0.2
FFreeDA	**LADD (cls)**	**40.2** ± **1.0**
FFreeDA	**LADD (all)**	38.8 ± 1.8

is available for training images, the upper bound of the target oracle and the FTDA results cannot be provided.

The CrossCity dataset introduces additional challenges due to its diverse road scenes, which originate from multiple geographic locations and result in a highly heterogeneous target distribution. Compared to the more uniform Cityscapes dataset, this increased heterogeneity further complicates federated training. For instance, a significant performance drop is observed when extending the MCD method from a centralized to a federated framework.

In contrast, the proposed method achieves considerably higher accuracy in a federated setting, surpassing federated MCD by more than 17%, while eliminating the need to reuse source data after the initial pre-training phase, demonstrating the robustness of the approach in handling the statistical diversity of client target data.

Furthermore, by allowing only a minimal subset of network parameters to be cluster-dependent, the method achieves an accuracy close to the best-performing configuration, which does not impose any parameter sharing across client clusters and highlights the efficiency of the approach, maintaining high performance while imposing minimal communication overhead compared to standard FedAvg.

GTA5 → Mapillary

The results for the adaptation from GTA5 to Mapillary are presented in Table 5.12. The target data was collected from diverse global locations and is distributed among

Table 5.13: Analysis of the impact of the LADD components on the CrossCity dataset. ST indicates self-training, KD indicates knowledge distillation, and SWAt indicates that the teacher model is updated using the SWA-inspired method introduced in Section 5.3.7.

FDA	ST	KD	SWAt	Cluster Aggregation	mIoU (%)
✗	✗	✗	✗	✗	26.5 ± 1.5
✗	✓	✗	✗	✗	30.6 ± 0.6
✓	✗	✗	✗	✗	32.4 ± 0.6
✗	✓	✓	✓	✓	32.8 ± 0.1
✓	✓	✗	✗	✗	33.6 ± 1.3
✓	✓	✓	✗	✗	37.5 ± 0.1
✓	✓	✓	✓	✗	38.8 ± 0.1
✓	✓	✓	✗	✓	39.2 ± 0.2
✓	✓	✓	✓	✓	**40.1** ± **0.2**

clients based on geographic proximity (see Section 5.3.9). This distribution introduces even higher heterogeneity in client data than in previous setups.

When utilizing target supervision with the Oracle method, a significant performance drop of approximately 11.5% in mIoU can be observed when transitioning from a centralized to a federated setting. This performance drop is even more pronounced with the MCD method, which struggles in the FL scenario, experiencing a similar decrease of around 12% in mIoU.

In contrast, the proposed LADD method demonstrates substantial performance improvements. It outperforms the Source Only baseline by more than 8% in mIoU in its best configuration, where only the classifier weights are kept cluster-specific (LADD (cls)), and it approaches the performance of the federated oracle result. Additionally, LADD surpasses the simpler method based on FedAvg + self-training, showcasing its effectiveness in addressing the domain adaptation problem within a FL framework. Finally, the lightweight aggregation scheme employing a cluster-specific classifier produces better results than LADD (all), where full-model cluster-specific parameters are required.

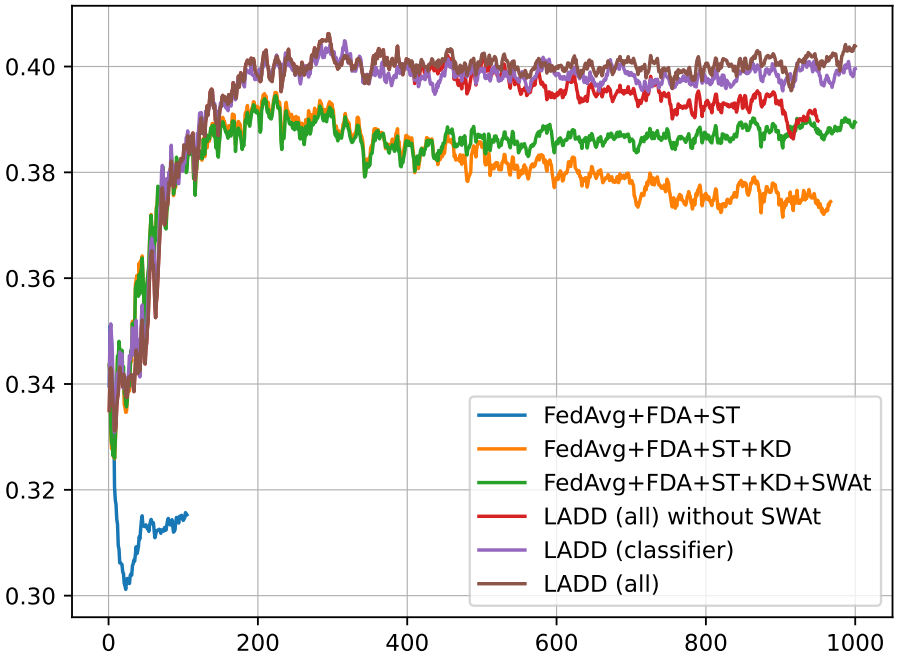


Figure 5.9: Learning curves of the LADD components on the CrossCity dataset. FDA indicates that the source dataset is augmented with the styles of the clients, ST indicates self-training, KD indicates knowledge distillation, and SWAt indicates that the teacher model is updated using the SWA-inspired method introduced in Section 5.3.7. The x-axis represents the rounds, while the y-axis represents the mIoU.

5.3.11 Impact of the LADD components

In Table 5.13, the target mIoU is reported with the LADD components added incrementally to highlight the contribution of each module in the GTA5 → CrossCity setup.

The introduction of the style transfer technique during server pre-training leads to a significant improvement of nearly 6% in mIoU with the lower bound (`Source Only` pre-training) matching the results of the centralized `Source Only` experiment. The naïve federated adaptation using only the self-training methods presented in Section 5.3.6, along with `FedAvg` aggregation, provides an initial 2% improvement in mIoU but results in unstable training curves, as shown in Figure 5.9. When self-training is employed only in client-side optimization, the training is highly

Table 5.14: IoU by class and mIoU (%) while increasing the number of LADD components on the CrossCity dataset. FDA indicates that the source dataset is augmented with the styles of the clients, ST indicates self-training, KD indicates knowledge distillation, SWAt indicates that the teacher model is updated using the SWA-inspired method introduced in Section 5.3.7, and Cl. Aggr. indicates that the aggregation is performed cluster-wise with global and cluster-specific parameters as in the complete LADD algorithm.

FDA	ST	KD	SWAt	Cl. Aggr.	road	sidewalk	building	traffic light	traffic sign	vegetation	sky	person	rider	car	bus	motorcycle	bicycle	mIoU
✗	✗	✗	✗	✗	25.6	21.6	65.9	3.9	8.6	67.5	73.5	33.1	2.1	43.0	6.6	0.3	0.2	26.5 ± 1.5
✓	✗	✗	✗	✗	38.2	24.0	74.8	7.0	8.9	70.5	80.9	37.0	**4.0**	63.6	12.0	3.5	0.0	32.4 ± 0.6
✓	✓	✗	✗	✗	21.9	17.9	81.3	9.5	14.5	77.4	85.2	41.0	2.3	66.1	10.7	8.0	0.9	33.6 ± 1.3
✓	✓	✓	✗	✗	49.5	26.7	81.2	11.7	12.4	**77.6**	87.0	40.4	1.0	68.9	16.3	11.3	3.4	37.5 ± 0.1
✓	✓	✓	✓	✗	53.3	28.6	81.1	12.1	12.0	77.5	**87.1**	**42.1**	1.9	68.9	17.2	**16.7**	**4.9**	38.8 ± 0.1
✓	✓	✓	✗	✓	63.4	32.3	**81.5**	12.1	12.2	77.5	86.9	41.0	1.1	69.0	16.0	12.5	3.8	39.2 ± 0.2
✓	✓	✓	✓	✓	**64.3**	**33.7**	81.0	**12.5**	**14.4**	77.2	86.8	**42.1**	1.4	**69.1**	**18.1**	15.6	4.8	**40.1** ± 0.2

unstable, with a small initial performance burst followed by a rapid decline after a few rounds.

Adding knowledge distillation further boosts performance by almost 3% mIoU, helping stabilize the optimization process. Moreover, updating the teacher model using the SWA-inspired method introduced in Section 5.3.7 increases mIoU and training stability, ensuring consistent convergence during training. Notably, the standard deviation drops significantly when knowledge distillation and the SWA-inspired teacher models update are activated.

When cluster aggregation is introduced, there is a slight increase in mIoU, albeit at the cost of higher instability. Finally, when both cluster aggregation and the SWA-inspired update method are applied, LADD achieves a final mIoU of 40.1%.

Notably, the complete LADD configuration demonstrates steady, converging learning curves, highlighting the effectiveness of the proposed approach.

Finally, Tables 5.14 to 5.16 show the per-class IoUs achieved when different components of LADD are enabled. As it is possible to observe, each component improves the overall mIoU score, and typically, it also improves the single IoU scores.

Table 5.15: IoU by class and mIoU (%) while increasing the number of LADD components on the Cityscapes dataset. FDA indicates that the source dataset is augmented with the styles of the clients, ST indicates self-training, KD indicates knowledge distillation, SWAt indicates that the teacher model is updated using the SWA-inspired method introduced in Section 5.3.7, and Cl. Aggr. indicates that the aggregation is performed cluster-wise with global and cluster-specific parameters as in the complete LADD algorithm.

FDA	ST	KD	SWAt	Cl. Aggr.	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU
✓	✓	✗	✗	✗	**84.5**	**36.8**	**77.3**	**23.9**	11.3	20.5	29.1	22.6	**76.9**	**26.5**	68.9	**53.4**	13.7	79.0	**15.2**	14.0	1.4	11.0	5.1	35.1 $\pm$ 0.7
✓	✓	✓	✗	✗	79.3	34.0	73.6	22.0	16.4	24.6	30.3	31.3	61.7	23.2	**70.1**	51.2	**19.3**	73.7	13.6	17.9	**7.3**	12.1	**15.3**	35.6 $\pm$ 0.1
✓	✓	✓	✓	✓	80.0	36.1	74.1	22.8	**18.3**	**26.3**	**30.6**	**33.0**	65.2	25.4	69.4	52.3	19.1	**74.5**	13.4	**18.0**	7.2	**12.6**	14.2	**36.5** $\pm$ 0.1

Table 5.16: IoU by class and mIoU (%) while increasing the number of LADD components on the Mapillary dataset. FDA indicates that the source dataset is augmented with the styles of the clients, ST indicates self-training, KD indicates knowledge distillation, SWAt indicates that the teacher model is updated using the SWA-inspired method introduced in Section 5.3.7, and Cl. Aggr. indicates that the aggregation is performed cluster-wise with global and cluster-specific parameters as in the complete LADD algorithm.

FDA	ST	KD	SWAt	Cl. Aggr.	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU
✓	✓	✗	✗	✗	67.4	36.9	**74.7**	24.8	25.4	10.9	21.0	33.3	72.8	**40.8**	91.2	46.1	**23.1**	**73.7**	31.1	**22.7**	3.1	30.6	11.9	39.0 $\pm$ 0.2
✓	✓	✓	✓	✗	75.4	**37.7**	73.4	25.2	**25.2**	18.3	26.6	37.1	**73.5**	38.1	**91.4**	45.5	13.8	71.3	30.9	22.0	3.0	29.9	19.1	40.0 $\pm$ 0.1
✓	✓	✓	✓	✓	**75.5**	37.0	69.1	24.6	**25.6**	**18.9**	**26.7**	**38.2**	72.5	36.4	89.4	**46.3**	17.2	70.7	**32.6**	20.2	**4.1**	**31.4**	**21.0**	**40.2** $\pm$ 1.0

5.3.12 Effects of the style-based augmentation in the pre-training phase

Table 5.17 shows the impact of the augmentation mechanism based on style translation during pre-training by evaluating different sizes of the amplitude window, measured in number of pixels, for the GTA5 $\rightarrow$ CrossCity setup. The findings reveal that a 1×1 window is sufficient to capture and transfer useful domain-dependent information across domains. Additionally, increasing the size of the style window to 3×3 and 5×5 pixels results in a 1% improvement in mIoU. While the 3×3 window yields similar final mIoU results as the 5×5 size, it provides more stable pre-training, as evidenced by a lower standard deviation due to fewer artifacts in the reconstructed images.

Notably, the style data occupies a minimal amount of memory (in the order of a few bytes), requiring minimal communication overhead to be transmitted from client

Table 5.17: Sensitive analysis on the window size used for style-translation for the pre-training, introduced in Section 5.3.3.

Size	mIoU (%)
0×0	26.5 $\pm$ 1.5
1×1	31.6 $\pm$ 0.7
3×3	32.4 $\pm$ 0.6
5×5	**32.5** $\pm$ **0.8**

Table 5.18: Results with different choices for the cluster-specific parameters on the GTA5 $\rightarrow$ CrossCity scenario.

Cluster-specific parameters	mIoU (%)
None	38.8 $\pm$ 0.1
BN	38.7 $\pm$ 0.2
Backbone	39.3 $\pm$ 0.1
Head	39.9 $\pm$ 0.1
All	**40.1** $\pm$ **0.2**

to server before federated rounds begin. The style window corresponds to a small portion of the amplitude spectrum of the Fourier transform, and prior to transmission, it is averaged over all data samples within each client.

5.3.13 On the Clustered Federated Learning aggregation for LADD

Table 5.18 shows the impact of different cluster-based aggregation schemes on the final mIoU by testing various groups of model parameters as the cluster-specific and global parameters. Results from the GTA5$\rightarrow$CrossCity setup are presented in Table 5.18.

Keeping only the batch normalization parameters cluster-dependent (second row) yields performance similar to the standard `FedAvg` training (first row), where all model parameters are global. When the backbone or the head blocks are treated as cluster-specific (third and fourth rows), the mIoU improves. Performance is further slightly boosted when the entire model is cluster-specific.

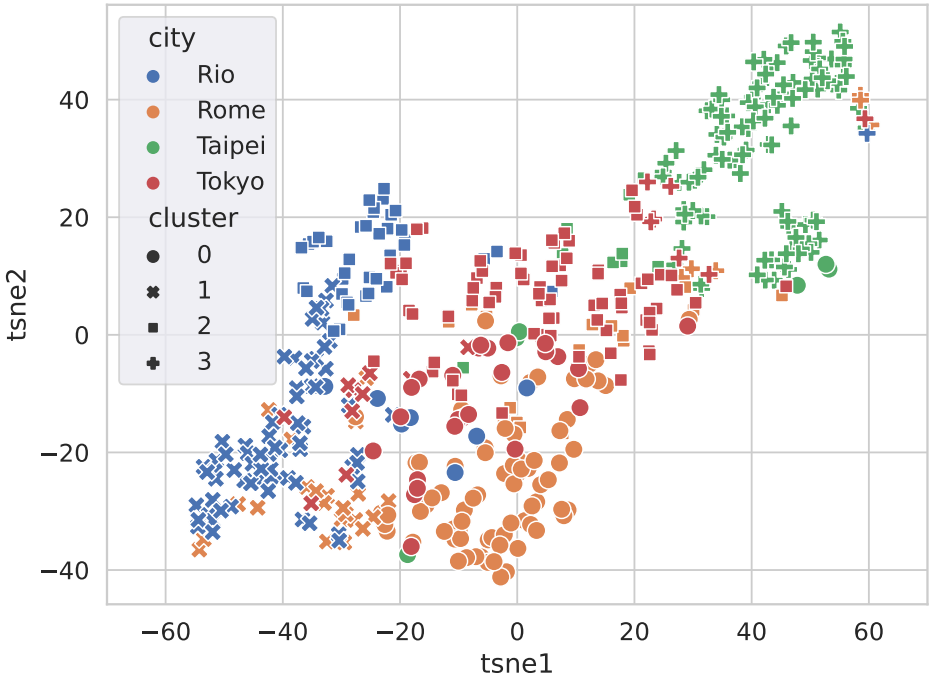


Figure 5.10: tSNE visualization [258] of the styles of the clients for the CrossCity dataset. The colors represent the different cities, while the symbols represent the clusters. The clustering accuracy, considering each cluster label as the city of the majority of the clients of the cluster, is approximately 68%.

5.3.14 Analysis of the clustering based on the styles of the clients

In a realistic FL setting, different clients may still observe similar samples. For example, self-driving cars operating in the same area are likely to collect similar images, which means they are not affected by statistical heterogeneity during server aggregation. To group together the clients whose local datasets share similar visual features, a style-driven client clustering is integrated as a foundational component of `LADD`.

Interestingly, with the CrossCity dataset, `LADD` finds four as the best number of clusters, the same number of cities of CrossCity. Moreover, the four identified clusters from the images predominantly consist of clients from one single city.

Figure 5.10 illustrates the distribution of the clusters and cities for the CrossCity dataset with a tSNE [258] visualization generated from the clients' styles. Specifically,

Table 5.19: Number of clients belonging to a specific city assigned to each cluster in the CrossCity scenario.

City	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Rome	**76**	22	17	6
Rio	7	**70**	38	1
Tokyo	26	10	**73**	8
Taipei	6	0	9	**103**

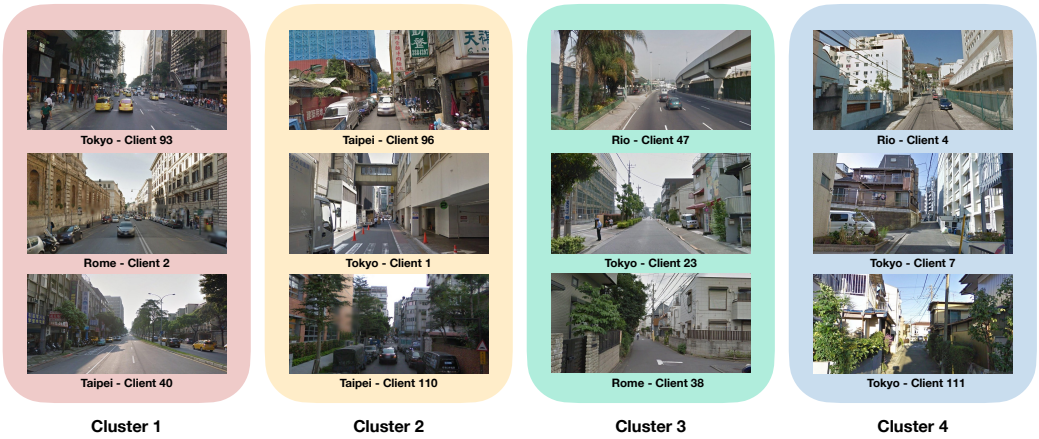


Figure 5.11: Some examples of images from different cities, from each of the four clusters identified by the LADD algorithm in the CrossCity split.

each style tensor is flattened, and the tSNE dimensional reduction method is applied to project it into a 2D space.

Additionally, Table 5.19 shows the number of clients from each specific city assigned to each cluster in the same scenario. Overall, the clustering accuracy, viewing each cluster as representing a city, is 68%, indicating some correlation but not a one-to-one correspondence between the clusters and the cities.

To investigate this aspect, Figure 5.11 presents samples from clients belonging to each of the four clusters in the federated CrossCity dataset. As expected, the clusters tend to group images with similar semantic characteristics. For instance, *Cluster 1* predominantly includes images depicting large, congested streets under a grayish sky. *Cluster 2* consists of images featuring narrow streets with minimal vegetation, numerous buildings, a few parked cars, and a whitish sky. *Cluster 3* primarily contains images of empty roads surrounded by green vegetation. Lastly,

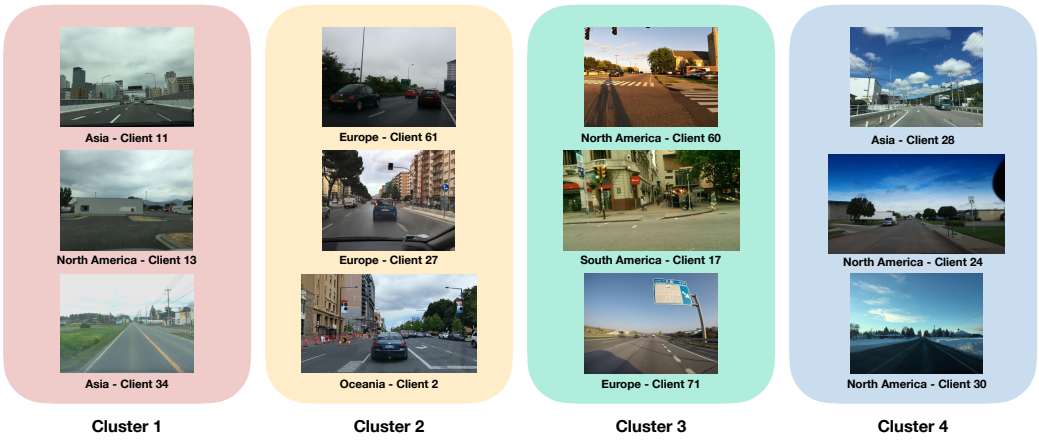


Figure 5.12: Some examples of images from each of the four clusters identified by the LADD algorithm in the Mapillary split.

Cluster 4 comprises images captured in sunny weather with a blue sky, narrow streets with no traffic, and green vegetation.

Similarly, Figure 5.12 shows a similar clustering for the Mapillary dataset. Unlike in the CrossCity dataset, a clear correspondence between clusters and specific locations is not observed, as the optimal number of clusters found by LADD does not align with the number of towns or continents. Again, the clustering appears primarily driven by visual characteristics, grouping images based on stylistic similarities. For instance, *Cluster 1* predominantly includes images captured in cloudy or foggy conditions, resulting in a grayish appearance. *Cluster 2* consists of images featuring a grayish sky and yellowish buildings, with some degree of semantic similarity across clients. *Cluster 3* contains images taken at sunrise or sunset, characterized by warm yellow-toned lighting. *Cluster 4* comprises images where blue hues dominate the sky.

5.3.15 LADD effectiveness in a cross-silo scenario

This section compares LADD with the DualAdapt method proposed by [220] in the GTA5 to CrossCity adaptation setup they propose, where the target data is distributed across four clients only, with each client containing images from one of the four cities in CrossCity. The same segmentation network chosen in [220] is adopted for these experiments to ensure a fair comparison.

Table 5.20: Results on the CrossCity split proposed in [220].

Method	Rio	Rome	Taipei	Tokyo	Avg
`Source Only`	27.9	27.6	26.0	28.2	27.4
`Cent-MCD` [207]	31.3	30.6	28.8	31.6	30.5
`Fed-DAN` [259]	27.3	26.4	26.0	28.5	27.1
`Fed-DANN` [204]	28.6	26.0	26.6	28.6	27.5
`Fed-MCD` [207]	27.7	27.3	26.5	29.0	27.6
`DualAdapt` [220]	29.2	28.0	27.6	30.7	28.9
`LADD`	**35.4**	**34.0**	**31.5**	**32.4**	**33.3**

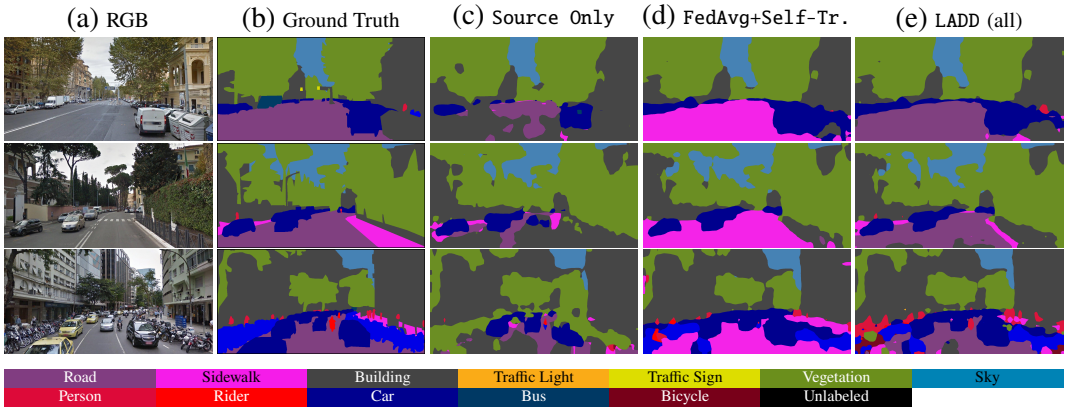


Figure 5.13: GTA5→CrossCity qualitative results.

In this simplified federated setting, a restricted version of our method, which does not include cluster-level aggregation, is still effective. Table 5.20 shows the mIoU computed for each city and the average value for various methods in [220]. `LADD` guarantees a consistent performance improvement, with the average target mIoU increasing by almost 5%, thereby demonstrating the superiority of the proposed approach.

5.3.16 Qualitative results

Qualitative results are presented as segmentation maps of target images generated by the segmentation model under different adaptation schemes. Figures 5.13 to 5.15 show the predictions for some sample images from CrossCity, Cityscapes, and Mapillary as target datasets, respectively. The comparison includes naïve source-only

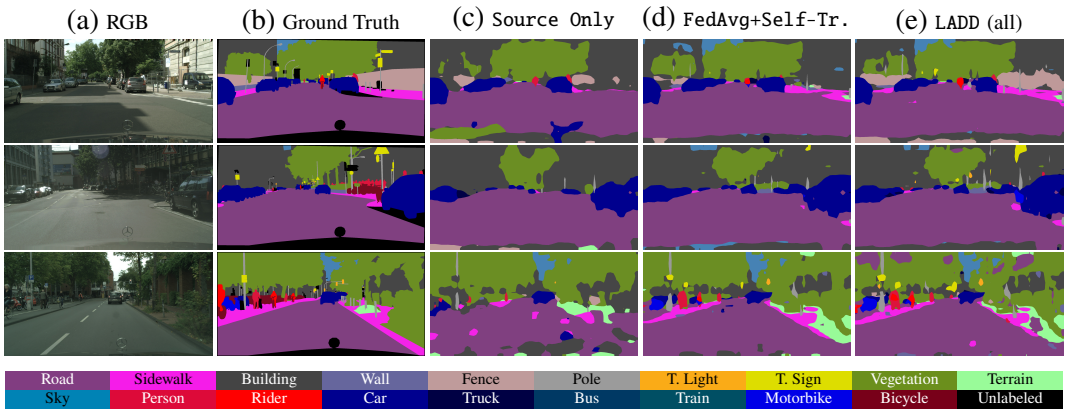


Figure 5.14: GTA5→Cityscapes qualitative results.

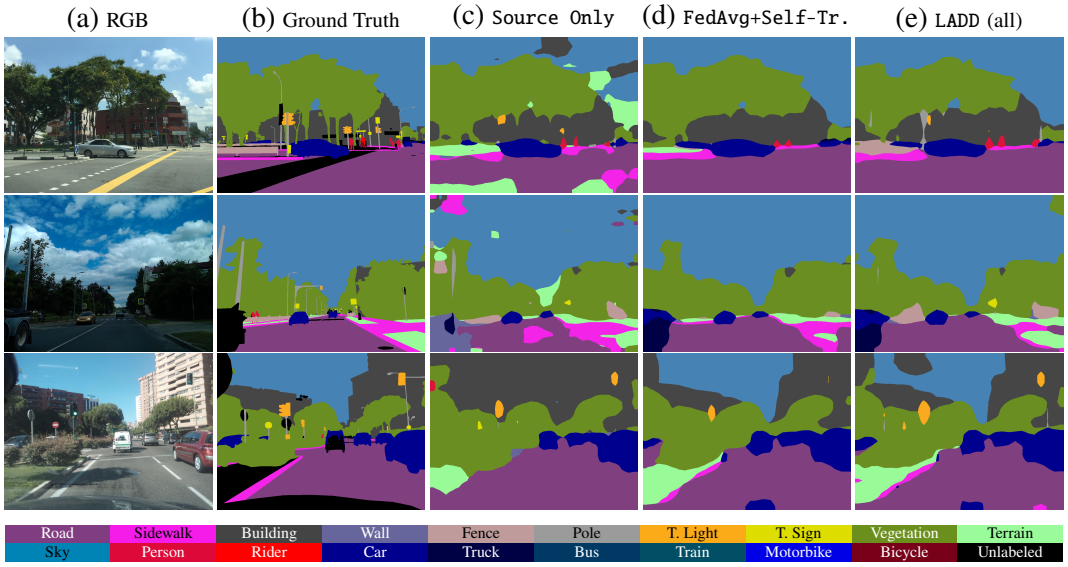


Figure 5.15: GTA5→Mapillary qualitative results.

training (third columns in all referenced figures), a baseline federated adaptation strategy (fourth columns) based on `FedAvg` aggregation with local self-training, and the proposed `LADD` method, where cluster-specific aggregation is extended to all layers of the segmentation network (last columns).

The segmentation maps reveal that the source-only model produces inconsistent and noisy predictions, frequently confusing semantically similar classes such as *sidewalk* and *road* or *terrain*, as observed across all reported samples. While local self-training combined with standard `FedAvg` aggregation partially mitigates the mIoU degradation caused by domain shift, misclassifications of semantically similar

classes persist, as evident in the first sample of Figure 5.13. Conversely, LADD enhances optimization stability, which would otherwise be hindered by the limited amount of training data and the absence of supervision on the client side, leading to more accurate and less noisy predictions.

5.3.17 Impact and future work

FFreeDA is a challenging and realistic setting for Source-Free Domain Adaptation in FL for SS. In FFreeDA, a server-side labeled dataset is used for pre-training, while local training relies solely on clients' unlabeled data. To address this problem, LADD was proposed, integrating 1) style transfer, 2) knowledge distillation, 3) SWA-based teacher regularization on pseudo-labels, and 4) style-driven clustering to learn both global and personalized parameters.

Since FFreeDA introduces a novel problem formulation, LADD has no direct competitors. However, it achieves performance comparable to state-of-the-art methods designed for less challenging settings. Additionally, two new federated datasets have been introduced, adapting the CrossCity and Mapillary Vistas datasets to the federated scenario, providing a benchmark for future research in FL for SS. These contributions establish a foundation for further exploration of adaptive learning strategies in federated settings, including more robust personalization techniques and improved unsupervised adaptation methods.

Chapter 6

Conclusion

6.1 Main contributions

This manuscript focuses on solutions to mitigate the adverse effects of statistical heterogeneity in FL, particularly in real-world computer vision applications. In FL, the goal is to train a global model that generalizes well across all clients without exposing their data. However, since client data is typically non-i.i.d., statistical heterogeneity arises, posing a fundamental challenge. Indeed, local clients' updates can become biased toward specific individual data, and if the data distribution varies widely, these updates may significantly diverge from each other — an issue referred to as client drift [16].

Prior research [75] has empirically demonstrated that FL training would be significantly more stable, converge faster, and perform better if clients had access to uniform data. However, unlike traditional DL, where data can be freely allocated across the clients, FL must preserve clients' privacy. Therefore, the server cannot redistribute data to balance distributions. As a possible way to mitigate the negative consequences of statistical heterogeneity and client drift, [91] shows that pre-trained architectures help stabilize the FL training. However, other works [92, 93] show that the client drift issue primarily affects the heads of the models, hampering the predictive capabilities even in the presence of a robust, eventually pre-trained feature extractor.

This manuscript first addresses client drift in a general FL setting for computer vision, using image classification as the representative task. To this end, Chapter 3

introduces Federated Recursive Ridge Regression (Fed3R), a gradient-free method designed to construct a robust prediction head leveraging a pre-trained feature extractor. In Fed3R, clients compute local feature embeddings, which are used to extract ridge regression statistics and are shared with the server. The server then constructs the optimal regularized least squares classifier, equivalent to the centralized classifier that can be achieved in a centralized setting if all the client data is simultaneously available. Given that Fed3R solution is equivalent to the centralized ridge regression solution, it is completely immune to statistical heterogeneity.

However, the performance of the model relies on the quality of the pre-trained feature extractor when making predictions based on clients' data. If the distribution of the pre-trained data differs significantly from that of the target clients' data, the quality of the feature maps may not be adequate to create an effective classifier. To address this problem, this manuscript empirically demonstrates that Fed3R classifiers can be fine-tuned using a different gradient-based FL algorithm. Experiments show that this initialization largely improves convergence speed and robustness while drastically reducing communication and computational costs up to 1000 times compared with the other methods.

One alternative strategy proposed to address statistical heterogeneity is allowing each client to find the optimal model that fit its local data distribution. This represents a paradigm shift, as the goal is no longer to find a single global model, but rather to find one optimal model for each client. This approach is referred to in the literature as Personalized FL [19]. In PFL, particularly in cross-device scenarios, finding the right balance between leveraging the global knowledge obtained by training on all clients' data and fine-tuning the model on local distributions is crucial. On the one hand, clients in cross-device scenarios have limited data samples and can benefit from standard FL training because the model will also be trained on data from other clients. However, on the other hand, the varying distributions of data among clients can lead to client drift. Consequently, from this perspective, it could be more advantageous to fine-tune the model locally. Yet, with too few data points, there is a high risk of overfitting.

Chapter 4 studies the trade-off between utilizing global and local knowledge in PFL systematically, by dividing the experiments into four distinct phases. The results indicate that initializing the model with Fed3R, then personalizing the clients models with PP, and finally fine-tuning the local models, results in an improved

performance-to-cost ratio, allowing for a reduction of up to two-thirds in the number of training rounds required to achieve the same results if not using `Fed3R` initialization. Furthermore, Chapter 4 introduces `OLL`. This simple yet effective algorithm significantly reduces the complexity and the possible sources of errors in local classifiers. By initializing the classifier with `Fed3R` parameters, filtering its parameters using `OLL`, and then fine-tuning it locally, it is possible to reach WMAs comparable with other costly strategies with negligible costs. Moreover, the best strategies in terms of performance-to-cost ratio include both `Fed3R` initialization and `OLL` filtering.

Chapters 3 and 4 provide a general solution to statistical heterogeneity in computer vision, with the potential for domains outside computer vision. However, real-world computer vision applications may pose specific challenges. Chapter 5 studies explicitly one of these scenarios, the one of SS for autonomous driving. Here, clients may collect data under varying conditions, such as different locations and weather, and use different hardware to record the scenes, exacerbating statistical heterogeneity and hindering model convergence. This chapter introduces `FedDrive`, the first FL benchmark for SS for autonomous driving. `FedDrive` introduces 12 FL datasets based on the Cityscapes [229] and IDDA [200] datasets with different complexities and compares several methods for domain generalization, including style transfer techniques and methods designed to improve the management of batch normalization statistics in FL.

The setting of `FedDrive` is fully supervised and intended to serve as a reference for future research. However, autonomous vehicles cannot access ground truth labels in real-world applications, as labeling is a complex and costly process. Chapter 5 additionally introduces Federated Source-Free Domain Adaptation (`FFreeDA`), a new real-world FL problem for SS in the context of autonomous driving cars where clients do not have access to ground-truth labels, and Learning Across Domains and Devices (`LADD`), a new algorithm to solve `FFreeDA`. `LADD` has no direct competitors in the literature due to the novelty of the `FFreeDA` problem, but results show that it still outperforms the provided baseline and adaptation of other methods to this setting. Moreover, two additional splits adapting the CrossCity [256] and Mapillary Vistas [257] datasets have been provided, which could also be helpful to future research.

6.2 Possible limitations and future works

The effectiveness of Fed3R depends on the quality of the pre-trained feature extractor. Future research could focus on developing a Fed3R classifier that allows for the joint fine-tuning of the feature extractor, or even the entire model, while extracting Fed3R statistics to update the Fed3R classifier. Additionally, future research could explore closed-form approaches similar to Fed3R to enhance the quality of the feature extractor. Fed3R could also be adapted for other settings and tasks, enabling evaluations of its effectiveness in computer vision domains other than image classification, such as those seen in FedDrive, and even in areas beyond computer vision. Furthermore, Fed3R could be extended to unsupervised or semi-supervised learning contexts, like in FFreeDA, by integrating it with techniques such as pseudo-labeling.

Regarding the FedDrive benchmark, it is designed specifically for supervised learning methods. A potential future direction could adapt unsupervised and semi-supervised learning algorithms using the same proposed FL datasets. Moreover, LADD in the FFreeDA setting could be improved by reducing its reliance on pre-trained models, which are currently used to extract styles for clustering and data augmentation. Additionally, at present, LADD clusters adapt independently. Implementing cross-cluster knowledge distillation or inter-cluster regularization could facilitate the transfer of valuable knowledge across stylistically diverse groups without violating the source-free constraint.

Finally, since the interest in FL applied to computer vision continues to grow, it is essential for future research to address the hidden nuances and challenges related to heterogeneity in other domains, including tasks like object detection [260], pose estimation [261], panoptic segmentation [262], and image retrieval [263].

List of Figures

List of Tables

List of Acronyms

Acronym	Description
AI	Artificial Intelligence
IoT	Internet of Things
GDPR	General Data Protection Regulation
FL	Federated Learning
DL	Distributed Learning
i.i.d.	independent and identically distributed
`FedAvg`	Federated Averaging [13]
`FedOpt`	Adaptive Federated Optimization [18]
`Fed3R`	Federated Recursive Ridge Regression
PFL	Personalized Federated Learning
`OLL`	Only Local Labels
`FedDrive`	FedDrive benchmark
`FFreeDA`	Federated Source-Free Domain Adaptation
`LADD`	Learning Across Domains and Devices
EB	Exabytes
TB	Terabytes
ZB	Zettabytes
OOM	Orders Of Magnitude
HFL	Horizontal Federated Learning
VFL	Vertical Federated Learning
FTL	Federated Transfer Learning
ERM	Empirical Risk Minimization
SGD	Stochastic Gradient Descent
RGB	Red, Green, Blue

CE	Cross-Entropy
SQ	Squared
RR	Ridge Regression
KRR	Kernel Ridge Regression
RBF	Radial Basis Function
RFF	Random Fourier Features
`FedNCM`	Federated Nearest Class Means [132]
`Fed3R-RF`	Federated Recursive Ridge Regression with RFF approximation
`Fed3R-Sync`	Synchronous `Fed3R`
`FedProto`	`FedProto` algorithm from [134] or [195]
`Scaffold`	Stochastic Controlled Averaging for FL [16]
`FedDyn`	FL based on Dynamic Regularization [62]
`Fed3R-RF-Sync`	Synchronous `Fed3R-RF`
ψ`-init`	Classifier initialization
`FT`	Federated Fine-Tuning
`FedAvgM`	`FedAvg` with server momentum [118]
`Mime`	Mimicking Centralized Stochastic Algorithms in FL [77]
FP32	32-bit floating point
FLOPs	Floating Point Operations per Second
MFLOPs	MegaFLOPs
`RF-RR`	RR using RFF
MB	Megabytes
GB	Gigabytes
GFLOPs	GigaFLOPs
TFLOPs	TeraFLOPs
`FedRDN`	FedRDN algorithm [157]
`OLL`	Only Local Labels
CFL	Clustered Federated Learning
`PP`	Partial Personalization
`FP`	Full Personalization
`FedSim`	FedSim algorithm [170]
`FedAlt`	FedAlt algorithm [170]

pFedMe	Personalized Federated Learning with Moreau Envelopes [166]
Ditto	Ditto algorithm [167]
SS	Semantic Segmentation
DG	Domain Generalization
DA	Domain Adaptation
BN	Batch Normalization
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
mIoU	mean Intersection over Union
IoU	Intersection over Union
SFDA	Source-Free Domain Adaptation
UDA	Unsupervised Domain Adaptation
FDA	Fourier Domain Adaptation
MNIST	Modified National Institute of Standards and Technology dataset [222]
CFSI	Continuous Frequency Space Interpolation [230]
MMD	Maximum Mean Discrepancy [231]
LAB	Lightness green-red blue-yellow color space [232, 233]
IDDA	Italdesign dataset [200]
FedRobust	FedRobust algorithm [234]
FedBN	FedBN algorithm [236]
SiloBN	SiloBN algorithm [235]
AdaBN	AdaBN algorithm [238]
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IDFT	Inverse DFT
GPS	Global Positioning System
KL	Kullback-Leibler
TS	Teacher-Student
KD	Knowledge Distillation
GTA5	Grand Theft Auto V dataset [198]
FTDA	Fine-Tuning Domain Adaptation
MCD	Maximum Classifier Discrepancy [207]

DAFormer	DAFormer algorithm [217]
tSNE	t-distributed Stochastic Neighbor Embedding
DualAdapt	DualAdapt algorithm [220]

List of Symbols

Notation	Description
$\mathcal{S}$	Server
$\mathcal{K}$	Set of all the clients in the federation
$K \in \mathbb{N}$	Number of clients in the federation, *i.e.*, $K = \lvert\mathcal{K}\rvert$
$k \in \mathcal{K}$	Client identifier
$\mathcal{D}_k$	Local dataset of client k
$n_k \in \mathbb{N}$	Number of samples in the local dataset $\mathcal{D}_k$, *i.e.*, $\lvert\mathcal{D}_k\rvert = n_k$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Target space
$x \in \mathcal{X}$	Generic input
$y \in \mathcal{Y}$	Generic target
$f : \mathcal{X} \to \mathcal{Y}$	Model function
Θ	Parameters space
$\theta \in \Theta$	Parameters of f
$\theta^* \in \Theta$	Optimal parameters for some minimization problem
$\boldsymbol{x}$	1) Random variable associated with the inputs 2) Random variable associated with the number of times a specific client is sampled
$\boldsymbol{y}$	Random variable associated with the targets
p	1) Global distribution; 2) Generic input dimensionality for a linear predictor, $p \in \mathbb{N}$
p_k	Local distribution of client k

$q_k \in [0,1]$	Weight associated to client k in the global FL distribution
$\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$	Loss function
$\mathcal{D}$	Union of all the local datasets of all the clients, *i.e.*, $\mathcal{D} = \bigcup_{k \in \mathcal{K}} \mathcal{D}_k$
$T \in \mathbb{N}$	1) Number of FL rounds 2) Last iteration of recursive algorithms
t	1) Round index $t \in [T]$ 2) Index for recursive algorithms
$\mathcal{M}_{i \to j}^t$	Message communicated from node i to node j at round t
$\mathcal{K}^t \subseteq \mathcal{K}$	Subset of active clients at round t
θ^t	Global parameters at round t
θ_k^{t+1}	Local parameters of client k at the end of round t
$\kappa \in [K]$	Constant number of random clients activated in every round
$E \in \mathbb{N}$	Number of local epochs
$n_t \in \mathbb{N}$	Total number of images in all the clients in $\mathcal{K}^t$, *i.e.*, $n_t = \sum_{k \in \mathcal{K}^t} n_k$
$\Delta_k^{t+1} \in \Theta$	Pseudo-gradient of client k at round t
$\Delta^{t+1} \in \Theta$	Aggregate pseudo-gradient ta round t
$\eta_{\mathcal{S}} \in \mathbb{R}^+$	Server learning rate
$\eta \in \mathbb{R}^+$	Clients learning rate
W	1) Image width, measured in number of pixels, $W \in \mathbb{N}$ 2) Least squares or RR parameters, $W \in \mathbb{R}^{p \times C}$
$H \in \mathbb{N}$	Image height, measured in number of pixels
$\mathcal{C}$	1) Set of all the possible classes 2) Function $\mathcal{C}(c, y)$ that, given a class $c \in \mathcal{C}$ and a SS ground truth (or prediction) y, returns the set of pixels of class c in y
$C \in \mathbb{N}$	Number of classes of $\mathcal{C}$, *i.e.*, $\lvert\mathcal{C}\rvert = C$

c	1) Depending on the context, it may indicate the class label or the class index of a given class, $c \in \mathcal{C}$ 2) Size of the classifier, expressed in number of parameters, $c \in \mathbb{N}$ 3) Channel index
e_c	One-hot encoding vector of class c
$\delta_{cc'}$	Kronecker delta
Δ^C	C-dimensional probability simplex
$\hat{y}$	Output of the model f, *i.e.,* $f(x;\theta) = \hat{y}$ for some x, θ
$\mathcal{L}_{\text{CE}}$	Cross-entropy loss function
$\tilde{y}_c$	Logit predicted by f for class c
$\tau \in \mathbb{R}^+$	Softmax temperature
n	Number of samples in $\mathcal{D}$, *i.e.,* $n = \lvert\mathcal{D}\rvert = \sum_{k \in \mathcal{K}} n_k$
$\mathbb{1}$	Indicator function
$\mathcal{L}_{\text{SQ}}$	L2 loss
β	1) Least squares parameters bias, $\beta \in \mathbb{R}^C$ 2) `FDA` scaling hyper-parameter, $\beta \in (0, 1)$
$W^* \in \mathbb{R}^{p \times C}$	Optimal least squares or regularized least squares parameters
$X \in \mathbb{R}^{n \times p}$	Matrix of all the n stacked inputs
$Y \in \mathbb{R}^{n \times C}$	Matrix of all the n stacked one-hot encoded targets
λ	1) Tikhonov hyper-parameter, $\lambda \in \mathbb{R}^+$ 2) `FDA` hyper-parameter, $\lambda \in (0, 1]$ 3) `LADD` hyper-parameter, $\lambda \in \mathbb{R}^+$
$I_p \in \{0, 1\}^{p \times p}$	Identity matrix of dimension $p \times p$
b	1) RFF bias vector, $b \in \mathbb{R}^P$ 2) Sum of aggregated latent vectors per class, $b \in \mathbb{R}^{d \times C}$ 3) Size of the feature extractor, expressed in number of parameters, $b \in \mathbb{N}$ 4) batch of samples

$n_b \in \mathbb{N}$	Size of batch $b \in [B]$
$X_b \in \mathbb{R}^{n_b \times p}$	Matrix of all the stacked inputs in batch b
$Y_b \in \mathbb{R}^{n_b \times C}$	Matrix of all the stacked targets in batch b
$M_b \in \mathbb{R}^{p \times n_b}$	Recursive RR gain matrix
$P_b \in \mathbb{R}^{p \times p}$	Recursive RR covariance matrix
$0_{p \times C}$	$p \times C$ matrix of zeros
$\mathcal{X}'$	Generic higher-dimensional feature space
$\phi : \mathcal{X} \to \mathcal{X}'$	Mapping function
$\texttt{ker} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	Kernel function
F	RR objective
$\mathtt{K} \in \mathbb{R}^{n \times n}$	Kernel matrix
σ	1) RBF bandwidth hyper-parameter, $\sigma \in \mathbb{R}^+$ 2) Non-linear activation function, $\sigma : \mathbb{R} \to \mathbb{R}$
$P \in \mathbb{N}$	Random Fourier features dimensionality, $P > p$
$\mathcal{N}$	Normal distribution
$\mathcal{U}$	Uniform distribution
$\omega \in \mathbb{R}^{p \times P}$	Matrix of RFF normal vectors
$\phi(X)$	Matrix of stacked feature mappings $\phi(x)$
$\mathcal{C}_k \subseteq \mathcal{C}$	Set of all the classes of client $k \in \mathcal{K}$
$C_k \in [C]$	Number of classes of client $k \in \mathcal{K}$, *i.e.*, $\lvert\mathcal{C}_k\rvert = C_k$
$d \in \mathbb{N}$	Latent dimensionality
$\mathcal{Z} \subseteq \mathbb{R}^d$	Latent feature space
$\varphi : \mathcal{X} \to \mathcal{Z}$	Feature extractor function
$\psi : \mathcal{Z} \to \mathcal{Y}$	Classifier function
θ_φ	Feature extractor parameters
θ_ψ	Classifier parameters
$Z \in \mathbb{R}^{n \times d}$	Matrix of mapped inputs such that $Z_i = \varphi(X_i)$
A	Covariance matrix of mapped input samples in the latent space, $A \in \mathbb{R}^{d \times d}$ or $A \in \mathbb{R}^{D \times D}$
$Z_k \in \mathbb{R}^{n_k \times d}$	Defined as Z, but for the samples of $\mathcal{D}_k$
A_k	Defined as A, but for the samples of $\mathcal{D}_k$, $A_k \in \mathbb{R}^{d \times d}$ or $A_k \in \mathbb{R}^{D \times D}$
b_k	Defined as b, def. 2), but for the samples of $\mathcal{D}_k$, $b_k \in \mathbb{R}^{d \times C}$ or $b_k \in \mathbb{R}^{D \times C}$

$b_k^c \in \mathbb{R}^C$	Non-zero columns of b_k
A_t	Covariance matrix after aggregating the matrices A_k of t clients
b_t^c	b column (Def. 2)) after aggregating the vectors b_k^c of t clients
W_t	`Fed3R` solution computed using A_t and b_t
$D \in \mathbb{N}$	Latent dimensionality of `Fed3R-RF`, $D > d$
$\hat{Z}_k \in \mathbb{R}^{n \times D}$	Matrix of mapped inputs such that $\hat{Z}_k = \phi(\varphi(X_k))$
$Y_k \in \mathbb{R}^{n_k \times C}$	Matrix of all the n_k stacked one-hot encoded targets of client $k \in \mathcal{K}$
$J \in [K]$	Number of groups of clients
$\mathcal{K}_j \subseteq \mathcal{K}$	Group of clients
A_j	Securely aggregated covariance matrix for group of clients j, $A_j \in \mathbb{R}^{d \times d}$ or $A_j \in \mathbb{R}^{D \times D}$
b_j	Securely aggregated matrices b_k for group of clients j, $b_j \in \mathbb{R}^{d \times C}$ or $b_j \in \mathbb{R}^{D \times C}$
$\mathcal{D}_{\text{test}}$	Test dataset
$\bar{J} \in [0, 1]$	Mean Jaccard Index
α	1) Dirichlet parameter, $\alpha \in \mathbb{R}^+$ 2) BN momentum, $\alpha \in [0, 1]$
$m \in \mathbb{N}$	Size of the model, expressed in number of parameters
$\bar{C} \in [C]$	Average number of visible classes across all the clients
$\mathcal{T}$	Total average computational cost per round for a single client
$\mathcal{T}_t$	Cumulative average computational cost for a single client up to round t
F_*	Computation costs of one forward pass through $*$, where $*$ is one among f, φ, ψ
B_*	Computation costs of one backward pass through $*$, where $*$ is one among f, φ, ψ
$\delta \in [1, +\infty)$	Domain imbalance intensity

$\theta_k^* \in \Theta$	Optimal parameters for client $k \in \mathcal{K}$ according to some minimization problem
$f_k : \mathcal{X} \to \mathcal{Y}$	Personalized model function
$\mathcal{U}$	Shared parameters space
$u_k \in \mathcal{U}$	Shared parameters of client $k \in \mathcal{K}$
$\mathcal{V}$	Local parameters space
$v_k \in \mathcal{V}$	Local parameters of client $k \in \mathcal{K}$
u^t	Global shared parameters at round t
$u_k^t \in \mathcal{U}$	Shared parameters of client $k \in \mathcal{K}$ at round t
$v_k \in \mathcal{V}$	Local parameters of client $k \in \mathcal{K}$ at round t
$\mathcal{D}_k^{\text{test}}$	Test dataset of client $k \in \mathcal{K}$
$n_k^{\text{test}} \in \mathbb{N}$	Number of images in the test dataset of client $k \in \mathcal{K}$
$n^{\text{test}} \in \mathbb{N}$	Sum of the number of images in all the test datasets of all the clients
$\hat{y}_k$	Output of the model f_k, *i.e.,* $f_k(x; \theta_k) = \hat{y}$ for some x, θ_k
$\theta_{\psi,k} \in \mathbb{R}^{d \times C_k}$	Parameters of the `OLL` classifier of client $k \in \mathcal{K}$
$\bar{F}_{\hat{\psi}}$	Average costs of a forward pass through the `OLL` classifiers of all the clients $k \in \mathcal{K}$
$\mathbf{k}$	Feature map index
$x^{\mathbf{k}}$	Activation in the $\mathbf{k}$-th feature map
$w_{p,q,c}^{\mathbf{k}}$	Filter weights for channel c at the spatial positions p and q for the $\mathbf{k}$-th feature map
$b^{\mathbf{k}}$	Filter bias term for the $\mathbf{k}$-th feature map
$\mu_{\text{running}}^{\mathbf{k}}$	Running mean for the $\mathbf{k}$-th feature map
$\left(\sigma_{\text{running}}^{\mathbf{k}}\right)^2$	Running variance for the $\mathbf{k}$-th feature map
B	1) Batch of activations 2) Batch size, $B \in \mathbb{N}$
$m \in \mathbb{N}$	Batch size
$\mu_B^{\mathbf{k}}$	Mini-batch mean for the $\mathbf{k}$-th feature map
$\left(\sigma_B^{\mathbf{k}}\right)^2$	Mini-batch variance for the $\mathbf{k}$-th feature map
$\hat{x}^{\mathbf{k}}$	Normalized activation for the $\mathbf{k}$-th feature map

$\varepsilon \in \mathbb{R}$	Small BN constant for numerical stability
$\gamma^{\mathtt{k}}$	Learnable BN parameter for the $\mathtt{k}$-th feature map
$\beta^{\mathtt{k}}$	Learnable BN parameter for the $\mathtt{k}$-th feature map
$x^{\mathrm{S}} \in \mathbb{R}^{3\times W\times H}$	Source image
$x^{\mathrm{T}} \in \mathbb{R}^{3\times W\times H}$	Target image
$\mathcal{F}$	DFT function
j	Imaginary unit, *i.e.,* $j^2 = -1$
$\mathfrak{R}$	Real part
$\mathfrak{I}$	Imaginary part
$\mathcal{A}$	Function that extracts the amplitude of a complex number (or matrix of complex numbers)
Φ	Function that extracts the phase of a complex number (or matrix of complex numbers)
$M_\beta : \mathbb{R}^{W\times H} \rightarrow \mathbb{R}^{\lfloor \beta W \rfloor \times \lfloor \beta H \rfloor}$	Masking function
$\mathtt{A}^k$	Union of the amplitude spectrum of all the images of client $k \in \mathcal{K}$
$\mathtt{A}$	1) `CFSI` amplitudes bank 2) `LADD` styles bank
$\mathtt{L}^k$	Set of means and standard deviations of all the images of client $k \in \mathcal{K}$
$\mathtt{L}$	`LAB` means and standard deviations bank
μ^{S}	Mean of the source image in the `LAB` color space
σ^{S}	Standard deviation of the source image in the `LAB` color space
μ^{T}	Mean of the target image in the `LAB` color space
σ^{T}	Standard deviation of the target image in the `LAB` color space
$x^{\mathrm{S}}_{\mathrm{RGB}}$	Source image in the RGB space
$x^{\mathrm{S}}_{\mathrm{LAB}}$	Source image in the LAB space
$\hat{x}^{\mathrm{S}}_{\mathrm{LAB}}$	Source image in the LAB space with the LAB style of the target image
$\hat{x}^{\mathrm{S}}_{\mathrm{RGB}}$	Source image in the RGB space with the LAB style of the target image
$N \in \mathbb{N}$	Number of domains

$S \in \mathbb{N}^K$	List of desired number of samples for each client
$\mathcal{D}^c \subseteq \mathcal{D}$	Set of all $(x, y) \in \mathcal{D}$ such that class c appears in y
$c^* \in \mathcal{C}$	Class such that $\mathcal{D}^c$ is the smallest set among all $c \in \mathcal{C}$
$\mathcal{E}$	Random set of pairs from Algorithm 8
$\mathcal{D}_k^{\mathrm{T}}$	Target dataset of client $k \in \mathcal{K}$
$\mathcal{D}^{\mathrm{S}}$	Source dataset available at the server
$n_{\mathrm{S}} \in \mathbb{N}$	Number of samples of the source datasets, *i.e.,* $\lvert\mathcal{D}_{\mathrm{S}}\rvert = n_{\mathrm{S}}$
p^{S}	Source data distribution
a_k	Style of client $k \in \mathcal{K}$
θ^0	Final parameters after training on $\mathcal{D}^{\mathrm{S}}$
$\mathcal{S}$	Final `LADD` clustering
Γ	Function that, for any clustering $\mathcal{S}$ and client $k \in \mathcal{K}$, returns the cluster where k belongs to
$s \in \mathcal{S}$	Generic cluster
$\mathcal{R}$	Set of random seeds
$r \in \mathbb{N}$	Random seed
$o \in [K]$	`k-means` number of clusters
$\mathcal{S}_r^o$	`k-means` clustering with o clusters and random seed r
s_k	cluster of client $k \in \mathcal{K}$
δ_k	Function that computes the intra-cluster distance of client $k \in \mathcal{K}$ with its cluster
r^*	Best random seed
$\mathcal{S}^o$	Best `k-means` clustering with a given number of clusters o
Δ_k	Function that computes the inter-cluster distance of client $k \in \mathcal{K}$ with all the clusters except the one it belongs to
σ^o	Silhouette score for the best `k-means` clustering with o clusters
$o^* \in [K]$	Best number of `k-means` clusters
u	Global parameters

v_s	Cluster-specific parameters (of cluster $s \in \mathcal{S}$
S	Number of clusters of $\mathcal{S}$, *i.e.,* $\lvert\mathcal{S}\rvert = S$
$\hat{\theta}_s$	Teacher parameters of cluster $s \in \mathcal{S}$
u^0	Pre-trained global parameters
v^0	Pre-trained cluster-specific parameters
u^t	Global parameters at round t
v_s^t	Cluster-specific parameters (of cluster $s \in \mathcal{S}$ at round t
$\mathcal{L}_{\text{E}}$	Entropy loss
$\mathcal{L}_{\text{TS}}$	Teacher-student loss
$\mathcal{L}_{\text{PSEUDO}}$	Sum of $\mathcal{L}_{\text{E}}$ and $\mathcal{L}_{\text{TS}}$
$\mathcal{L}_{\text{KD}}$	Knowledge distillation loss
$\omega \in \mathbb{N}$	Teacher model update interval
$t_{\text{START}} \in [T]$	`LADD` hyper-parameter
τ^t	Fraction of the difference between the round t and t_{START}, and the teacher model update interval
μ_s	Centroid of cluster $s \in \mathcal{S}$
s^*	Nearest cluster for an image x
$\mathcal{B}$	Set of batches

References

[1] Caiming Zhang and Yang Lu. Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23:100224, 2021.

[2] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, 2(4), 2017.

[3] Arash Bahrammirzaee. A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, 19(8):1165–1195, 2010.

[4] Hongmei He, John Gray, Angelo Cangelosi, Qinggang Meng, T Martin McGinnity, and Jorn Mehnen. The challenges and opportunities of artificial intelligence for trustworthy robots and autonomous systems. In *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*, pages 68–74. IEEE, 2020.

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[6] KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[7] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349, 2018.

[8] Eyad Elyan, Pattaramon Vuttipittayamongkol, Pamela Johnston, Kyle Martin, Kyle McPherson, Carlos Francisco Moreno-García, Chrisina Jayne, and Md Mostafa Kamal Sarker. Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. *Artificial Intelligence Surgery*, 2(1):24–45, 2022.

[9] Nitin Kanagaraj, David Hicks, Ayush Goyal, Sanju Tiwari, and Ghanapriya Singh. Deep learning using computer vision in self driving cars for lane and traffic sign detection. *International Journal of System Assurance Engineering and Management*, 12(6):1011–1025, 2021.

[10] Zhonghe Ren, Fengzhou Fang, Ning Yan, and You Wu. State of the art in defect detection based on machine vision. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 9(2):661–691, 2022.

[11] Waqar Ali, Wenhong Tian, Salah Ud Din, Desire Iradukunda, and Abdullah Aman Khan. Classical and modern face recognition approaches: a complete review. *Multimedia tools and applications*, 80:4825–4880, 2021.

[12] European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance), May 2016.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[14] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[15] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[16] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[17] Donald Shenaj, Giulia Rizzoli, and Pietro Zanuttigh. Federated learning in computer vision. *Ieee Access*, 11:94863–94884, 2023.

[18] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[19] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in neural information processing systems*, 33:3557–3568, 2020.

[20] Alessandro Licciardi, Davide Leo, Eros Fanì, Barbara Caputo, and Marco Ciccone. Interaction-aware gaussian weighting for clustered federated learning. *Under review.*, 2025.

[21] Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Resource-efficient personalization in federated learning with closed-form classifiers. *IEEE Access*, 2025.

[22] Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Accelerating heterogeneous federated learning with closed-form classifiers. *International Conference on Machine Learning*, 2024.

[23] Mattia Dutto, Gabriele Berton, Debora Caldarola, Eros Fanì, Gabriele Trivigno, and Carlo Masone. Collaborative visual place recognition through federated learning. In *International FedVision Workshop in Conjunction with the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2024*, pages 4215–4225, 2024.

[24] Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Fed3r: Recursive ridge regression for federated learning with strong pre-trained models. *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with Neural Information Processing Systems (NeurIPS)*, 2023.

[25] Eros Fanì, Marco Ciccone, and Barbara Caputo. Feddrive v2: an analysis of the impact of label skewness in federated semantic segmentation for autonomous driving. In *5th Italian Conference on Robotics and Intelligent Machines (I-RIM)*, 2023.

[26] Donald Shenaj*, Eros Fanì*, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 444–454, 2023.

[27] Lidia Fantauzzo*, Eros Fanì*, Debora Caldarola, Antonio Tavera, Fabio Cermelli, Marco Ciccone, and Barbara Caputo. Feddrive: Generalizing federated learning to semantic segmentation in autonomous driving. In *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[28] Gergely D Németh, Eros Fanì, Yeat Ng Jeng, Barbara Caputo, Miguel A Lozano, Nuria M Oliver, and Novi Quadrianto. Feddiverse: Tackling data heterogeneity in federated learning with diversity-driven client selection. *Under review.*, 2025.

[29] Brendan McMahan and Daniel Ramage. Google ai blog: Federated learning: Collaborative machine learning without centralized training data. *URL https://ai. googleblog. com/2017/04/federated-learning-collaborative. html*, 2017.

[30] Pablo Villalobos and Anson Ho. Trends in training dataset sizes, 2022.

[31] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[32] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

[33] Edge Delta. Breaking down the numbers: How much data does the world create daily in 2024?, 2024.

[34] Mariusz Michalowski. How much data is generated every day in 2024?, 2024.

[35] Bernard Marr. How much data do we create every day? the mind-blowing stats everyone should read, 2021.

[36] David Reinsel, John Gantz, and John Rydning. The digitization of the world from edge to core, 2018.

[37] Western Digital. The ai data cycle, 2023.

[38] Mariusz Michalowski. 55 cloud computing statistics for 2024, 2024.

[39] Emily Bonnie and Anna Fitzgerald. 101 data privacy statistics: The facts you need to know in 2024, 2024.

[40] ISACA. Privacy in practice 2024 report, 2024.

[41] Meghan Rimol. Gartner identifies top five trends in privacy through 2024, 2022.

[42] Sufen Ren, Yule Hu, Shengchao Chen, and Guanjun Wang. Federated distillation for medical image classification: Towards trustworthy computer-aided diagnosis. *arXiv preprint arXiv:2407.02261*, 2024.

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

[44] Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019.

[45] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lizhen Cui. A survey on federated recommendation systems. *arXiv preprint arXiv:2301.00767*, 2022.

[46] Jiaxu Miao, Zongxin Yang, Leilei Fan, and Yi Yang. Fedseg: Class-heterogeneous federated learning for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8042–8052, 2023.

[47] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.

[48] Christos Anagnostopoulos, Alexandros Gkillas, Nikos Piperigkos, and Aris S Lalos. Personalized federated learning for cross-view geo-localization. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2024.

[49] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, 8(16):12806–12825, 2021.

[50] CBICA. The federated tumor segmentation (fets) initiative, 2024.

[51] Yahiaoui Mohammed Elbachir, Derdour Makhlouf, Gasmi Mohamed, Mohammed Mounir Bouhamed, and Kouzou Abdellah. Federated learning for multi-institutional on 3d brain tumor segmentation. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–8. IEEE, 2024.

[52] Filip Ślazyk, Przemysław Jabłecki, Aneta Lisowska, Maciej Malawski, and Szymon Płotka. Cxr-fl: deep learning-based chest x-ray image analysis using federated learning. In *International Conference on Computational Science*, pages 433–440. Springer, 2022.

[53] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pages 18–32. Springer, 2019.

[54] Fanglan Zheng, Kun Li, Jiang Tian, Xiaojia Xiang, et al. A vertical federated learning method for interpretable scorecard and its application in credit scoring. *arXiv preprint arXiv:2009.06218*, 2020.

[55] Bahar Farahani, Shima Tabibian, and Hamid Ebrahimi. Towards a personalized clustered federated learning: A speech recognition case study. *IEEE Internet of Things Journal*, 2023.

[56] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. End-to-end federated learning for autonomous driving vehicles. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[57] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[58] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated learning: privacy and incentive*, pages 240–254. Springer, 2020.

[59] Ahmed Imteaj and M Hadi Amini. Leveraging asynchronous federated learning to predict customers financial distress. *Intelligent Systems with Applications*, 14:200064, 2022.

[60] Yixing Liu, Bo Liu, Xiaoyu Guo, Yiqiao Xu, and Zhengtao Ding. Household profile identification for retailers based on personalized federated learning. *Energy*, 275:127431, 2023.

[61] Usman Ahmed, Gautam Srivastava, and Jerry Chun-Wei Lin. Reliable customer analysis using federated learning and exploring deep-attention edge intelligence. *Future Generation Computer Systems*, 127:70–79, 2022.

[62] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *International Conference on Learning Representations*, 2021.

[63] Tianyue Zheng, Ang Li, Zhe Chen, Hongbo Wang, and Jun Luo. Autofed: Heterogeneity-aware federated multimodal learning for robust autonomous driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[64] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13172–13179, 2020.

[65] Qiang Meng, Feng Zhou, Hainan Ren, Tianshu Feng, Guochao Liu, and Yuanqing Lin. Improving federated learning face recognition via privacy-agnostic clusters. *arXiv preprint arXiv:2201.12467*, 2022.

[66] Abraham Woubie, Enoch Solomon, and Joseph Attieh. Maintaining privacy in face recognition using federated learning method. *IEEE Access*, 2024.

[67] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):12598, 2020.

[68] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[69] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12):9587–9603, 2022.

[70] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[71] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[72] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[73] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[74] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 76–92. Springer, 2020.

[75] Debora Caldarola, Barbara Caputo, and Marco Ciccone. Improving generalization in federated learning by seeking flat minima. In *European Conference on Computer Vision*, pages 654–672. Springer, 2022.

[76] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2018.

[77] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

[78] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[79] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[80] Sunny Gupta, Vinay Sutar, Varunav Singh, and Amit Sethi. Fedalign: Federated domain generalization with cross-client feature alignment. *arXiv preprint arXiv:2501.15486*, 2025.

[81] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. *International Conference on Learning Representations (ICLR)*, 2021.

[82] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022.

[83] Di Wu, Jun Bai, Yiliao Song, Junjun Chen, Wei Zhou, Yong Xiang, and Atul Sajjanhar. Fedinverse: Evaluating privacy leakage in federated learning. In *The twelfth international conference on learning representations*, 2024.

[84] Li Bai, Haibo Hu, Qingqing Ye, Haoyang Li, Leixia Wang, and Jianliang Xu. Membership inference attacks and defenses in federated learning: A survey. *ACM Computing Surveys*, 57(4):1–35, 2024.

[85] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[86] David Byrd, Vaikkunth Mugunthan, Antigoni Polychroniadou, and Tucker Balch. Collusion resistant federated learning with oblivious distributed differential privacy. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 114–122, 2022.

[87] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR, 2020.

[88] Chunlu Chen, Ji Liu, Haowen Tan, Xingjian Li, Kevin I-Kai Wang, Peng Li, Kouichi Sakurai, and Dejing Dou. Trustworthy federated learning: privacy, security, and beyond. *Knowledge and Information Systems*, pages 1–36, 2024.

[89] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

[90] Yong Li, Yipeng Zhou, Alireza Jolfaei, Dongjin Yu, Gaochao Xu, and Xi Zheng. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet of Things Journal*, 8(8):6178–6186, 2020.

[91] John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? exploring the impact of pre-training and initialization in federated learning. *International Conference on Learning Representations*, 2023.

[92] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[93] Zexi Li, Xinyi Shang, Rui He, Tao Lin, and Chao Wu. No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5319–5329, 2023.

[94] Yilin Lyu, Liyuan Wang, Xingxing Zhang, Zicheng Sun, Hang Su, Jun Zhu, and Liping Jing. Overcoming recency bias of normalization statistics in continual learning: Balance and adaptation. *Advances in Neural Information Processing Systems*, 36, 2024.

[95] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[96] Yuan Liang, Yange Guo, Yanxia Gong, Chunjie Luo, Jianfeng Zhan, and Yunyou Huang. Flbench: A benchmark suite for federated learning. In *Intelligent Computing and Block Chain: First BenchCouncil International Federated Conferences, FICC 2020, Qingdao, China, October 30–November 3, 2020, Revised Selected Papers 1*, pages 166–176. Springer, 2021.

[97] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[98] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[99] Stephen M Stigler. Gauss and the invention of least squares. *the Annals of Statistics*, pages 465–474, 1981.

[100] Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[101] Ryan Rifkin, Gene Yeo, Tomaso Poggio, et al. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

[102] Christopher Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:531–537, 2006.

[103] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[104] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[105] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[106] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[107] Ruohan Wang, Marco Ciccone, Giulia Luise, Massimiliano Pontil, Andrew Yapp, and Carlo Ciliberto. Schedule-robust online continual learning. *arXiv preprint arXiv:2210.05561*, 2022.

[108] Raffaello Camoriano, Giulia Pasquale, Carlo Ciliberto, Lorenzo Natale, Lorenzo Rosasco, and Giorgio Metta. Incremental robot learning of new objects with fixed update time. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3207–3214. IEEE, 2017.

[109] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Prentice Hall, 2000.

[110] Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[111] William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

[112] M A Aizerman, E M Braverman, and L I Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[113] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery.

[114] John Shawe-Taylor and Nello Cristianini. Kernel methods for pattern analysis. *Cambridge University Press*, 2:181–201, 2004.

[115] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[116] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[117] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[118] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *Neurips Workshop on Federated Learning*, 2019.

[119] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *International Conference on Learning Representations*, 2019.

[120] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. Fedcm: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.

[121] Tailin Zhou, Jun Zhang, and Danny HK Tsang. Fedfa: Federated learning with feature anchors to align features and classifiers for heterogeneous data. *IEEE Transactions on Mobile Computing*, 2023.

[122] Yaodong Yu, Alexander Wei, Sai Praneeth Karimireddy, Yi Ma, and Michael Jordan. Tct: Convexifying federated learning using bootstrapped neural tangent kernels. *Advances in Neural Information Processing Systems*, 35:30882–30897, 2022.

[123] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *Eighth International Conference on Learning Representations (ICLR)*, 2020.

[124] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[125] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[126] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott 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*, pages 3589–3599, 2021.

[127] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[128] Gwen Legate, Lucas Caccia, and Eugene Belilovsky. Re-weighted softmax cross-entropy to control forgetting in federated learning. In *Conference on Lifelong Learning Agents*, pages 764–780. PMLR, 2023.

[129] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3403–3411. PMLR, 2021.

[130] Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. *arXiv preprint arXiv:2204.13399*, 2022.

[131] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[132] Gwen Legate, Nicolas Bernier, Lucas Caccia, Edouard Oyallon, and Eugene Belilovsky. Guiding the last layer in federated learning with pre-trained models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[133] Ali H Sayed. *Adaptive Filters*. Wiley-IEEE Press, 2008.

[134] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8432–8440, 2022.

[135] Jianqing Zhang, Yang Liu, Yang Hua, and Jian Cao. Fedtgp: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 16768–16776, 2024.

[136] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[137] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3347–3366, 2021.

[138] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[139] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[140] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380, 2022.

[141] Haokun Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4), 2021.

[142] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. Smpai: Secure multi-party computation for federated learning. In *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*, volume 21. MIT Press Cambridge, MA, USA, 2019.

[143] Andrei Afonin and Sai Praneeth Karimireddy. Towards model agnostic federated learning using knowledge distillation. In *International Conference on Learning Representations*, 2022.

[144] Jianping Cai, Ximeng Liu, Zhiyong Yu, Kun Guo, and Jiayin Li. Efficient vertical federated learning method for ridge regression of large-scale samples. *IEEE Transactions on Emerging Topics in Computing*, 2022.

[145] Lingxiao Huang, Zhize Li, Jialin Sun, and Haoyu Zhao. Coresets for vertical federated learning: Regularized linear regression and k-means clustering. *Advances in Neural Information Processing Systems*, 35:29566–29581, 2022.

[146] Celeste Damiani, Yulia Rodina, and Sergio Decherchi. A hybrid federated kernel regularized least squares algorithm. *Knowledge-Based Systems*, 305:112600, 2024.

[147] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. *Advances in neural information processing systems*, 25, 2012.

[148] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.

[149] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

[150] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2575–2584, 2020.

[151] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.

[152] David Solans, Mikko Heikkila, Andrea Vitaletti, Nicolas Kourtellis, Aris Anagnostopoulos, Ioannis Chatzigiannakis, et al. Non-iid data in federated learning: A systematic review with taxonomy, metrics, methods, frameworks and future directions. *arXiv preprint arXiv:2411.12377*, 2024.

[153] Guixun Luo, Naiyue Chen, Jiahuan He, Bingwei Jin, Zhiyuan Zhang, and Yidong Li. Privacy-preserving clustering federated learning for non-iid data. *Future Generation Computer Systems*, 154:384–395, 2024.

[154] Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, 22(4):866–882, 1990.

[155] Marco Ferrante and Monica Saltalamacchia. The coupon collector's problem. *Materials matemàtics*, pages 0001–35, 2014.

[156] Marco Ferrante and Nadia Frigo. A note on the coupon-collector's problem with multiple arrivals and the random sampling. *arXiv preprint arXiv:1209.2667*, 2012.

[157] Yunlu Yan and Lei Zhu. A simple data augmentation for feature distribution skewed federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[158] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[159] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.

[160] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. *Advances in Neural Information Processing Systems*, 32, 2019.

[161] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[162] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[163] Guangjing Huang, Xu Chen, Tao Ouyang, Qian Ma, Lin Chen, and Junshan Zhang. Collaboration in participant-centric federated learning: A game-theoretical perspective. *IEEE Transactions on Mobile Computing*, 22(11):6311–6326, 2022.

[164] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 228–237. IEEE, 2021.

[165] Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodolà, and Barbara Caputo. Cluster-driven graph federated learning over multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2749–2758, 2021.

[166] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in neural information processing systems*, 33:21394–21405, 2020.

[167] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International conference on machine learning*, pages 6357–6368. PMLR, 2021.

[168] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[169] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, and Chao Wu. Federated mutual learning. *arXiv preprint arXiv:2006.16765*, 2020.

[170] Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pages 17716–17758. PMLR, 2022.

[171] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[172] Filip Hanzely, Boxin Zhao, and Mladen Kolar. Personalized federated learning: A unified framework and universal optimization techniques. *arXiv preprint arXiv:2102.09743*, 2021.

[173] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[174] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems*, 34:11220–11232, 2021.

[175] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[176] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR, 2021.

[177] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[178] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[179] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[180] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[181] Gabriel L Oliveira, Wolfram Burgard, and Thomas Brox. Efficient deep models for monocular road segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4885–4891. IEEE, 2016.

[182] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[183] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[184] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

[185] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in neural information processing systems*, 34:17864–17875, 2021.

[186] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021.

[187] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[188] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, 34:9355–9366, 2021.

[189] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[190] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[191] Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*, pages 92–104. Springer, 2019.

[192] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M Jorge Cardoso, et al. Privacy-preserving federated brain tumour segmentation. In *Machine Learning in Medical Imaging: 10th International Workshop, MLMI 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13, 2019, Proceedings 10*, pages 133–141. Springer, 2019.

[193] Liping Yi, Jinsong Zhang, Rui Zhang, Jiaqi Shi, Gang Wang, and Xiaoguang Liu. Su-net: an efficient encoder-decoder model of federated learning for brain tumor segmentation. In *International Conference on Artificial Neural Networks*, pages 761–773. Springer, 2020.

[194] Cosmin I Bercea, Benedikt Wiestler, Daniel Rueckert, and Shadi Albarqouni. Feddis: Disentangled federated learning for unsupervised brain pathology segmentation. *arXiv preprint arXiv:2103.03705*, 2021.

[195] Umberto Michieli and Mete Ozay. Prototype guided federated learning of visual feature representations. *arXiv preprint arXiv:2105.08982*, 2021.

[196] Marco Toldo, Umberto Michieli, Gianluca Agresti, and Pietro Zanuttigh. Unsupervised domain adaptation for mobile semantic segmentation based on cycle consistency and feature alignment. *Image and Vision Computing*, 95:103889, 2020.

[197] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Unsupervised domain adaptation for semantic image segmentation: a comprehensive survey. *arXiv preprint arXiv:2112.03241*, 2021.

[198] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 102–118. Springer, 2016.

[199] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.

[200] Emanuele Alberti, Antonio Tavera, Carlo Masone, and Barbara Caputo. Idda: A large-scale multi-domain dataset for autonomous driving. *IEEE Robotics and Automation Letters*, 5(4):5526–5533, 2020.

[201] Paolo Testolina, Francesco Barbato, Umberto Michieli, Marco Giordani, Pietro Zanuttigh, and Michele Zorzi. Selma: Semantic large-scale multimodal acquisitions in variable weather, daytime and viewpoints. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):7012–7024, 2023.

[202] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[203] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4544–4553, 2020.

[204] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[205] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[206] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[207] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018.

[208] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7472–7481, 2018.

[209] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2507–2516, 2019.

[210] Umberto Michieli, Matteo Biasetton, Gianluca Agresti, and Pietro Zanuttigh. Adversarial learning and self-teaching techniques for domain adaptation in semantic segmentation. *IEEE Transactions on Intelligent Vehicles*, 5(3):508–518, 2020.

[211] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[212] Fabio Pizzati, Raoul de Charette, Michela Zaccaria, and Pietro Cerri. Domain bridge for unpaired image-to-image translation and unsupervised domain adaptation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2990–2998, 2020.

[213] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4085–4095, 2020.

[214] Qing Lian, Fengmao Lv, Lixin Duan, and Boqing Gong. Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6758–6767, 2019.

[215] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

[216] Francesco Barbato, Marco Toldo, Umberto Michieli, and Pietro Zanuttigh. Latent space regularization for unsupervised domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2835–2845, 2021.

[217] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9924–9935, 2022.

[218] Weiming Zhuang, Xin Gan, Xuesen Zhang, Yonggang Wen, Shuai Zhang, and Shuai Yi. Federated unsupervised domain adaptation for face recognition. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022.

[219] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[220] Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. Federated multi-target domain adaptation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1424–1433, 2022.

[221] Nan Lu, Zhao Wang, Xiaoxiao Li, Gang Niu, Qi Dou, and Masashi Sugiyama. Federated learning from only unlabeled data with class-conditional-sharing clients. *arXiv preprint arXiv:2204.03304*, 2022.

[222] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[223] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24, 2011.

[224] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[225] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125*, 2017.

[226] Quande Liu, Qi Dou, and Pheng-Ann Heng. Shape-aware meta-learning for generalizing prostate mri segmentation to unseen domains. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*, pages 475–485. Springer, 2020.

[227] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5715–5725, 2017.

[228] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018.

[229] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[230] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1013–1023, 2021.

[231] C Xing Tian, Haoliang Li, Yufei Wang, and Shiqi Wang. Privacy-preserving constrained domain generalization for medical image classification. *arXiv preprint arXiv:2105.08511*, 4, 2021.

[232] Commission Internationale de l'Éclairage (CIE). *Colorimetry—CIE 15:2004*. CIE, Vienna, Austria, 3rd edition, 2004.

[233] Jianzhong He, Xu Jia, Shuaijun Chen, and Jianzhuang Liu. Multi-source domain adaptation with collaborative learning for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11008–11017, 2021.

[234] Amirhossein Reisizadeh, Farzan Farnia, Ramtin Pedarsani, and Ali Jadbabaie. Robust federated learning: The case of affine distribution shifts. *Advances in neural information processing systems*, 33:21554–21565, 2020.

[235] Mathieu Andreux, Jean Ogier du Terrail, Constance Beguier, and Eric W Tramel. Siloed federated learning for multi-centric histopathology datasets. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 2*, pages 129–139. Springer, 2020.

[236] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[237] John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard Turner. Tasknorm: Rethinking batch normalization for meta-learning. In *International Conference on Machine Learning*, pages 1153–1164. PMLR, 2020.

[238] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[239] Leon N Piotrowski and Fergus W Campbell. A demonstration of the visual importance and flexibility of spatial-frequency amplitude and phase. *Perception*, 11(3):337–346, 1982.

[240] Nathalie Guyader, Alan Chauvin, Carole Peyrin, Jeanny Hérault, and Christian Marendaz. Image phase or amplitude? rapid scene categorization is an amplitude-based process. *Comptes Rendus Biologies*, 327(4):313–318, 2004.

[241] Matteo Frigo and Steven G Johnson. Fftw: An adaptive software architecture for the fft. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 3, pages 1381–1384. IEEE, 1998.

[242] Shu Jiang, Yu Wang, Weiman Lin, Yu Cao, Longtao Lin, Jinghao Miao, and Qi Luo. A high-accuracy framework for vehicle dynamic modeling in autonomous driving. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6680–6687. IEEE, 2021.

[243] Kosmas Tsiakas, Ioannis Kostavelis, Antonios Gasteratos, and Dimitrios Tzovaras. Autonomous vehicle navigation in semi-structured environments based on sparse waypoints and lidar road-tracking. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1244–1250. IEEE, 2021.

[244] Timothy Ha, Gunmin Lee, Dohyeong Kim, and Songhwai Oh. Road graphical neural networks for autonomous roundabout driving. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 162–167. IEEE, 2021.

[245] Viswadeep Lebakula, Bo Tang, Christopher Goodin, and Cindy L Bethel. Shape estimation of negative obstacles for autonomous navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4525–4531. IEEE, 2021.

[246] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, and Barbara Caputo. Detecting anomalies in semantic segmentation with prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–121, 2021.

[247] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. Deep federated learning for autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1824–1830. IEEE, 2022.

[248] Yijing Li, Xiaofeng Tao, Xuefei Zhang, Junjie Liu, and Jin Xu. Privacy-preserved federated learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8423–8434, 2021.

[249] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International journal of computer vision*, 129:3051–3068, 2021.

[250] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[251] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[252] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[253] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[254] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[255] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[256] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *Proceedings of the IEEE international conference on computer vision*, pages 1992–2001, 2017.

[257] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.

[258] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[259] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[260] Ravpreet Kaur and Sarbjeet Singh. A comprehensive review of object detection with deep learning. *Digital Signal Processing*, 132:103812, 2023.

[261] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 56(1):1–37, 2023.

[262] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9404–9413, 2019.

[263] Shiv Ram Dubey. A decade survey of content based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2687–2704, 2021.