Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (36th cycle)

Advancing Generalization in Heterogeneous Federated Learning for Real-world Vision Applications

By

Debora Caldarola

******

Supervisors:
Prof. Barbara Caputo
PhD Marco Ciccone

Doctoral Examination Committee (in alphabetical order):
Prof. Sophie Fosson, Politecnico di Torino, *President*
Prof. Samuel Horvath, Mohamed bin Zayed University of Artificial Intelligence
Prof. Martin Jaggi, École Polytechnique Fédérale de Lausanne, *Referee*
Prof. Sanmi Koyejo, Stanford University
Prof. Nicholas Lane, University of Cambridge, *Referee*

Politecnico di Torino
2024

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.

Debora Caldarola
2024

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

Alla mia famiglia
Mamma, Papà, Ale e Gianluigi

Acknowledgements

I would like to begin by expressing my gratitude to the professors, colleagues, collaborators and friends who have supported, guided, and believed in me throughout this journey. This PhD would have not been possible without you.

My heartfelt thanks go first to my supervisors, Barbara and Marco, to whom I am profoundly grateful. From my Master's thesis to planning my future path, Barbara has consistently shown her support while allowing me to make my own decisions. Thank you for choosing to bet on me time and time again. Marco has been both my mentor and my brother-in-arms. You taught me the true meaning of valid research and the values of a dedicated researcher, while always being there for the highs and the lows. Thank you for believing in me, for the successes we achieved together, for your guidance, and for sharing this path with me.

I am honored to thank Prof. Fosson, Prof. Horvath, Prof. Jaggi, Prof. Koyejo and Prof. Lane for agreeing to serve on my Examination Committee. I am grateful for the time you have dedicated to me and for the opportunity to present my work to such a distinguished group. I owe a special thanks to Prof. Jaggi and Prof. Lane for the effort they have put into reviewing this thesis. Your constructive feedback has enriched my work, and I deeply appreciated your support.

I would like to thank Sanmi for welcoming me into his laboratory at Stanford and making me feel like a member of the team from day one. It was an enriching experience, from the insightful discussions on campus to the memorable team-building moments.

These years would not have been the same without my VANDAL lab mates. Our time together was filled with laughter, shared adventures and misadventures, Spritzes, and great food. You made this experience truly unforgettable. A special recognition goes to those who have been there since day one - Alli, Chiara, Fabio, Silvia, Tav, Francesco, Dario, Mirco, Nick, all our Gabriele and Luca - and to my federated learning team - Eros and Riccardo, whom I am happy to call friends.

My strength, my constant source of support, and my anchor have always been my family. Mum and Dad, thank you for always being there for me and making the distance feel nonexistent, and for showing us that love does conquer all, even in the face of adversity. Throughout this journey, Alessandra has been my greatest supporter, from helping me wake up every morning and enjoy delicious meals to showing me how deeply she believes in me. Our relationship is our greatest strength.

Speaking of love, patience and immense support, meeting (Dr.) Gianluigi has been the greatest success of my PhD. This is just one of the many life achievements we will celebrate together.

Lastly, I would like to express my gratitude to those who took a chance on me and provided me with a wonderful opportunity for my future.

Abstract

Federated Learning (FL) is a machine learning framework that enables collaborative training of a global model across edge devices (*clients*) without compromising user privacy. Unlike traditional centralized approaches that require transferring raw data to a central server, FL operates by exchanging model parameters and performing training locally. Given that most data today originates from edge devices like smartphones and Internet of Things hardware, FL offers a path to leverage this vast and sensible resource while remaining compliant with privacy regulations.

This dissertation addresses key challenges in deploying FL in real-world scenarios, where personal habits introduce inherent bias in the local data distributions and users' devices vary widely in computational resources and network reliability. Within this context, optimizing communication efficiency and mitigating the effects of non-uniform data distributions are critical. This thesis seeks to uncover underlying causes of poor model generalization and develop an approach that generalizes effectively to the overall data distribution, emphasizing vision-oriented applications.

The core contribution of this work lies in leveraging the geometry of the loss landscape to understand and mitigate the behavior of models trained in heterogeneous federated scenarios. Indeed, this thesis shows FL models usually end up in sharp minima, providing a plausible explanation to their lack of generalization. By promoting convergence towards globally flat minima with sharpness-aware strategies, the global model's robustness to distribution shifts is enhanced.

Additionally, to accelerate the speed of convergence and reduce the communication cost, this work proposes a new training framework that facilitates exchanges between groups of *dissimilar* clients. This approach offers a winning alternative to both the conventional client-server architecture and existing methods that group clients based on similarity to learn cluster-specific models.

Finally, with most future data expected from vision-based edge systems, computer vision tasks are notably underrepresented in FL research, partly due to a lack of benchmarks and large-scale federated datasets. To address this gap, this thesis introduces three novel vision benchmarks for FL, focusing on semantic segmentation for autonomous driving and collaborative visual place recognition.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\mathscr{C}$	Clients set
$\mathscr{D}$	Dataset
$\mathscr{N}$	Negative images set
$\mathscr{P}$	Positive images set
$\mathscr{X}$	Input space
$\mathscr{Y}$	Output space
B	Batch size
C	Number of classes
E	Number of local epochs
f	Objective function
K	Number of clients
N	Number of images
T	Number of rounds
$\boldsymbol{w}$	Model parameters

Greek Symbols

α	Dirichlet distribution concentration parameter

ϵ	SAM perturbation
β	Momentum parameter
η	Learning rate
λ	Hessian eigenvalue
λ_1	Maximum Hessian eigenvalue
$\nabla_{\boldsymbol{w}}$	Gradient of model parameters $\boldsymbol{w}$
$\Delta_{\boldsymbol{w}}$	Pseudo-gradient of model parameters $\boldsymbol{w}$

Superscripts

q	Reference to query image q
t	Round index

Subscripts

g	Global, *i.e.*, server side
k	Client index
l	Local, *i.e.*, client side

Other Symbols

$[N]$	Indexing from 1 to N
$\|\|\cdot\|\|$	ℓ_2 norm
$\|\cdot\|$	Set cardinality

Acronyms / Abbreviations

non-*i.i.d.*	Non-independent and identically distributed
vs.	Versus
w.r.t.	With respect to
AI	Artificial Intelligence

BN	Batch Normalization
CE	Cross-entropy
CFL	Centralized Federated Learning
DFL	Decentralized Federated Learning
DL	Deep Learning
FL	Federated Learning
GN	Group Normalization
IoT	Internet of Things
KD	Knowledge Distillation
ML	Machine Learning
NN	Neural Network
SAM	Sharpness-aware Minimization
SGD	Stochastic Gradient Descent
SOTA	State-of-the-art
VPR	Visual Place Recognition

Chapter 1

Introduction

It is change, continuing change, inevitable change, that
is the dominant factor in society today.
No sensible decision can be made any longer without
taking into account not only the world as it is,
but the world as it will be.

ISAAC ASIMOV

This opening chapter establishes the framework and motivations for the research presented in this thesis. Section 1.1 explores current challenges in the world of Artificial Intelligence, introducing the specific setting of interest for this work, Federated Learning. The research questions and the thesis contributions are summarized in Section 1.2. Section 1.3 provides a roadmap for navigating the manuscript, while Section 1.4 offers details on the individual research papers included.

1.1 Context and Motivation

Artificial Intelligence (AI) has permeated nearly every facet of people's daily lives, shaping experiences from personalized movie recommendations [9] and voice interactions with smartphones [10] to advancements in autonomous driving [11]. Modern AI models can now understand and generate text [12–15], comprehend and respond to human speech [16], analyze video and image content [17–20], generate their own visual data [21], and process both text and images simultaneously [22]. As illustrated in Figure 1.1, interest in AI systems has surged in recent years, particularly within the domains of vision and text after the rise of foundation models [23]. The 2024 Nobel Prizes recognized the transformative impact of AI on modern society, awarding the Physics Prize to Geoffrey E. Hinton and John J. Hopfield, pioneers of modern AI [24], and the Chemistry Prize to Demis Hassabis and John Jumper for their use of AlphaFold2 [25] "*for protein structure prediction*" [26].

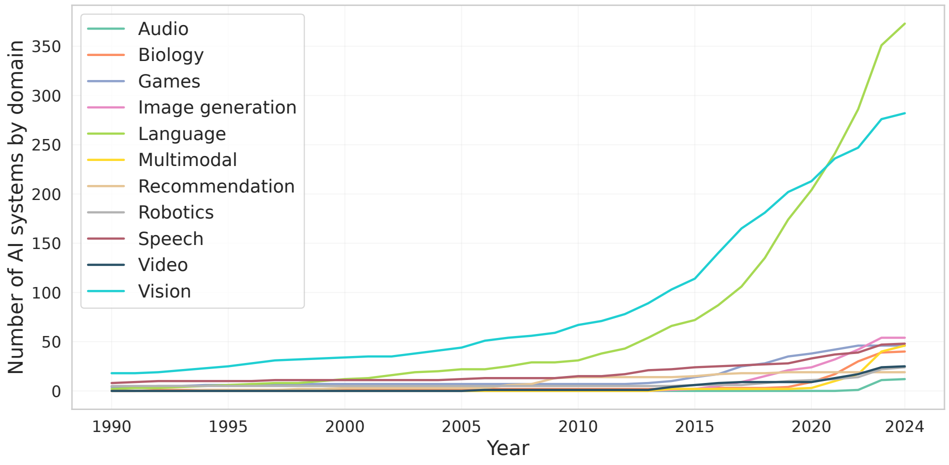


Fig. 1.1 Notable AI systems by domain over the years. This plot illustrates the rapid advancements and increasing interest in AI across various fields, with a particular emphasis on the significant growth in the *vision* and language domains. Data source: [1].

The first major factor impacting AI model performance is the quantity of data available [27], as these models are inherently *data-hungry* [28]. As Figure 1.2 shows, the size of training datasets has grown exponentially over the years, with recent models like GPT-4 [29] and Llama 3.1 [30] requiring over 10^{12} data points to achieve state-of-the-art performance. Large datasets are crucial for enabling models to *generalize* effectively, meaning they can maintain high accuracy when predicting outcomes for previously unseen data [31, 32].

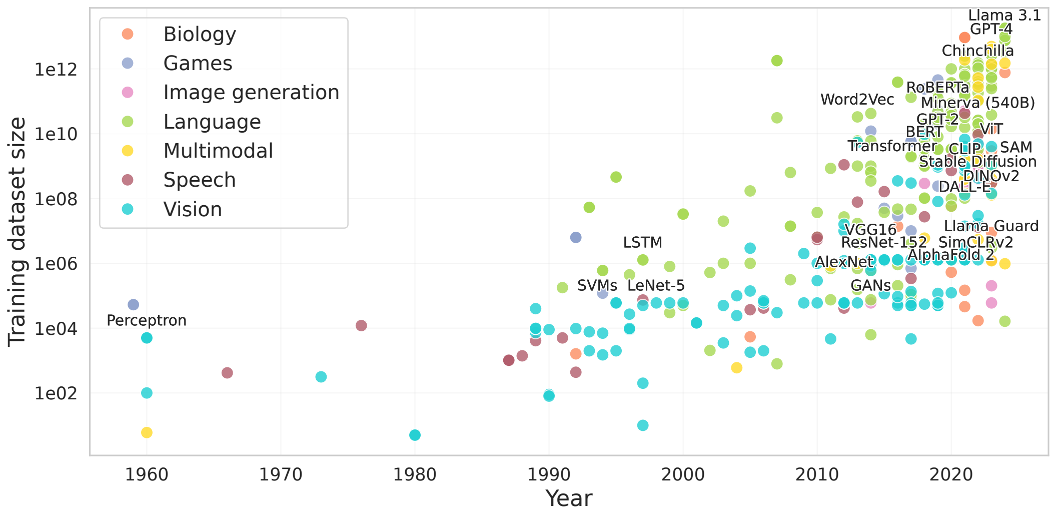


Fig. 1.2 Data points used to train notable AI systems: the dataset size exponentially increases as modern deep learning models become more complex. Data from [2].

Today, **a substantial amount of data is generated at the edge**, collected from devices such as smartphones, IoT sensors, and cameras. The International Data Corporation estimates that by 2025 there will be 55.9 billion connected devices producing 79.4ZB of data, predominantly belonging to the world of IoT [33]. Notably, **a significant portion of this data will come from video-based devices**, such as surveillance cameras. These large volumes of of data data are crucial for training AI models designed for real-world applications, where variability is high. The quality of the training data directly influences the model's ability to detect complex patterns, improve accuracy, and enhance generalization—critical factors for successfully addressing the challenges posed by dynamic, real-world environments. Data generated at the edge possesses two main characteristics: it is often heavily **influenced by user habits and preferences**, making it highly personalized, and it typically **contains sensitive information that raises privacy concerns**. These factors present significant challenges for training robust AI models, as much of this edge data cannot be accessed using conventional machine learning training methods, nor can it be easily collected or stored in off-premise locations. This necessitates the development of specialized techniques that respect user privacy while still allowing for effective decentralized model training.

Another significant constraint in training AI models is the **demand for computational resources**, which has been growing at an annual rate of approximately 4.2

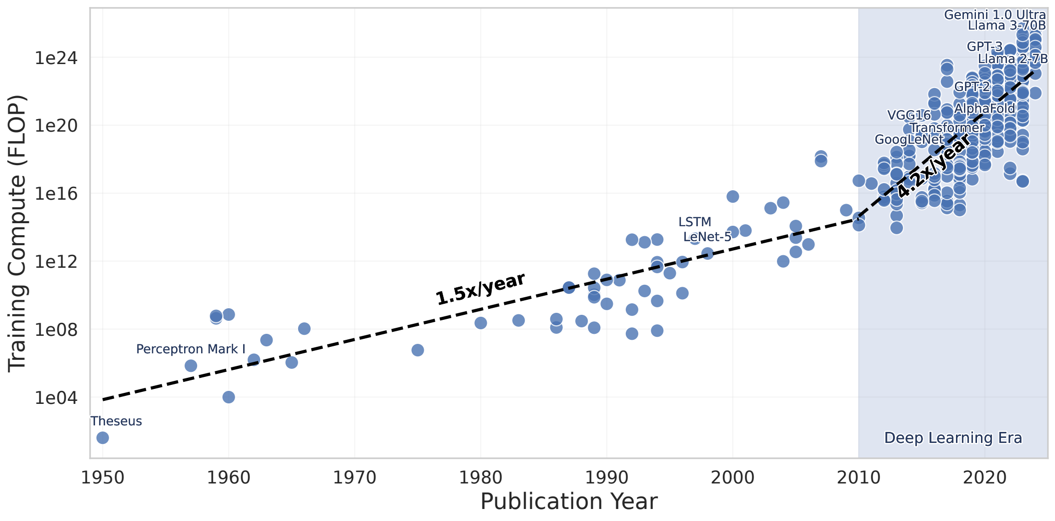


Fig. 1.3 Training computation, measured in FLOPs, required to train AI models over time. Prior to the deep learning era (*highlighted in blue*), the annual growth rate was approximately 1.5×. Starting in 2010, this rate accelerated to 4.2× per year. Data source: [3].

times since the beginning of the "deep learning era" (Figure 1.3). This need often translates into substantial hardware and energy costs and significant environmental impact. As shown in Figure 1.4, training advanced models like GPT-4 can exceed 10 million U.S. dollars, reflecting the extensive infrastructure and power required for such complex computations. In contrast, edge devices offer a unique advantage, as they possess their own computational resources that can be leveraged to conduct training operations. This approach can significantly reduce the need for additional infrastructure and mitigate overall costs associated with training sophisticated AI models. However, these devices may not possess the necessary resources to complete the entire training process independently and are further subject to factors such as battery life and network connectivity. These limitations necessitate a more efficient approach that leverages edge device capabilities while mitigating their constraints.

Federated Learning (FL) emerges as a promising solution to enable the collaborative training of AI models on edge devices (*clients*), overcoming their individual limitations. This approach facilitates collaborative learning across a network of devices without the need to share raw data directly [34]. Each client trains a local model on its own private data and then transmits only the model updates to a central server for aggregation. This process enables the global model to **benefit from the collective knowledge of all participants** and **ensures individual data privacy** by eliminating the need for transmitting large volumes of data to centralized

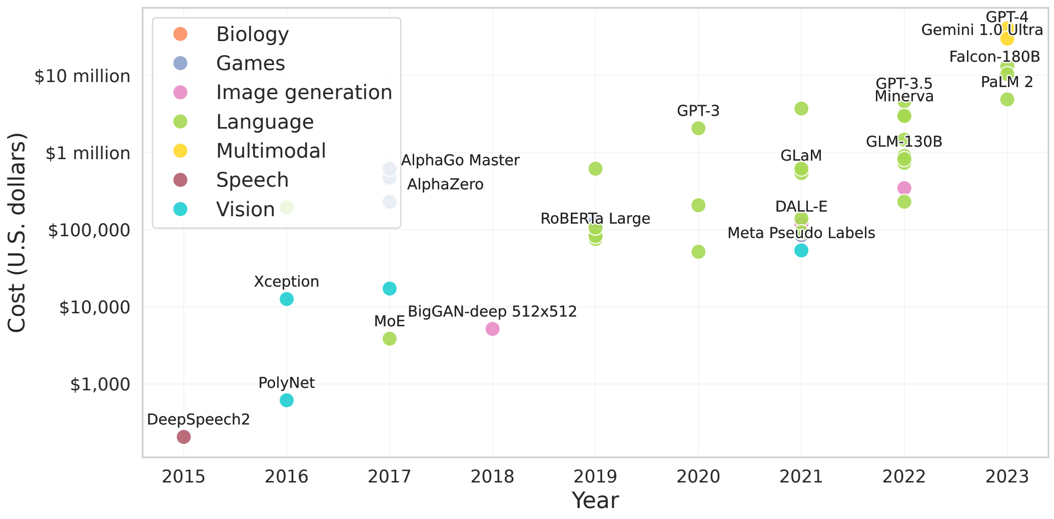


Fig. 1.4 Hardware and energy cost to train notable AI systems, expressed in US dollars. The cost to train modern models surpasses $10 million dollars. Data source: [4].

servers for processing. Privacy in FL is achieved by adhering to the *Principle of Data Minimization* [35–37]: only the data necessary for a specific computation is collected, *i.e.*, the model parameters (*focused collection*); the user's information is processed immediately upon receipt (*early aggregation*) and then discarded (*minimal retention*). In addition, users are informed and give their *consent* regarding the use of their data, ensuring *transparency*. Lastly, FL aims to uphold the *Principle of Data Anonymization*, where the model's final output does not disclose any sensitive information and aggregate statistics (*e.g.*, model parameters) do not significantly vary based on any individual user's data [38].

As multiple devices collaborate in the training process, **the effort to train large models is distributed**, making it more feasible [39, 40]. Since offline nodes can be replaced with available ones, training can proceed with minimal disruption, effectively **minimizing the limitations associated with standalone edge systems**. Thus, the key appeal of FL lies in its ability to efficiently learn from privacy-protected, distributed data while complying with regulations and leveraging edge resources.

Numerous real-world applications of FL have already demonstrated their success. For example, Google Keyboard leverages FL to enhance user experience by learning individual typing patterns - such as commonly used words - while safeguarding privacy by not storing private messages on centralized servers [41, 6]. Similarly, the voice assistants "Hey Google" and "Siri" use FL to recognize users' voices without uploading audio data to the cloud, while simultaneously developing a new speech model that incorporates learning from all participating devices [42, 43].

Since 2017, the research community FL has concentrated on addressing several core challenges that define and complicate this field [44]. To clarify them, consider the following example. Imagine a teenager in Europe using their smartphone, often capturing photos of their small dog on walks through historic city centers with ancient architecture. These photos vary in lighting, from bright daytime images to dimly lit evening scenes, and in location, featuring both indoor and outdoor settings. In contrast, imagine an adult in the United States, who also uses their smartphone but primarily takes photos of landscapes, often in suburban or rural areas, and typically in broad daylight. This difference extends to their language in text messages as well. The European teenager might use slang, emojis, and informal abbreviations frequently in their messages, creating a casual, playful tone. Meanwhile, the American adult might favor more standard language and less emoji use, producing a more formal or straightforward style. These variations in image content and linguistic style illustrate the challenge of *data heterogeneity* in FL, where different user behaviors and contexts lead to inconsistencies in data distribution, making it difficult to train a unified model effectively [44, 6]. *Communication constraints* add further complexity. In the previous scenario, smartphones may lose connectivity due to weak signals or depleted battery life, negatively impacting efficient exchange of model updates. In general, communication between devices and the central server is often limited by bandwidth restrictions or connectivity issues [6]. *Privacy concerns* also persist within the FL framework. Although FL reduces the need to share raw data, the risk of information leakage through model updates remains significant, necessitating the development of privacy-preserving techniques to protect sensitive data [38]. Finally, the availability of *computational resources* on individual devices can vary widely, introducing system heterogeneity that challenges the creation of efficient and fair learning processes across devices [45]. Together, these issues shape the direction of contemporary FL research, driving efforts to improve robustness, efficiency, and privacy in decentralized learning systems.

Lastly, while FL has shown promise in various domains, research on its application to computer vision tasks remains an emerging field [46]. As mentioned above, in the coming years, a substantial amount of sensible data is expected to be generated by video-based edge devices, highlighting **the need for ad-hoc FL algorithms for training robust computer vision models** to extract value from this data.

This thesis highlights the existence of a significant gap between research on FL and its deployment in real-world complex applications, particularly in the vision domain. By investigating the underlying issues behind the lack of model generalization and proposing novel solutions, this work contributes to the development of robust and generalizable FL models that can unlock the full potential of this technology in complex real-world scenarios.

1.2 Research Questions and Contributions

This thesis addresses the critical challenge of generalization in heterogeneous FL scenarios, specifically for real-world computer vision applications.

The term "real-world" in federated contexts refers to *heterogeneous* settings, where clients have diverse data distributions and computational capabilities, limited network access (*e.g.*, unreliable or bandwidth-constrained), and consequently, inconsistent availability (for instance, due to battery life). All those factors constitute a relevant challenge to the deployment of FL.

Aiming to learn a global model capable of generalizing to the underlying overall distribution under the aforementioned constraints, this thesis highlights and addresses the following gaps in current research:

Generalization through the loss landscape: recent research trends have identified a link between models' generalization ability and the geometry of their convergence point in the loss landscape, associating poor generalization with sharp minima. However, no prior work has examined the relationship between the geometry of the loss landscape and limited generalization in heterogeneous FL. This thesis explores the hypothesis that convergence to sharp minima may contribute to the poor generalization typical of FL models in heterogeneous settings, and shows that discrepancies between local and global loss surfaces are influenced by data heterogeneity. By encouraging convergence toward flat minima in the global loss landscape, substantial improvement in generalization performance can be achieved in several tasks (*e.g.*, large-scale image classification, semantic segmentation, domain generalization), while maintaining communication efficiency. This work not only leads to better performance across diverse data distributions in heterogeneous FL but also opens new avenues for understanding the dynamics of model performance in complex learning environments.

Training paradigm: this thesis proposes novel training orchestration paradigms that leverage privacy-preserving similarity metrics to facilitate the grouping of similar or dissimilar clients, optimizing collaboration in FL. Differently from previous existing research that often utilizes clustering techniques, this work proposes a new training framework that leverages groups of *dissimilar* clients to reduce communication exchanges while mitigating the models' destructive interference typical of heterogeneous FL environments, resulting in improved model quality and convergence speedup.

Graph-based layer parameters: the standard FL approach is based on learning a global model that *on average* minimizes all local empirical risks across the entire population. In contrast, personalized FL approaches often learn model parameters specific to each client, aiming for better local performance. However, the former approach may lead to models with limited generalization capabilities, while the latter can result in information loss, as knowledge gained from previous clients is not effectively incorporated. Addressing this challenge, this thesis proposes to leverage Graph Convolutional Neural Networks to enable learning domain-specific information while facilitating knowledge exchange between similar clients.

Federated vision benchmarks: to bridge the gap between FL research and real-world computer vision applications, this thesis proposes novel federated splits of vision datasets and establishes corresponding benchmarks in the fields of semantic segmentation for autonomous driving and collaborative visual place recognition. By demonstrating the effectiveness of state-of-the-art FL algorithms in these diverse and more complex tasks, this work paves the way for their wider adoption across various computer vision domains, broadening the applicability of FL.

1.3 Thesis Outline

The thesis is organized as follows:

Chapter 2 explores essential preliminary notions on machine learning, discussing the difference between supervised, weakly-supervised, unsupervised and self-supervised learning (Section 2.1). Section 2.2 details the architectures and training of Deep Neural Networks and Section 2.3 discusses optimization algorithms for machine learning. Generalization techniques proper of centralized learning are reviewed in Section 2.4.

Chapter 3 outlines the Federated Learning framework, discussing its typical settings (Section 3.1), the learning objective (Section 3.2) and the challenges arising in real-world scenarios (Section 3.3). The federated datasets employed in the experimental evaluations are presented in Section 3.4.

Chapter 4 introduces the first thesis' contribution, addressing generalization in heterogeneous FL. The generalization capability of the model is explored through the lens of the geometry of the loss landscape and is improved by encouraging convergence towards globally flat minima.

Chapter 5 presents novel approaches for improved generalization and convergence speedup in heterogeneous FL, based on clustering and grouping techniques. The local data distribution is used to detect and group similar (or dissimilar) clients, while preserving their privacy. Learning group-specific models effectively mitigates the issues deriving from data heterogeneity.

Chapter 6 addresses a critical aspect of current FL research: the lack of suitable benchmarks for complex vision tasks. While all the presented works focus on vision applications, this Chapter aims to enable future research in the federated vision field by introducing novel federated datasets and corresponding benchmarks. The addressed applications are semantic segmentation for autonomous driving (Section 6.2) and visual place recognition (Section 6.3).

Chapter 7 summarizes the thesis contributions (Section 7.1) and discusses open research directions and future works (Section 7.2).

1.4 Publications List

The author's publications are detailed below in chronological order.

- Caldarola, D., Mancini, M., Galasso, F., Ciccone, M., Rodolà, E., & Caputo, B. (2021).
  Cluster-driven Graph Federated Learning over Multiple Domains.
  In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop on Learning from Limited and Imperfect Data* (pp. 2749-2758) (**CVPRW 2021**).

- Caldarola, D., Caputo, B., & Ciccone, M. (2022).
  Improving Generalization in Federated Learning by Seeking Flat Minima.
  In *European Conference on Computer Vision* (pp. 654-672). Cham: Springer Nature Switzerland (**ECCV 2022**).

- Fantauzzo*, L., Fanì*, E., Caldarola, D., Tavera, A., Cermelli, F., Ciccone, M., & Caputo, B. (2022).
  FedDrive: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving.
  In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 11504-11511). IEEE. (**IROS 2022**)

- Zaccone*, R., Rizzardi*, A., Caldarola, D., Ciccone, M., & Caputo, B. (2022).
  Speeding Up Heterogeneous Federated Learning with Sequentially Trained Superclients.
  In *2022 26th International Conference on Pattern Recognition* (pp. 3376-3382). IEEE. (**ICPR 2022**)

- Shenaj*, D., Fanì*, E., Toldo, M., Caldarola, D., Tavera, A., Michieli, U., Ciccone, M., Zanuttigh, P., & Caputo, B. (2023).
  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* (pp. 444-454). (**WACV 2023**)

- Caldarola, D., Caputo, B., & Ciccone, M. (2023).
  Window-based Model Averaging Improves Generalization in Heterogeneous Federated Learning.
  In *Proceedings of the IEEE/CVF International Conference on Computer Vision, Women in Computer Vision Workshop* (pp. 2263-2271). (**ICCVW 2023**)

- Dutto, M., Berton, G., Caldarola, D., Fanì, E., Trivigno, G., & Masone, C. (2024).
  Collaborative Visual Place Recognition through Federated Learning.
  In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop on Federated Learning for Computer Vision.* (**CVPRW 2024**)

- Silvi*, A., Rizzardi*, A., Caldarola*, D., Caputo, B., & Ciccone, M. (2024).
  Accelerating Federated Learning via Sequential Training of Grouped Heterogeneous Clients.
  IEEE Access.

Preprint (currently under review in peer-reviewed conference):

- Caldarola, D., Cagnasso, P., Caputo, B., & Ciccone, M. (2024).
  Beyond Local Sharpness: Communication-Efficient Global Sharpness-aware Minimization for Federated Learning.

Chapter 2

Preliminaries

We are drowning in information
and starving for knowledge

JOHN NAISBITT

This Chapter establishes the essential background knowledge necessary for understanding the contributions presented in this thesis. Section 2.1 explores the fundamental principles of machine learning, differentiating between supervised, unsupervised and self-supervised learning. Particular emphasis will be placed on tasks relevant to the field of computer vision, namely image classification, semantic segmentation, and visual place recognition. These tasks will be the subject of further examination in subsequent Chapters. Section 2.2 explains the architecture of deep neural networks and their training. The concept of "generalization" and the challenge of developing models robust to data distribution shifts are discussed in Section 2.4, with a particular emphasis on methods that leverage the geometry of the loss landscape.

2.1 Machine Learning

Decades ahead of its time, Alan Turing's 1950 paper, *Computing Machinery and Intelligence* [47], predicted the potential for machines to learn. Turing's insightful analogy compared this learning process to a child's education, where the teacher provides the roadmap towards a desired outcome but may not fully understand the internal learning mechanisms at play. Today, Turing's vision has become a reality, as machines are now capable of remarkable feats of comprehension, including processing and generating text, images, and even spoken language, while learning from their past. Extending Turing's analogy further, the child in this scenario becomes the machine learning model. Just as a child learns from experience and instruction, the model learns from vast amounts of data, progressively improving its ability to understand and elaborate information. This learning process, also known as *training*, can be *supervised* or *unsupervised.* In supervised learning, the data comes with pre-defined labels or ground truth information, acting as the "teacher" guiding the model towards the desired outcome. Conversely, unsupervised learning involves unlabeled data, where the model must identify patterns and relationships within the data itself. *Self-supervised* and *weakly-supervised* learning instead take an intermediate approach. In the former, the data remains unlabeled and the model learns by creating its own supervisory signals from the inherent structure of the data. This allows the model to extract meaningful features and representations that can be beneficial for various downstream tasks. On the other hand, in weakly-supervised learning the algorithm learns from partially labeled or noisy data. For a comprehensive overview of machine learning paradigms, *reinforcement* learning [48] also deserves mention, which operates through a reward-penalty system. Here, the model learns by interacting with an environment and receiving positive or negative feedback (rewards or penalties) for its actions. This feedback loop allows the model to refine its behavior over time to maximize its cumulative reward. However, as reinforcement learning falls outside the scope of this thesis, it will not be further detailed. Readers interested in exploring this topic further can refer to [49–51].

The next Sections formalize supervised, weakly-supervised, unsupervised and self-supervised learning paradigms.

2.1.1 Supervised Learning

Supervised learning revolves around a **training set** made of labeled input-output pairs of the form $\mathscr{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ drawn from a distribution $P(x, y)$. Here, each $\boldsymbol{x}_i$ represents an input sample drawn from a feature space $\mathscr{X}$, y_i denotes its corresponding ground truth label residing in the label space $\mathscr{Y}$ and N is the number of samples. In the context of this thesis, $\boldsymbol{x}_i$ typically represents an image or a sentence, while y_i corresponds to the correct prediction for that specific input. This could be the image's class label (*e.g.*, "cat" or "dog") or the sentiment of a sentence (*e.g.*, "angry" or "happy").

In supervised training, the goal is to learn a function $f : \mathscr{X} \to \mathscr{Y}$ parametrized by $\boldsymbol{w} \in \mathbb{R}^d$ that effectively maps the input samples to their appropriate outputs, *i.e.*, $y_i = f(\boldsymbol{x}_i)$. The training process consists on fitting $\boldsymbol{w}$ to the input data. The quality of the learned mapping is measured via a **loss function** $\ell : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}^+ \cup \{0\}$, *i.e.*, a non-negative real-valued function which measures the difference between the true label y and its prediction $\hat{y}$, which corresponds to the class label with the highest predicted probability. Typically, $\ell(y, y) = 0$ and $\ell(\hat{y}, y) \geq 0 \; \forall y, \hat{y}$. The choice of loss function depends heavily on the specific task at hand. Examples of commonly used loss functions will be discussed in Section 2.1.1.

The *risk* is the expected loss of the function f across the entire data distribution P, *i.e.*, $R_P(\boldsymbol{w}) \triangleq \mathbb{E}_{(\boldsymbol{x}_i, y_i) \sim P}[\ell(f_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i)]$. However, the true data distribution P is often unknown in real-world scenarios. The concept of *empirical* risk, denoted by $R(\boldsymbol{w})$, tackles this challenge by estimating the true risk as the average loss of f over the available training set $\mathscr{D}$ made of N data points, *i.e.*,

$$R(\boldsymbol{w}) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{y}_i, y_i) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i), y_i\right). \tag{2.1}$$

A learned predictor is said to generalize well if the *generalization gap* [52–56] $R(\boldsymbol{w}) - R_P(\boldsymbol{w})$ is small. Intuitively, it quantifies how well the performance of the learned function transfers from seen to unseen samples. In other words, a smaller generalization gap signifies that the model has effectively learned the underlying patterns from the training data without simply memorizing specific examples. This process is named **generalization**. The opposite phenomenon is **overfitting** [57–60], happening when the learned function is too closely aligned with the training data.

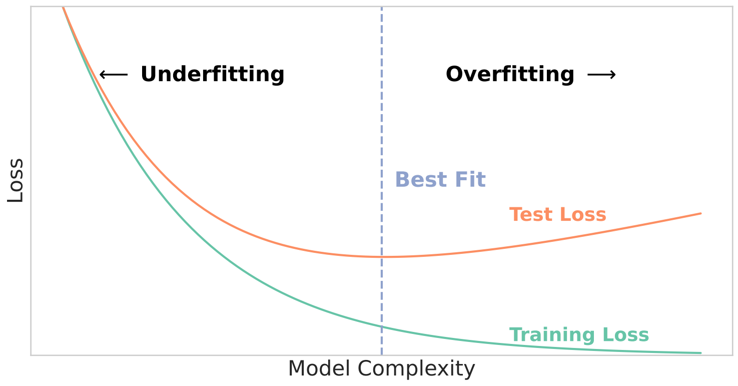


Fig. 2.1 **Underfitting *vs*. Overfitting**. **Underfitting** (*left*): a simple model struggles to capture the underlying patterns in the data, resulting in high errors on both the training and test sets. **Overfitting** (*right*): a complex model memorizes the specifics of the training data, including noise. While it performs well on the training set, it fails to generalize to unseen examples, leading to high errors in the test set. The ideal model (*center*) achieves a balance between simplicity and complexity, with strong generalization ability.

This results in exceptional performance on seen examples, but poor generalization on unseen ones. Lastly, **underfitting** occurs when the model fails to capture the underlying patterns between input and output data, resulting in high loss on both the training set and unseen samples [58, 60, 61]. As shown in Figure 2.1, these phenomenons are highly correlated with the model complexity: a too simple model may not be able to capture complex patterns in the data, but a overly complex model is likely to capture noise in the training data rather than the underlying signal [57, 62, 59].

Image Classification

Supervised learning aims to uncover the relationships between labeled inputs and desired outputs. The specific task at hand depends on the nature of the training data, $\mathscr{D}$, and the corresponding ground truth information, $\mathscr{Y}$. In this thesis, the focus is on **classification**, where the model learns to categorize input data points into predefined classes. **Semantic Segmentation** takes a step further, aiming to classify each pixel of the input image (*e.g.*, segmenting an image to identify individual pixels belonging to a cat or a dog).

Classification aims to learn a mapping from inputs $\boldsymbol{x} \in \mathscr{X}$ to outputs $y \in \mathscr{Y}$, where $\mathscr{Y}$ is a set of C discrete labels representing the number of classes, *i.e.*, $\mathscr{Y} = [C] =$

$\{1,\cdots,C\}$. The number of classes C determines the classification type: binary for $C=2$ and multiclass for $C>2$. In image classification, the classes refers to objects depicted in the images. For instance, animals (*e.g.*, dog *vs.* cat), or cities (*e.g.*, Rome *vs.* New York).

For classification tasks, the most commonly employed loss function is **cross-entropy** (CE) [63], ℓ^{ce}. The CE loss is based on the concept of the Shannon *entropy* [64] H of a random variable X, *i.e.*, the average uncertainty inherent in the variables possible outcome,

$$H(X) = -\sum_{x\in X} p(x)\log p(x), \tag{2.2}$$

where p is the probability distribution and X is a discrete variable. The larger the value of $H(X)$, the greater the uncertainty of the probability distribution. The *cross-entropy* measures the distance between the estimated output probabilities and the true output distribution, where each predicted class probability is compared to the real desired output (0 or 1). The loss penalizes the probability based on how far it is from the ground truth. Its logarithmic nature leads to large penalties for large differences and small penalties for divergences tending to 0, *i.e.*, the function penalizes the confident and wrong predictions the most. The CE loss is defined as

$$\ell^{\text{ce}} \triangleq -\sum_{i=1}^{C} y_i \log(p_i), \tag{2.3}$$

where y_i is the ground truth label and p_i the softmax probability of the ith class.

To measure the model performance in multiclass classification, the main reference metrics is the **accuracy**, defined as

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}. \tag{2.4}$$

Semantic Segmentation. Semantic segmentation [65–67] is a classification task whose goal is to provide a semantic prediction for each pixel in the image, without distinguishing between multiple instances of the same class (*e.g.*, all people are labeled as "person"). For instance, semantic segmentation may identify the portion of the image that corresponds to "sky", "car", or "person" (see Figure 2.2). Each image $\boldsymbol{x} \in \mathscr{X}$ made of N_P pixels is associated with a ground truth vector $\boldsymbol{y} \in \mathscr{Y}^{N_P}$, denoting the set of N_P tuples with elements belonging to the output space $\mathscr{Y}$. Given

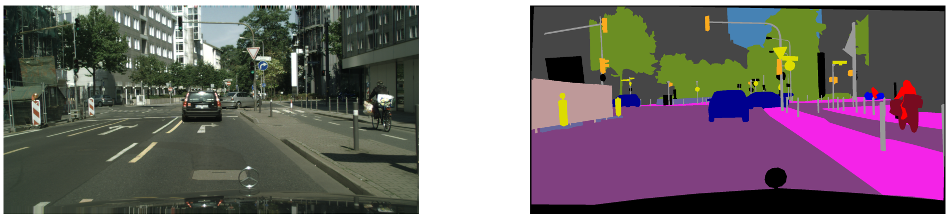

(a) Original image (b) Semantic segmentation map

Fig. 2.2 Example of **semantic segmentation**. *(a)* Original image. *(b)* Each pixel is assigned a color-coded label, revealing the objects it depicts. For instance, the label `tree` is depicted in green, `person` in red and `car` in blue. Images from the Cityscapes dataset.

the input $\boldsymbol{x}$, the goal is to learn a function $f : \mathscr{X} \rightarrow \mathscr{Y}^{N_P}$ mapping each pixel x_i of the image $\boldsymbol{x}$ to a class $y_i \in \mathscr{Y}$. Pixels that fall outside the identified C classes are assigned to a special label, *e.g.*, "background" $\mathtt{b} \in \mathscr{Y}$ or "ignore-index" $\mathtt{i} \in \mathscr{Y}$.

The performance of a semantic segmentation model is usually evaluated in terms of **mean Intersection over Union** (mIoU) [68]. The Intersection over Union computes the overlap between the class' predicted segment and ground truth as

$$IoU(c) = \frac{TP(c)}{TP(c)+FP(c)+FN(c)} \tag{2.5}$$

$$\Rightarrow mIoU = \frac{1}{C}\sum_{c=1}^{C} IoU(c), \tag{2.6}$$

where $TP =$ True Positive, $FP =$ False Positive and $FN =$ False Negative for each class c. The IoU is equal to 1 if the prediction matches the ground truth, *i.e.*, the intersection corresponds to the union, and is zero if the intersection is null.

Knowledge Distillation

Knowledge Distillation (KD) [69] is a technique employed to transfer the knowledge from a complex, high-performing model (*teacher*) , f_t, to a usually smaller model (*student*), f_s. This process enables the lightweight model to achieve better performance by learning to mimic the teacher's knowledge.

Given an input sample $\boldsymbol{x}_i$, a neural network f typically outputs class probabilities by leveraging a softmax function that converts the logits $\boldsymbol{z}_i$ into a probability p_i as

$$p_i(\boldsymbol{z}_i, \mathrm{T}) = \frac{\exp\left(z_i/\mathrm{T}\right)}{\sum_j \exp\left(z_j/\mathrm{T}\right)}, \tag{2.7}$$

by comparing z_i with the other logits, smoothing the probability distribution with the use of the temperature T, normally set to 1.

In supervised settings, a common approach to KD is to leverage the teacher's probabilities to train the student model in predicting the correct labels by minimizing the distillation loss, as

$$\ell^{\mathrm{KD}}(\boldsymbol{x}_i, \mathrm{T}) \triangleq \gamma \cdot \ell^{\mathrm{ce}}\left(p_i\left(f_s(\boldsymbol{x}_i), \mathrm{T}=1\right), y_i\right) + \tag{2.8}$$

$$+\left(1-\gamma\right) \cdot \ell^{\mathrm{ce}}\left(p_i\left(f_t(\boldsymbol{x}_i), \mathrm{T}=\tau\right), p_i\left(f_s(\boldsymbol{x}_i), \mathrm{T}=\tau\right)\right), \tag{2.9}$$

where x_i is the input, y_i the ground truth label, p_i is the softmax function (Equation (2.7)), T the temperature and $\gamma \in \mathbb{R}$ a coefficient balancing the importance of the teacher's guidance.

2.1.2 Weakly-Supervised Learning

Weakly-supervised learning emerges as a powerful approach when acquiring fully labeled data proves challenging or expensive [70, 71]. This method enables models to learn from data with partial labels or inherent noise. Visual Place Recognition (VPR), explored in the following section, exemplifies a successful application of weakly-supervised learning. While GPS information provides a convenient way to label image locations (often embedded automatically in photos), it falls short in capturing the complete picture. Crucially, VPR needs to account for variations in camera viewpoint, which are often absent from GPS data. This necessitates the use of alternative techniques proper of the weak supervision to overcome this limitation.

Visual Place Recognition

Visual Place Recognition [72, 73] aims to estimate the location of a given input image, called *query*, and acts as a *image retrieval* system specifically designed for places. VPR is based on a large geo-tagged database, *i.e.*, images with known GPS position. Samples from the same place (or within a certain radius) are referred to as *positives* (*e.g.*, images taken from different viewpoints), while images from different locations are called *negatives* (*e.g.*, photos of various cities). Given a query image, the goal is to find the most similar match within the database to estimate the input's geographical location. This is achieved by learning a function $f : \mathscr{X} \rightarrow \mathscr{E}$ which projects the input sample onto the embedding space $\mathscr{E}$, where representations of the same place are close to each other while being apart from representations of other locations.

VPR models are usually trained using contrastive losses [74–76], which aim to bring representations of positive images closer in the embedding space while distancing representations of negatives. In general, VPR favors the **triplet loss** [77, 75]. Received a query (*anchor*) $\boldsymbol{q}$, a positive $\boldsymbol{p}^q$ and a negative $\boldsymbol{n}^q$ images as input, the triplet loss aims to minimize the distance from the anchor to the positive, while maximizing the distance from the anchor to the negative input, as

$$\ell^{\text{triplet}} \triangleq \max(\|f(\boldsymbol{q}) - f(\boldsymbol{p}^q)\| - \|f(\boldsymbol{q}) - f(\boldsymbol{n}^q)\| + m, 0), \tag{2.10}$$

with m being the margin between positive and negative pairs, treated as a hyperparameter.

According to Equation (2.10), if the chosen negative is far away from the query, *i.e.*, $\|f(\boldsymbol{q}) - f(\boldsymbol{n}^q)\| \rightarrow \infty$, $\ell^{\text{triplet}} = 0$, implying a non-informative choice. To avoid this issue, *hard negatives* are to be favored, *i.e.*, closer or visually similar negatives w.r.t. query [76]. Acquiring positive training examples for VPR is relatively straightforward. The GPS information is leveraged to identify images within a specific radius (τ) of the query location, creating the set of positive images denoted as $\mathscr{P} \triangleq \{\boldsymbol{p}_i^q\}$. However, selecting the set of hard negatives $\mathscr{N} \triangleq \{\boldsymbol{n}_i^q\}$ represents a challenge, since all images outside the designated radius are considered. This process is known as **mining** and can be time-consuming and computationally expensive. In addition, the GPS information alone is not sufficient to determine whether two images are visually similar. Thus, the best positive and negative among the candidates are chosen as

$$\boldsymbol{p}^q = \underset{\boldsymbol{p}_i^q \in \mathscr{P}}{\arg\min} D_{\boldsymbol{w}}(\boldsymbol{q}, \boldsymbol{p}_i^q), \tag{2.11}$$

$$\boldsymbol{n}^q = \underset{\boldsymbol{n}_i^q \in \mathscr{N}}{\arg\min} D_{\boldsymbol{w}}(\boldsymbol{q}, \boldsymbol{n}_i^q), \tag{2.12}$$

where $D_{\boldsymbol{w}}$ is the Euclidean distance computed using the current estimate of $f_{\boldsymbol{w}}$. Since VPR relies on estimates of the best $\boldsymbol{p}^q$ and $\boldsymbol{n}^q$, going beyond strictly using ground truth information, it becomes a prime example of weakly-supervised learning.

2.1.3 Unsupervised Learning

The key difference between supervised and unsupervised learning lies in the input data. In unsupervised learning [78–80], the training set is **unlabeled**, *i.e.*, it is of the form $\mathscr{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$. The learning objectives are multiple and vary from revealing hidden structures or patterns within the data [81] and reducing dimensionality [82] to detecting anomalies [83, 84] and clustering similar data [85–87]. The absence of labels presents a challenge, as the model doesn't have pre-defined categories or outputs to learn from. However, this also unlocks a key advantage: broad applicability. Unsupervised learning algorithms can be applied to a vast range of data types and domains because they don't require human-labeled data, which can be expensive and time-consuming to obtain in large quantities.

Clustering

Clustering is a fundamental technique in unsupervised learning [88]. It involves grouping unlabeled data points together based on their similarities, measured using a defined metric, such as distance in a specific feature space. By grouping similar data points together, clustering helps uncover hidden structures within the data and provides valuable insights into the underlying relationships between data points.

Among the clustering techniques, K-means [89, 90] is of interest in this thesis. The K-means clustering algorithm partitions a dataset into K clusters by iteratively assigning data points to the nearest cluster centroid and updating the centroids based on the mean of the points in each cluster:

$$\mathfrak{C}_i = \left\{ x^{(j)} : \|x^{(j)} - \mu_i\| \leq \|x^{(j)} - \mu_{i'}\| \text{ for all } i' \neq i \right\}.$$

$\mathfrak{C}_i$ denotes the set of data points assigned to cluster i and μ_i represents the centroid of cluster i, identified as the mean of all points currently assigned to that cluster, as $\mu_i = \frac{1}{|\mathfrak{C}_i|} \sum_{x \in \mathfrak{C}_i} x$. This process iterates until convergence, minimizing the total intra-cluster variance. The k-means algorithm is efficient and straightforward, revealing

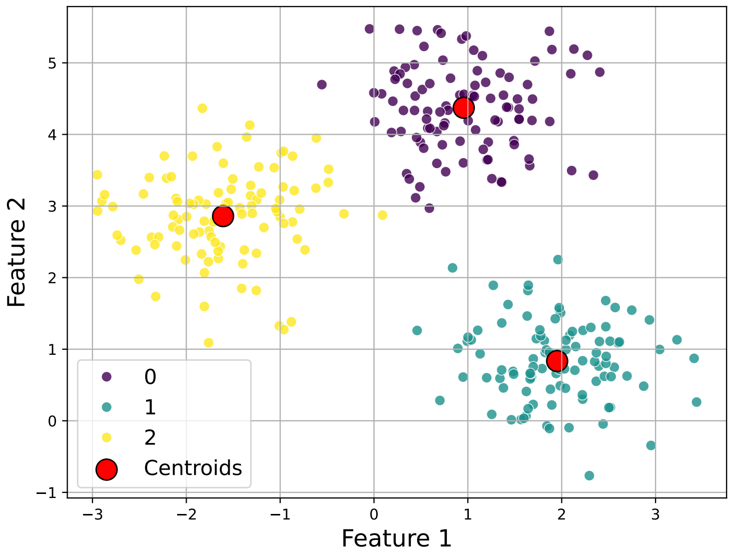


Fig. 2.3 Example of application of K-means with $K = 3$ clusters

underlying patterns in the data's structure, as illustrated in Figure 2.3 for $k = 3$ clusters.

2.1.4 Self-Supervised Learning

In self-supervised learning (SSL) [91], the system learns to understand and represent data without relying on manually labeled examples. Differently from unsupervised learning, it generates its own labels by leveraging the inherent structure of the data. This is often achieved by formulating auxiliary tasks, such as predicting missing parts of data or identifying transformations applied to the data. SSL has gained importance for its ability to use large amounts of unlabeled data, making it particularly valuable in scenarios where labeled data is scarce or expensive to obtain. By learning from the data itself, models trained with SSL techniques can achieve high levels of performance on downstream tasks, often rivaling those trained with extensive labeled datasets. Examples of SSL applications are next character and next word predictions, where part of the sentence is masked and the model is asked to predict the missing part [92, 93].

2.2 Deep Neural Networks

Deep Learning (DL) [94] is a subset of machine learning that leverages Neural Networks (NNs) to process information. Building upon the biological concept of neurons and neural networks in the brains of animals and humans, Artificial Neural Networks (ANNs) consist of interconnected units or nodes called *artificial neurons*. These artificial neurons exchange signals in the form of real numbers through connections akin to synapses in the brain.

A neuron processes N inputs $\boldsymbol{x}$ (*e.g.*, images) using a non-linear function f, which maps them to an output value $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}+b\right), \tag{2.13}$$

where $\boldsymbol{w}\cdot\boldsymbol{x} = \sum_{i=1}^{N} w_i x_i$ is the dot product between the weights $\boldsymbol{w}$ and the inputs $\boldsymbol{x}$, b is the bias and σ is a non-linear activation function, *e.g.*, ReLU [95]. Equation (2.13) represents the formulation of the **perceptron** [96]. To improve the capability of the network, multiple perceptrons can be combined in L connected **layers**, where the output of each layer l becomes the input of the subsequent layer $l+1$ as

$$\boldsymbol{x}_{l+1} \leftarrow \sigma(\boldsymbol{w}_l\cdot\boldsymbol{x}_l+b_l), \tag{2.14}$$

until the final representation $\boldsymbol{x}_L = \text{Sigmoid}(\boldsymbol{w}_{L-1}\cdot\boldsymbol{x}_{L-1}+b_{L-1})$ is computed. The Sigmoid function transforms the output of the model into a probability vector. A **hidden layer** in a neural network is an intermediate layer situated between the input and output layers. Unlike the input and output, which directly handle the data and final predictions, hidden layers perform computations and transformations on the input data through weighted connections and activation functions. They are responsible for capturing complex patterns and representations within the data, enabling the network to model intricate relationships and enhance predictive accuracy. The last layer of a neural network is usually named *classifier*, while the backbone is the *feature extractor*. A **Deep Neural Network** (DNN) consists of multiple stacked hidden layers and is the most used model for complex and modern tasks, *e.g.*, computer vision [97–99], natural language processing [14, 100, 101], image and text generation [102, 15]. Figure 2.4 shows an example of DNN.

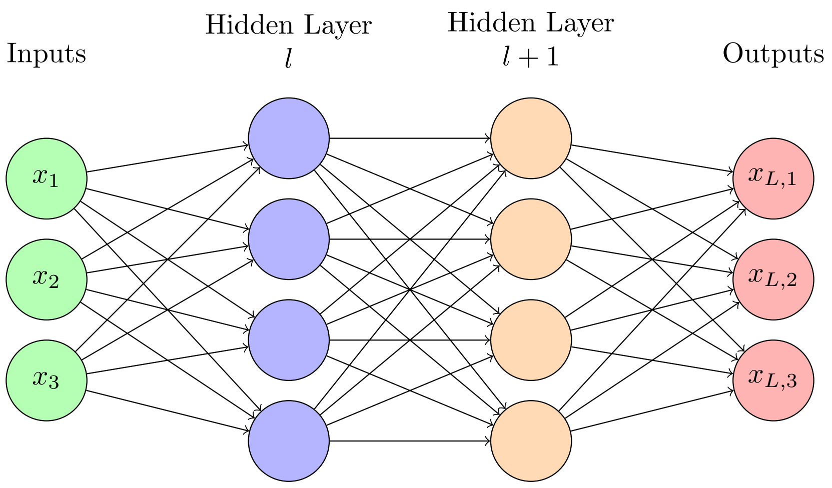


Fig. 2.4 Example of an **Artificial Neural Network**. The network receives the inputs $\{x_1, x_2, x_3\}$ and processes them through multiple hidden layers (l and $l+1$), until the final representation $x_{L,i}$ is reached for $i = 1, \ldots, 3$.

Depending on the nature of the functions applied to inputs, weights, and biases, as well as the connections between layers, multiple types of DNNs can be defined. The **multilayer perceptron** (MLP) [103, 104], or **feedforward neural network**, consists of fully connected neurons, as illustrated in Figure 2.4. While the full connections in MLPs enable complex mappings from inputs to outputs, they also make training more difficult due to the large number of parameters and potential for overfitting. To this end, **Convolutional Neural Networks** [94, 105] reduce the number of parameters and improve the robustness of the network, as discussed in Section 2.2.1. This thesis also focuses on Graph Neural Networks [106, 107], which process data representable via the means of a graph (Section 2.2.2). For completeness, another relevant example of DNNs is **Recurrent Neural Networks** [108, 109], which enable the use of memory when processing subsequent inputs, making them applicable to tasks like handwriting and speech recognition [110, 111].

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specialized type of deep neural network designed to process and analyze grid-like data structures, such as images. In a convolutional layer, neurons are organized in three dimensions: width, height, and depth. Each layer transforms an input 3D volume into an output 3D volume. Unlike

Table 2.1 Formulas for Batch Normalization (BN) and Group Normalization (GN). BN layers act on a batch of size m, while GN layers act on groups g of channels.

Batch Normalization (BN)	**Group Normalization (GN)**
$\mu_{\text{BN}} = \frac{1}{m}\sum_{i=1}^{m} x_i$	$\mu_g = \frac{1}{m}\sum_{i\in\text{group}_g} x_i$
$\sigma_{\text{BN}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\text{BN}})^2$	$\sigma_g^2 = \frac{1}{m}\sum_{i\in\text{group}_g}(x_i - \mu_g)^2$
$\hat{x}_{\text{BN},i} = \frac{x_i-\mu_{\text{BN}}}{\sqrt{\sigma_{\text{BN}}^2+\epsilon}}$	$\hat{x}_{\text{GN},i} = \frac{x_i-\mu_g}{\sqrt{\sigma_g^2+\epsilon}}$
$y_{\text{BN},i} = \gamma\hat{x}_{\text{BN},i} + b = \text{BN}_{\gamma,b}(x_i)$	$y_{\text{GN},i} = \gamma\hat{x}_{\text{GN},i} + b = \text{GN}_{\gamma,b}(x_i)$

traditional fully connected networks, CNNs utilize convolutional layers that apply a set of learnable filters to the input data, efficiently capturing spatial hierarchies and patterns. These filters are spatially small but extend through the full depth of the input volume. The core operation within each layer is the convolution between the input and the filter weights. As the filter slides over the input, it produces an activation map that highlights relevant visual features. These filters, combined with *pooling layers* that reduce dimensionality, enable CNNs to learn hierarchical features such as edges, textures, and shapes. This architecture makes CNNs highly effective for tasks such as image classification, object detection, and segmentation, where spatial context and local dependencies are crucial. Examples of CNNs are LeNet5 [112], AlexNet [113] and ResNet [114].

ResNet. A case study relevant to this thesis is the Residual Neural Network (ResNet), that uses identity skip connections inside the model architecture, also known as “residual connections”, to make the layers fit a residual mapping. Thus, given an input $\boldsymbol{x}$, the ResNet aims to modify the original mapping function from $f(\boldsymbol{x})$ to $f(\boldsymbol{x}) + \boldsymbol{x}$. The shortcut connections perform the identity mapping, which is added to the output of the stacked layers. This approach improves the robustness of the model by reducing the *vanishing gradient*, discussed in Section 2.3.2.

Normalization

Data normalization is often used to ensure stability and accelerated convergence during training. A common technique is the use of Batch Normalization (BN) layers [115], which are often added between hidden layers of a CNN to normalize the input through re-centering and re-scaling to have zero mean and unit variance.

Table 2.1 summarizes the step of the BN transform for each sample $\boldsymbol{x}$ of a mini-batch $\mathscr{B} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$. The scale γ and the bias b are learnable parameters, while μ and σ are the BN statistics, namely *running mean* and *running variance*, respectively. ε is a small scalar that prevents division by 0. Thus, $\mathrm{BN}_{\gamma,b}$ depends on *all the examples in the mini-batch*. This approach makes the training of neural networks faster and more stable. However, it may require large mini-batches to calculate stable statistics, which may not be feasible in memory-constrained environments or with limited data and the performance can degrade when the batch size is small.

Differently from BN, Group Normalization (GN) [116] computes normalization over the spatial dimensions and groups of channels, not across the batch dimension. Thus, the normalization is independent of the batch size and works well with limited data. Given an input $\boldsymbol{x}$, its C channels are divided into G groups, each containing C/G channels, which are normalized independently.

2.2.2 Graph Neural Networks

Graph Neural Networks (GNNs) are models capable of learning functions over graphs and are usually leveraged to learn predictive models over graph-structured data. Formally, a graph is represented as $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ are the sets of vertices and edges respectively. Let $v_i \in \mathscr{V}$ denote a node, $e_{ij} = (v_i, v_j) \in \mathscr{E}$ be the edge connecting v_i and v_j, and $n = |\mathscr{V}|$ the number of nodes. The *adjacency matrix* $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is defined such that $\boldsymbol{A}_{ij} = \gamma_{ij}$ if $e_{ij} \in \mathscr{E}$ and $\boldsymbol{A}_{ij} = 0$ otherwise, with $\gamma \in \mathbb{R}^+$ being the weight of e_{ij}. GNNs usually focus on node or edge classification, link prediction and graph classification [117].

Graph Convolutional Neural Networks. Among the existing architectures, this thesis focuses on Graph Convolutional Neural Networks (GCNs) [118]. GCNs leverage both the features of a node and its neighborhood to make predictions, leading to successful results in various scenarios [119]. Each layer l of the GCN is based on the following propagation rule:

$$\boldsymbol{Z}^{l+1} = \sigma\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{Z}^l \boldsymbol{W}^l\right), \tag{2.15}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$ is the adjacency matrix with added self-connections, $\boldsymbol{I}_n$ is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $\boldsymbol{W}^l$ is a trainable weight matrix of layer l and σ a non-linear activation function. $\boldsymbol{Z}^l$ is the activation matrix and is initially set equal to the input $\boldsymbol{X}$. By conditioning $\boldsymbol{Z}$ on the adjacency matrix, the gradient information from the supervised loss is propagated across the entire graph, with weights eventually adjusted based on the proximity of the nodes.

2.3 Optimization in Machine Learning

Section 2.1.1 discussed the empirical risk (Equation (2.1)) as a measure to evaluate the loss occurring in the current model's predictions. This Section describes the training process to minimize the loss and, by extension, the empirical risk.

The standard approach to train a predictor is **Empirical Risk Minimization** (ERM) [120], where the best model parameters $\boldsymbol{w}^*$ are the ones that minimize the *expected loss* of the function f over the training set $\mathscr{D}$. In simpler terms, the goal is to find $f_{\boldsymbol{w}^*}$ such that Equation (2.1) is minimized for the data in $\mathscr{D}$. The ERM problem is formalized as follows

$$f_{\boldsymbol{w}^*} = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\arg\min} R(\boldsymbol{w}) = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\arg\min} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(f_{\boldsymbol{w}}(\boldsymbol{x}), y)], \tag{2.16}$$

and is solved using optimization algorithms. Among those, **gradient descent** is of interest for this work.

2.3.1 Gradient Descent

Gradient descent (GD) [121] is an iterative first-order optimization algorithm, aiming to find local minima (or maxima) of a given function. In ML applications, it is used to minimize loss functions [122, 123].

Given a function $f : \mathscr{X} \to \mathscr{Y}$ parametrized by $\boldsymbol{w} \in \mathbb{R}^d$, its gradient ∇ is a vector that contains all the partial derivatives of f with respect to each component of the parameter vector $\boldsymbol{w}$. Formally, if $f_{\boldsymbol{w}}$ is a scalar-valued function (*i.e.*, $f : \mathbb{R}^d \to \mathbb{R}$), the gradient of f with respect to $\boldsymbol{w}$ is denoted as $\nabla_{\boldsymbol{w}} f$ and is defined as:

$$\nabla_{\boldsymbol{w}} f \triangleq \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}. \tag{2.17}$$

If $f_{\boldsymbol{w}}$ is a vector-valued function (*i.e.*, $f : \mathbb{R}^d \to \mathbb{R}^m$), the gradient is generalized to the Jacobian matrix, which contains the partial derivatives of each component of f with respect to each element of $\boldsymbol{w}$. In such cases, the Jacobian $\boldsymbol{J}_{\boldsymbol{w}}(f)$ is defined as:

$$\boldsymbol{J}_{\boldsymbol{w}}(f) \triangleq \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \frac{\partial f_1}{\partial w_2} & \cdots & \frac{\partial f_1}{\partial w_d} \\ \frac{\partial f_2}{\partial w_1} & \frac{\partial f_2}{\partial w_2} & \cdots & \frac{\partial f_2}{\partial w_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial w_1} & \frac{\partial f_m}{\partial w_2} & \cdots & \frac{\partial f_m}{\partial w_d} \end{bmatrix}. \tag{2.18}$$

The second-order partial derivatives form the **Hessian matrix** [124], which captures the local *curvature* of the function. This curvature provides insights on the *sharpness* of a local minimum: a high curvature indicates a sharp minimum, whereas a low curvature suggests a flat minimum. For a scalar-valued function, the Hessian matrix is defined as

$$\boldsymbol{H}_{\boldsymbol{w}}(f) \triangleq \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_d} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_d \partial w_1} & \frac{\partial^2 f}{\partial w_d \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_d^2} \end{bmatrix}. \tag{2.19}$$

Thus, the gradient acts as a compass in the landscape of a function. It points in the direction of the steepest ascent, indicating how quickly the function's value increases as moved in that direction. The magnitude of the gradient reflects the rate of this increase – a larger magnitude signifies a steeper climb. In the context of ML, the gradient of the loss function points towards the direction of higher-loss points. This implies that the opposite direction of the gradient, $-\nabla_{\boldsymbol{w}}$, indicates the path of steepest *descent* and guides towards the lower-loss points.

Gradient descent leverages the negative gradient direction of the loss function to guide the model towards stationary points. If the function is convex, this will be

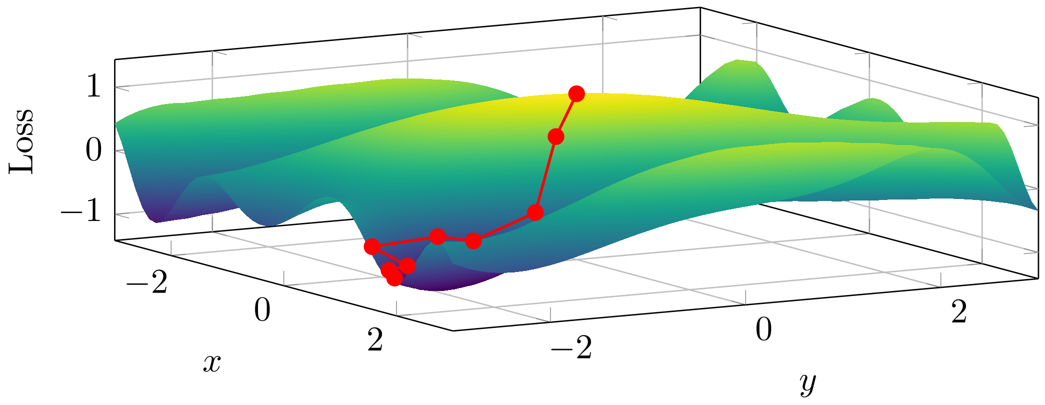


Fig. 2.5 Example of visualization of **Stochastic Gradient Descent** path in a high-dimensional *non-convex* loss landscape. The path taken by SGD is marked by red dots indicating individual gradient steps. Starting from a high-loss point, SGD leads the model towards a local minimum.

a global minimum, otherwise it could end up in a local minimum [125]. Starting from a point $\boldsymbol{w}$ in the loss landscape at iteration i, GD iteratively updates the model parameters as

$$\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - \eta \nabla_{\boldsymbol{w}_i} f_{\boldsymbol{w}_i}, \tag{2.20}$$

where $\eta \in \mathbb{R}^+$ is the **learning rate** and has a great impact on the learning process. Too small learning rates slow down the model, increasing the number of iterations needed to reach a local minimum. Conversely, excessively large learning rates might cause the algorithm to overshoot the optimal point or even diverge entirely.

Unfortunately, computing the gradient over all data points for each step, as done in Equation (2.20), is computationally intensive, and its cost increases linearly with the size of the training set. As a solution, **Stochastic Gradient Descent** (SGD) [126, 121] estimates the true gradient using a randomly sampled batch of data of size B and takes a small step in the resulting negative direction. Such approach enables fast computations, regardless of the dataset size, while still giving an unbiased estimate of the true gradient. In addition, for SGD to converge, the step size η has to decrease over time. Various learning rate *schedulers* exist, such as cosine annealing, time-based decay and exponential decay [127]. Algorithm 1 summarizes SGD.

Algorithm 1 Stochastic Gradient Descent (SGD)

```
Require: Learning rate $\eta$, initial parameter $\boldsymbol{w}_0$
1: while not converged do
2:     Sample a minibatch of data points $\{x_i, y_i\}_{i=1}^{B}$
3:     Compute the gradient estimate: $\nabla_{\boldsymbol{w}} \hat{\mathscr{L}}(\boldsymbol{w}) = \frac{1}{B}\sum_{i=1}^{B} \nabla \ell(f_{\boldsymbol{w}}(x_i), y_i)$
4:     Update the parameters: $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - \eta_i \nabla_{\boldsymbol{w}} \hat{\mathscr{L}}(\boldsymbol{w})$
5: end while
```

2.3.2 Training Deep Neural Networks

The goal of training a DNN is to learn the weights $\boldsymbol{w}_l$ and the biases b_l of each layer l (Equation (2.14)) so that the model f can accurately map the inputs $\boldsymbol{x}$ to their corresponding outputs $\boldsymbol{y}$, *i.e.*, $\hat{y} = f_{\boldsymbol{w}}(\boldsymbol{x})$ with $\hat{y}$ being the predicted output.

The learning objective is to minimize the empirical risk, based on a loss function ℓ (Equations (2.1) and (2.16)). In deep learning contexts, ℓ typically takes the form of a non-convex function. This means it may have multiple local minima (Figure 2.5), making the optimization process more challenging. The training of a DNN happens in two phases: *i*) **forward pass**, and *ii*) **backward pass** or **backpropagation** [128]. In the former, the input data $\boldsymbol{x}$ traverses each layer of the network as in Equation (2.14) and the produced layer-specific output is stored. During backpropagation instead, the weights and biases of each layer are modified such that the loss between the model's predictions and the ground truth is minimized, solving Equation (2.16). The adjustment is determined by the gradient of the loss function w.r.t. those parameters. Given a DNN of L layers with weights $\boldsymbol{w}_l$ and biases $b_l \forall l \in [L]$, the gradient $\nabla \ell$ of the loss function (Equations (2.17) and (2.18)) is

$$\nabla \ell = \left[\frac{\partial \ell}{\partial w_1}, \frac{\partial \ell}{\partial b_1}, \dots, \frac{\partial \ell}{\partial w_L}, \frac{\partial \ell}{\partial b_L}\right]^{\top} \tag{2.21}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \ell = \left[\frac{\partial \ell}{\partial w_1}, \dots, \frac{\partial \ell}{\partial w_L},\right]^{\top} \quad \text{and} \quad \nabla_b \ell = \left[\frac{\partial \ell}{\partial b_1}, \dots, \frac{\partial \ell}{\partial b_L}\right]^{\top}. \tag{2.22}$$

The gradients in Equation (2.22) are computed using a technique called the **chain rule**, that leverages the equivalence $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z}\frac{\partial z}{\partial x} \forall x, z$. First, the gradient of the loss function ℓ with respect to the predicted output $\hat{y}$ is computed, $\frac{\partial \ell}{\partial \hat{y}} = \frac{\partial \ell}{\partial \hat{y}}$. Then, the gradient is propagated backward through each layer l from the output layer L to the input one ($l = 1$). For the layer L, being z_L the input to the activation function of the output layer,

$$\delta_L = \frac{\partial \ell}{\partial z_L} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_L}. \tag{2.23}$$

For the layers $l = L-1, \dots, 1$,

$$\delta_l = \left(\delta_{l+1} \frac{\partial z_{l+1}}{\partial x_l} \right) \cdot \frac{\partial x_l}{\partial z_l}, \tag{2.24}$$

where x_l is the output of layer l. Lastly, the gradient of the loss function with respect to the weights and biases for each layer l is

$$\frac{\partial \ell}{\partial w_l} = \delta_l \frac{\partial z_l}{\partial w_l} \quad \text{and} \quad \frac{\partial \ell}{\partial b_l} = \delta_l. \tag{2.25}$$

Once $\nabla \ell$ is computed, the chosen optimization algorithms determines the update rule of the parameters, *e.g.*, SGD.

During training, DNNs can be subject to two relevant problems: *vanishing* and *exploding* gradients [129, 130]. The vanishing gradient problem occurs when the gradients of the loss function with respect to the model parameters become exceedingly small as they are propagated back through the layers. This results in very slow updates to the weights of the earlier layers, effectively preventing the network from learning. Conversely, the exploding gradient problem arises when these gradients become excessively large, causing the weights to update too aggressively and potentially leading to numerical instability and divergence. Both issues are more prevalent in deep networks with many layers and can hinder the effective training of the model. Techniques such as proper weight initialization [131], gradient clipping [132], batch normalization [133, 134] and the use of advanced architectures like ResNets are often employed to mitigate these problems.

2.4 Generalization in the Real World

Achieving strong generalization performance is a fundamental objective in developing deep learning models [135]. As introduced in Section 2.1.1, generalization refers to a model's capacity to accurately make predictions on previously unseen data, demonstrating its ability to learn underlying patterns and not simply memorize the training data. Generalization is crucial for real-world applications, where deep learning models encounter data that may differ statistically from the training set. For example, a medical image analysis model should not only excel at classifying diseases on the training images but also maintain this accuracy on new patient data. Insufficient generalization can lead to overfitting or underfitting, with consequent poor performance in practical settings. In essence, a well-generalized model can effectively adapt its knowledge to new situations. This concept is formalized by the *generalization gap*, *i.e.*, the difference between the true and the empirical risk (Equation (2.1)).

In over parametrized settings, where neural networks have a large number of parameters, the learning objective has multiple local minima (Section 2.3.2). While these minima all achieve low training error, they may not generalize equally well to unseen data. Essentially, getting stuck in the "wrong" local minimum can lead to significantly different performance on new data. Several factors influence the optimization path during training and can affect the likelihood of ending up in a sub-optimal minimum [135]. These include:

- **Choice of the optimizer**: different optimization algorithms have varying tendencies when navigating the loss landscape [136].
- **Hyperparameter tuning**: hyperparameters, such as the learning rate, control the learning process and influence the path taken by the optimizer. Careful selection of these hyperparameters can help steer the optimization towards better minima. Techniques like grid search or random search can be employed for hyperparameter optimization [137, 138].
- **Initialization strategy**: the initial values of the network's parameters play a fundamental role, since they determine the starting point of the optimization path [139, 131]. A common technique to achieve a better starting point is to leverage *pre-trained* models. These models have already been trained on a different, potentially larger dataset, and their learned parameters can be used to initialize a new network. This approach can significantly improve the convergence speed and performance of the new model, especially when dealing with limited training data [140].
- **Data distribution**: imbalanced data distributions, where one or a few classes significantly outnumber the others, can severely affect a model's ability to generalize. The presence of a majority class/group can bias learning models,

potentially leading them to overlook patterns in the minority class [141, 142]. This can lead to *unfair* behaviors towards the minority group, such as performance disparity and inherent bias in the model's predictions [143].

- **Sharpness**: the sharpness measures how sensitive a model's training error is to slight parameter changes. Keskar *et al.* [52] introduced the notion of sharpness as a proxy for generalization. Flatter solutions are believed to indicate better generalization, as the training error remains relatively stable under small parameter perturbations.

The following sections discuss in detail the aforementioned choices. Section 2.4.1 introduces the difference between Domain Adaptation and Domain Generalization, together with common techniques to improve the model's generalization ability. Next, the thesis focuses on the impact of momentum on SGD, the preferred optimizer. Section 2.4.3 discusses the challenges posed by imbalanced data distributions and their effect on the learning process, while the relevance of the model architecture and its initialization strategy are analyzed in Section 2.4.4. Section 2.4.5 discusses the recent research trend linking generalization with convergence to flat minima in the loss landscape.

2.4.1 Domain Adaptation *vs*. Domain Generalization

Domain transfer learning [144] is a technique used to learn robust features that generalize from the training samples (*source* domains) to out-of-distribution test data (*target* domains). The learning process usually involves two main steps [140]:

1. *Pre-training*: the model is trained on an upstream task using a large dataset, aiming to extract general patterns and robust features from the input data.
2. Knowledge transfer, or *adaptation*: the model is fine-tuned on downstream target domains. In this step, depending on the access to the target domain, it is possible to distinguish between *Domain Adaptation* (DA) [145], which uses both labeled source data and a subset of unlabeled target data, and *Domain Generalization* (DG) [146, 147], that only leverages data from source domains.

Farahani *et al.* [145] categorize DA techniques into methods with shallow and deep architectures. Belong to the former approaches that aim to minimize the distance between source and target domains [148–150], using metrics like the Wasserstein distance and Kullback-Leibler divergence [151]. Methods belonging to the latter approach levearge DNNs instead. Models like adversarial or convolutional networks, and autoencoders are used to reduce the domain gap [152, 153].

Since its conceptualization in 2011, DG has received much attention for the research community, as demonstrated by the significant amount of related works [147]. The DG literature has explored various approaches, such as aligning source domain distributions to learn domain-agnostic representations [154], or using meta-learning to expose the model to the domain shifts at training time [155], or creating synthetic data to bridge the gap between the source and potential target domains [31]. This thesis focuses on domain generalization.

2.4.2 Momentum Improves Generalization

Stochastic Gradient Descent with *momentum* [156] is one of the most widely used optimizers in deep learning literature [157]. Momentum can be likened to a rolling ball descending a hill: as it gets closer to the lowest point, it gains speed. Similarly, in SGD with momentum, the optimization process accelerates as it approaches the minimum. This results in gradients being accelerated towards the lowest-loss points of the loss landscape, leading to faster convergence and dampened oscillations.

Formally, momentum is defined as a moving average of the gradients, and modifies the update rule of SGD (Algorithm 1.Line 4) at iteration i as

$$\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - \boldsymbol{v}_i, \tag{2.26}$$

$$\boldsymbol{v}_i \leftarrow \beta \boldsymbol{v}_{i-1} + \eta_i \nabla_{\boldsymbol{w}} \hat{\mathscr{L}}(\boldsymbol{w}), \tag{2.27}$$

with $\beta \in \mathbb{R}^+$ being a hyperparameter, usually in $[0.5, 0.99]$. The term $\boldsymbol{v}$ is the *velocity* and allows the algorithm to maintain a memory of previous updates, avoiding local minima [158].

Several works have examined the impact of momentum on generalization [157–162]. Among these, Jelassi *et al.* [162] emphasize that the key strength of momentum in enhancing generalization lies in its ability to amplify historical gradients.

2.4.3 On the Impact of Data Distribution

The data distribution plays a crucial role in the generalization capabilities of deep learning models. When training data is representative of the overall data distribution, models are more likely to generalize well to unseen data. However, in real-world scenarios, achieving a balanced distribution is often unrealistic, as certain subjects may appear more frequently than others. This imbalance can lead to models that overfit to the more prevalent classes and underperform on less common ones, highlighting the importance of addressing data distribution challenges to improve generalization.

Imbalanced data

Class imbalance occurs in supervised learning problems when a subset of classes is underrepresented compared to the others. High class imbalance is characteristic of various real-world scenarios, such as cancer or fraud detection [163, 164, 142], and federated learning, where each user independently collects data based on personal habits [34]. In many cases [83, 165, 163, 164], the minority class is the primary focus of the learning process. When data distribution is skewed towards certain classes, models often fail to capture relevant information about the minority group and may overfit the majority one due to its increased prior probability.

According to R. Anand *et al.* [166], the gradient of the model parameters is dominated by the majority class when training DNNs on imbalanced datasets. Consequently, the training error for over-represented classes rapidly decreases, while the error for the minority class likely increases, resulting in slow convergence [142]. Possible solutions include balancing the data [167], using sophisticated sampling methods [168], or dividing training in two phases [169].

Domain shifts

The unbalanced distribution of data is not limited to class representation. In computer vision, data is also characterized by the presence of visual *domains*. The *domain shift* refers to variations in data distribution across different environments or conditions. For instance, images captured outdoors can vary significantly in terms of lighting, reflections, viewpoints, and meteorological conditions.

Formally, given a probability distribution P and two sample pairs drawn from the training dataset $\mathscr{D}$, namely (x_i, y_i) and (x_j, y_j) with $i \neq j$, the presence of domain shifts is formalized as a difference in the features distribution: $P(y_i|x_i) = P(y_j|x_j)$ and $P(y_i) = P(y_j)$, but $P(x_i|y_i) \neq P(x_j|y_j)$ and $P(x_i) \neq P(x_j)$. This means that while the class probabilities are shared, the feature distributions vary. For example, x_i and x_j could be pictures of the same location taken at different times of the day.

The presence of domains highly impacts the generalization capabilities of deep learning models. When a model trained on a specific domain (*source*) encounters data from a different domain (*target*), performance can degrade due to discrepancies in features [170–173], affecting the model robustness. This phenomenon often occurs in real-world applications like autonomous driving and medical imaging [174, 175].

Section 2.4.4 details some approaches to generalize to multiple domains.

Data Augmentation

Data augmentation is a widely used technique in deep learning to enhance the generalization capabilities of models [176–178]. By artificially expanding the training dataset through various transformations such as rotations, translations, scaling, flipping, and color adjustments [179], data augmentation helps create a more diverse set of training examples. This diversity encourages the model to learn more robust and invariant features, reducing its tendency to overfit to the limited original dataset. Consequently, models trained with data augmentation perform better on unseen data, leading to improved generalization and higher accuracy in real-world applications.

In addition to these standard techniques, advanced data augmentations exist, such as Cutout [180] and MixUp [181]. Cutout regularizes learning by randomly masking out square regions of the input during training. MixUp instead trains the neural network using convex combinations of images and their labels, leveraging the prior knowledge that linear interpolations of features result in linear interpolations of their corresponding targets. Given two input images $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ and their corresponding one-hot label encodings $\{y_i, y_j\}$, virtual training examples are constructed as follows:

$$\bar{x} = \beta x_i + (1-\beta)x_j \text{ and } \bar{y} = \beta y_i + (1-\beta)y_j, \tag{2.28}$$

with $\beta \sim \text{Beta}(\gamma, \gamma)$ for $\gamma \in (0, \infty)$.

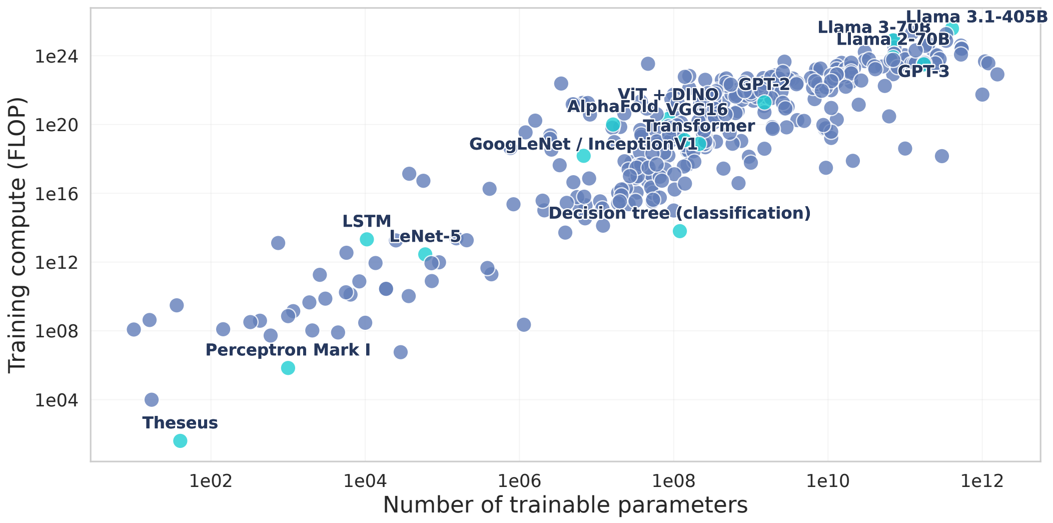


Fig. 2.6 Number of models parameters *vs*. corresponding training computation in notable AI systems. Computation is measured in floating point operations. The highlighted points correspond to the reported models' names. Data source: [5, 3].

2.4.4 The Critical Choice of the Model

This section explores the critical role of model choice in achieving strong generalization performance. Beyond the initial position of a model in the loss landscape, the very architecture of the model itself significantly impacts its ability to generalize. This thesis investigates three key factors related to model choice that influence generalization: model complexity, the design of the architecture's layers, and the utilization of pre-training.

Model Size

While overfitting is a concern for complex models with limited training data (as discussed in Section 2.1.1), the abundance of data in real-world scenarios offers a unique opportunity. Large datasets, often containing diverse distributions and intricate patterns, can mitigate the risk of overfitting even with complex models [59]. This is especially true when dealing with unsupervised scenarios, where data is readily available due to the absence of labeling requirements (*e.g.*, domain expertise) and related costs [19].

However, the allure of simply deploying the largest model needs to be tempered by practical considerations. Training large deep learning models incurs significant costs [182, 183]. As Figure 1.4 shows, the training cost of notable AI systems went from less than $1,000$ to over 10 millions dollars in just over a decade. In addition, larger models training necessitates access to substantial computational resources, such as GPUs or TPUs, and ample storage memory (see Figure 2.6). Furthermore, real-world applications often involve deploying these models on resource-constrained devices at the edge, like IoT sensors or smartphones [184]. Large model sizes can hinder deployment on such devices, and network transmission can be subject to congestion, overload, or even unavailability.

Therefore, the optimal model choice for generalization becomes a balancing act. While complex models may excel on large datasets, real-world constraints like resource limitations necessitate a focus on model efficiency [185]. This thesis explores strategies to achieve strong generalization performance while keeping model complexity in check for practical federated applications.

Impact of Layers on Generalization

Aside from the complexity of the model architecture, its layers play an important role as well. In particular, Batch Normalization (BN) layers and the classifier have a huge impact on the model generalization performance.

BN works by normalizing the inputs of each layer to have a mean of zero and a variance of one during training (Section 2.2.1) and is based over the assumption that training and test data follow the same distribution. This normalization process reduces internal covariate shift, allowing the model to learn more stable and reliable features, alleviating the need for very high learning rates. This can lead to faster convergence and improved generalization by reducing the sensitivity of the model to the specific scaling of its internal activations. Furthermore, BN can help alleviate the problem of vanishing or exploding gradients (Section 2.3.2), which can hinder training in deep networks, and acts as a regularizer, reducing the need for other forms of regularization like dropout [186]. Lastly, Santurkar *et al.* [134] note that the presence of BN layers makes the loss landscape significantly smoother, further stabilizing the learning process. These characteristics reflect in better generalization to unseen data in various tasks [187, 188, 133, 189–194].

Research suggests that the model's classifier layer is most impacted by shifts in data distribution [195–197] and by spurious correlations [198], *i.e.*, patterns that appear predictive in the training data but are misleading at test time. For instance, in classifying waterbirds versus landbirds, if waterbirds often appear with water in the background, the model might classify any bird with water as a waterbird, ignoring the actual bird features. Kirichenko *et al.* [199] demonstrate that retraining the last layer on a fraction of held-out balanced data can achieve equal performance across majority and minority groups. Building upon this work, Izmailov *et al.* [200] highlight that the learned feature representation itself is influenced by model architecture and pre-training methods. Luo *et al.* [201] instead show that the classifier is highly impacted by the heterogeneous distributions of clients in federated learning.

This thesis investigates the impact of both BN and output layers on learning a model in the federated learning scenario.

Impact of Pre-training

The initial values assigned to a network's parameters significantly impact its optimization path [139, 131]. A common approach to achieve a more favorable starting point is to use pre-trained parameters [140]. These models have been trained on a separate large dataset, and their learned parameters can be utilized to initialize a new network in an already favorable initial point in the optimization landscape.

In addition, this approach leverages the knowledge gained from the pre-trained model, potentially leading to better generalization on new tasks. The positive impact of the knowledge transfer is highly dependent on the quality of the learned representations. Nowadays, examples of strong backbone architectures are ResNets [18], EfficientNet [202], ConvNeXt [203] and Vision Transformers (ViT) [17].

2.4.5 Generalization through the Lens of the Loss Landscape

The loss landscape is a multidimensional surface that represents the relationship between the model's parameters and its loss function. The geometry of the loss surface is commonly described by the existence of both *sharp* and *flat* minima (Figure 2.7). Intuitively, sharp minima are characterized by steep gradients and narrow regions of low loss, where the function quickly increases as the parameters move away from the minimum. In contrast, flat minima exhibit gentle slopes and wider basins of attraction with a gradual increase in loss. Hochreiter *et al.* [204]

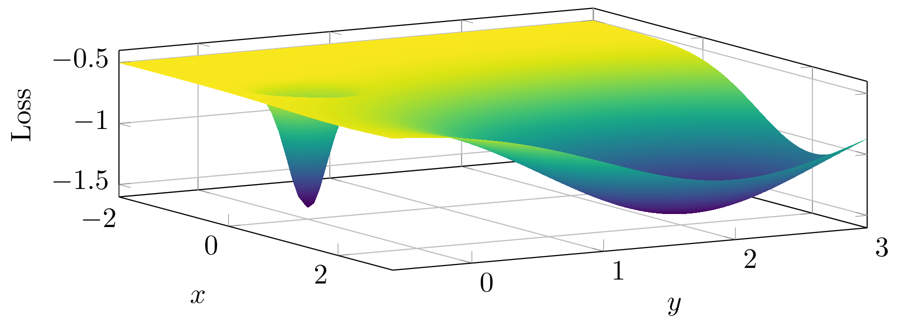


Fig. 2.7 Example of visualization of sharp and flat minima in a high-dimensional non-convex loss surface. The sharp minimum (*left*) is characterized by steep gradients, making it sensitive to small changes in the parameters, while the flat minimum (*right*) has shallow gradients, allowing for more flexibility in the parameter space.

defined the *flatness* of a solution as the dimension of the region connected around the minimum in which the training loss is low. This concept was later refined by Keskar *et al.* [52], who introduced the ϵ-sharpness which considers the maximum loss value within a given neighborhood of the minimum.

To understand the generalization ability of DNNs, several theoretical and empirical studies analyze its relationship with the geometry of the loss surface [205, 204, 52, 206–212], linking sharp minima with poor generalization. In particular, it has been shown that sharpness-based measures highly correlate with generalization performance [209]. Among these studies, special attention has been given to Sharpness-Aware Minimization (SAM) [213], an optimizer that explicitly seeks flatter regions and smoother loss surfaces through a simultaneous minimization of loss sharpness and value at training time. This method has proved effective across a variety of architectures and tasks, such as Vision Transformers [214] and language modeling [215] respectively.

As discussed in Section 2.3, the Hessian matrix, computed w.r.t. the model parameters, provides information about the curvature of the loss landscape, with flat minima corresponding to low-curvature regions. Building on this connection, Keskar *et al.* [52] proposed using the maximum eigenvalue λ_1 of the Hessian matrix as a proxy for sharpness: a lower λ_1 indicates a flatter minimum and potentially better generalization performance.

Sharpness-aware Minimization

SAM [213] aims to jointly minimize the loss value and the sharpness of the solution by solving the following min-max problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ F(\boldsymbol{w}) \triangleq \max_{\|\boldsymbol{\epsilon}\|\leq\rho} f(\boldsymbol{w}+\boldsymbol{\epsilon}) \right\}, \tag{2.29}$$

where $\boldsymbol{\epsilon}$ is the perturbation to estimate the sharpness, f the empirical risk, ρ the neighborhood size and $\|\cdot\|$ the ℓ_2 norm.

From Equation (2.29), given the point $\boldsymbol{w}$ in the parameter space, the **sharpness** of f is defined as

$$\mathscr{S}_{\boldsymbol{w}} \triangleq \max_{\|\boldsymbol{\epsilon}\|\leq\rho} f(\boldsymbol{w}+\boldsymbol{\epsilon}) - f(\boldsymbol{w}) \tag{2.30}$$

and measures how quickly the loss can be increased by moving from $\boldsymbol{w}$ to a nearby parameter. Using the first-order Taylor expansion of f, SAM efficiently solves the inner maximization in Equation (2.29) as

$$\arg\max_{\|\boldsymbol{\epsilon}\|\leq\rho} \left\{ f(\boldsymbol{w}) + \boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{w}} f(\boldsymbol{w}) \right\} = \rho \frac{\nabla_{\boldsymbol{w}} f(\boldsymbol{w})}{\|\nabla_{\boldsymbol{w}} f(\boldsymbol{w})\|} \triangleq \hat{\boldsymbol{\epsilon}}(\boldsymbol{w}). \tag{2.31}$$

Thus, $\hat{\boldsymbol{\epsilon}}$ is the scaled gradient of the loss w.r.t. the current parameters $\boldsymbol{w}$. The sharpness-aware gradient is then defined as $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$. Equation (2.29) is solved with a first gradient ascent step to compute $\hat{\boldsymbol{\epsilon}}$ and a descent step using the sharpness-aware gradient, updating the model as $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} f(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$.

Building Upon SAM: Recent Advancements

Since its publication in 2021, SAM has received a lot of attention from the ML community. Follow-up works mainly focus on improving SAM's efficiency or its performance, and on better understanding its behavior.

Efficient Sharpness-aware Minimization. SAM's optimization process involves a step of gradient ascent to approximate the sharpness within a given neighborhood and one step of gradient descent for joint loss and sharpness minimization, resulting in increased computation cost. To reduce SAM's doubled computational overhead, various methods have been proposed. ESAM [216] reduces the model parameters or the batch of data used to computed the perturbation, while δ-SAM [217] substitutes per-instance weight perturbations with per-batch approximations by reweighing the contribution of each instance. LOOKSAM [218] computes the perturbation only once every k steps, building upon the insight that the norm of the gradient pointing towards the flatness direction does not significantly change during training. DP-SAT [219] approximates the adversarial step with the gradient from the previous iteration in the context of differential privacy [220] and SAF [221] replaces SAM's sharpness approximation with the trajectory of weights learned during training. In K-SAM [222], both gradients are computed using only the k samples having highest loss, exploiting the idea that the examples with larger losses dominate the average gradient over a large batch. Lastly, SAM-ON [223] argues that perturbing only the normalization parameters in the ascent step outperforms using the whole model.

Performance-driven Sharpness-aware Minimization. A significant line of research focuses on improving SAM's performance. Among these methods, GSAM [224] shows that SAM's perturbed loss can be found in sharp minima. The *surrogate gap* is proposed as an alternative measure that agrees with sharpness. GSAM jointly minimizes the perturbed loss and the surrogate gap. IMBSAM [225] introduces a novel version of SAM for addressing class imbalance, while in WSAM [226] sharpness is used as a regularization term and a hyperparameter weights the importance of the perturbed gradient. SSAM [227] leverages the smaller parameters norm resulting from SAM's gradients to promote sparsity.

Understanding SAM. Kwon *et al.* [228] highlight that SAM is sensitive to parameter re-scaling, weakening the connection between loss sharpness and generalization gap. ASAM (Adaptive Sharpness-Aware Minimization) [228] solves such issue by introducing the concept of adaptive sharpness: ASAM reformulates sharpness to be invariant to weight scaling by conditioning the neighborhood region of each weight based on its magnitude. SAMSON [229] builds upon the concept of ASAM by incorporating the distribution of weight values within the loss function. While ASAM only considers the magnitude of individual weights, SAMSON additionally factors in the dynamic range of the weight distribution. This combined approach leads to normalized neighborhood sizes across all layers of the deep learning model. Furthermore, SAMSON focuses on outlier weights during the adaptation process, aiming to improve the model's generalization performance and robustness. Andriushchenko *et al.* [230] show how SAM mainly impacts training during the initial stages, and in the follow-up work [231] argue that SAM leads to low-rank features by implicitly pruning a significant number of activations. Dai *et al.* [232] explain that the normalization factor in SAM increases stability and robustness towards different values of ρ, and allows the algorithm to keep exploring the manifold of minimizers instead of being trapped in one minimum.

The Duality between SAM and Adversarial Perturbations

The robust formulation of deep learning models is often conceptualized as a two-player game between an adversary and the model. The adversary perturbs the input data to maximize the loss of the model, using perturbations that are usually constrained to a certain norm (*e.g.*, ℓ_p-norm) to ensure they remain small and imperceptible. The model (or defender) seeks to minimize the loss function over the perturbed inputs, maintaining good performance even in the presence of adversarial attacks. This two-player game can be expressed as a min-max optimization problem

$$\min_{\boldsymbol{w}} \max_{\|\delta\|_p \leq \rho_{\text{adv}}} \mathscr{L}(f_{\boldsymbol{w}}(\boldsymbol{x}+\delta), y), \tag{2.32}$$

where δ is the adversarial perturbation (AT), $\boldsymbol{x}$ the input data and y its true label, $\mathscr{L}$ the loss function and ρ_{adv} the margin of perturbation. An intuitive interpretation is that the adversary generates challenging inputs within a constrained region aiming to maximize the loss, while the model learns parameters that are robust to such challenges, improving generalization to both clean and adversarial examples, ultimately minimizing Equation (2.32).

Similarly, SAM can also be interpreted as a two-player game between a model and a "perturbator", as discussed in [233]. The first player represents the model parameters $\boldsymbol{w}$ and seeks to minimize the maximum loss caused by small perturbations to its parameters, aiming for a robust and generalizable solution. The second player represents adversarial perturbations (*i.e.*, $\boldsymbol{\epsilon}$) applied to the model parameters. The perturbator's objective is to find a parameter perturbation $\boldsymbol{\epsilon}$ that maximizes the training loss, effectively identifying sharp regions in the loss landscape. When solving SAM's min-max optimization problem (Equation (2.29)), the perturbator seeks the steepest direction (sharpest region) that increases loss, while the model adapts its parameters to avoid such sharp regions, effectively "flattening" the loss landscape and ensuring better generalization.

For clarity, both objectives are rewritten here in a unified form:

$$\text{SAM:}\ \min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}} \max_{\|\boldsymbol{\epsilon}\|_p \leq \rho} \mathscr{L}(f_{\boldsymbol{w}+\boldsymbol{\epsilon}}(\boldsymbol{x}), y) \tag{2.33}$$

$$\text{AT:}\ \min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}} \max_{\|\delta\|_p \leq \rho_{\text{adv}}} \mathscr{L}(f_{\boldsymbol{w}}(\boldsymbol{x}+\delta), y). \tag{2.34}$$

While both techniques use perturbation to learn a function f more robust to input or weight changes, **AT adds such perturbations to the input samples and transforms them into adversary examples, while SAM directly perturbs the weights**. As a result, the data distribution learned in the adversarial setting diverges from the original one, causing an inevitable loss of generalization when the model is tested on the natural distribution. In contrast, SAM's weight perturbation allows it to learn from the actual data distribution while implicitly weighing robust features more [234, 233].

Model Ensembling and Model Averaging

Model ensembling [235] is a successful technique to enhance the generalization of deep learning models by combining the predictions of multiple models. The underlying idea is that the ensemble generalization ability is usually stronger than that of a single learner. Different models, even when trained on the same data, may capture different patterns or make distinct types of errors. By aggregating their predictions, the ensemble can mitigate the effects of individual model biases and errors and become more robust.

Formally, given an input sample $\boldsymbol{x}_i$, model ensembling leverages an aggregation function G to combine a set of h classifiers $f_1, f_2, \dots, f_h$ into a single output as

$$y_i = \phi(\boldsymbol{x}_i) = G(f_1, f_2, \dots, f_h), \tag{2.35}$$

where all $f_i \forall i \in [h]$ were trained on the same input data, or on its subsets. The resulting y_i is the aggregate prediction. Common implementation of G include averaging, bagging, or major voting [235].

Garipov *et al.* [236] show that randomly initialized networks independently trained on the same task are connected by simple curves of low loss. Building on this insight, the authors propose FGE (Fast Geometric Ensembling), that collects models during training and averages their predictions. **Stochastic Weight Averaging** (SWA) [237] argues that solutions found by FGE are on the edge of most desirable ones. Thus, it propose to average points *in the weight space* rather than in the model space to move solutions towards the center of the minimum. In particular, SWA computes a moving average of the weights of models sampled at different points along SGD's training trajectory, while traversing the weight space using a cycling learning rate for wider exploration. At each step i of a cycle of length c, the learning rate decreases from γ_1 to γ_2 as

$$\gamma(i) = (1 - t(i))\,\gamma_1 + t(i)\gamma_2, \tag{2.36}$$

$$t(i) = \frac{1}{c}\,(mod(i-1,c)+1)\,. \tag{2.37}$$

The learning rate scheduling is constant for $c = 1$ and cyclical for $c > 1$. Starting from a pre-trained model $f_{\boldsymbol{w}}$, SWA captures all the updates $\boldsymbol{w}$ at the end of each cycle and averages them as

$$\boldsymbol{w}_{\text{SWA}} \leftarrow \frac{\boldsymbol{w}_{\text{SWA}} \cdot n_{\text{models}} + \boldsymbol{w}}{n_{\text{models}} + 1} \tag{2.38}$$

obtaining the final model $f_{\boldsymbol{w}_{\text{SWA}}}$, with $n_{\text{models}} \in \mathbb{N}$ counting the number of completed cycles. This approach creates a smoother and more robust solution space, leading to better performance on unseen data. However, SWA is most effective near convergence, *e.g.*, after 75% of training.

SWAD [238] leverages SWA for domain generalization. Kaddour *et al.* [239] compare SWA and SAM, showing that the best outcome is obtained when combining the two approaches, *i.e.*, by averaging SAM's models as in Equation (2.38). Aiming to reduce training hours of large models, **LAWA** (Latest Weight Averaging) [240] introduces a sliding window traversing the training iterations, containing the last W checkpoints. At the end of each epoch, LAWA averages the queued parameters, resulting in a robust model and improved convergence speed. Another successful example of model averaging is Model Soup [241], that combines the weights of different models trained on the same task but with varying hyperparameters or architectures to gain robustness.

2.5 Visualization of the Loss Landscape

This section describes the procedure to visualize the loss landscapes and compute the Hessian eigenvalues as a proxy for sharpness.

Visualizing 1D Loss Landscapes. The 1D loss landscape visualization is often used to highlight the presence of loss barriers between two interpolated models, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Given the interpolation coefficient $\gamma \in \mathbb{R}$

$$\boldsymbol{w} = \gamma \cdot \boldsymbol{w}_1 + (1-\gamma) \cdot \boldsymbol{w}_2 \,. \tag{2.39}$$

For each $\gamma \in [a,b]$, with a and $b \in \mathbb{R}$, the resulting $\boldsymbol{w}$ is tested on the dataset of interest. If no loss barrier appears, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ lie in the same basin.

Visualizing 2D Loss Landscapes. Following the methods outlined in [236] and [242], the procedure for visualizing 2D loss landscapes involves the following steps:

1. Three weight vectors, denoted by $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$, are chosen. Two basis vectors, $\vec{u}$ and $\vec{v}$, are constructed using these weight vectors: $\vec{u} = (\boldsymbol{w}_2 - \boldsymbol{w}_1)$ and $\vec{v} = (\boldsymbol{w}_3 - \boldsymbol{w}_1) - \frac{\langle \boldsymbol{w}_3 - \boldsymbol{w}_1, \boldsymbol{w}_2 - \boldsymbol{w}_1 \rangle}{||\boldsymbol{w}_2 - \boldsymbol{w}_1||^2} \cdot (\boldsymbol{w}_2 - \boldsymbol{w}_1)$.

2. These basis vectors are then normalized to form an orthonormal basis in the plain containing $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$, as $\hat{u} = {}^{u}/_{||u||}$ and $\hat{v} = {}^{v}/_{||v||}$.

3. A Cartesian grid with dimensions $N \times N$ points is defined within the basis formed by $\hat{u}, \hat{v}$. In this case, a grid size of $N = 21$ is used.

4. For each point on the grid, the corresponding weight vector is calculated and the resulting network's loss is evaluated. Given a point P on the grid with coordinates (x,y), the corresponding weight vector is computed as: $P = \boldsymbol{w}_1 + x \cdot \hat{u} + y \cdot \hat{v}$. Here, $\boldsymbol{w}_1$ serves as the reference point and is located at the origin $(0,0)$ of the grid.

The code for this process was adapted from the implementation provided by Garipov *et al.* [236][1].

[1]https://github.com/timgaripov/dnn-mode-connectivity

Visualizing 3D Loss Landscapes. The proposed plots of loss landscapes are computed leveraging the techniques from [243]. Their code is adapted to the introduced datasets and network architectures. The process involves calculating the loss function along random directions in the parameter space. This is achieved by perturbing the model's parameters within a defined range. In this case, the perturbations are constrained in the range of $[-1, 1]$ for both the x and y axes. To ensure consistent comparisons across models, the same set of random directions is used for all models.

Hessian Eigenvalues for Flatness Measure. Following prior work [213, 236, 243], the spectrum of the Hessian matrix is used to quantify the *flatness* of the loss landscape. Lower maximum eigenvalues correspond to flatter landscapes, implying less sharpness. Stochastic power iteration [244] is used to compute the largest eigenvalues (denoted by λ_1), with a maximum of 20 iterations, referring to the code of [245].

Chapter 3

Federated Learning

Privacy is not an option, and it should not be the price
we accept for just getting on the internet

GARY KOVACS

This chapter elaborates on the Federated Learning framework and its various configurations (Section 3.1), the learning objective (Section 3.2) and the challenges encountered in real-world deployments, with an emphasis on vision applications (Section 3.3). Lastly, Section 3.4 presents the federated datasets used in the empirical evaluations conducted in this thesis.

3.1 Federated Framework

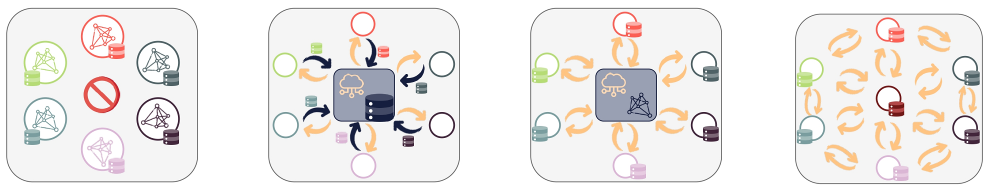
Federated Learning [34, 246] is a machine learning framework that enables collaborative training of a shared global model among multiple clients (*i.e.*, edge nodes, such as smartphones, or hospitals) on their privacy-protected data (*e.g.*, text messages, or patients' medical results), without compromising any sensitive information. Differently from centralized settings, where raw data needs to be transferred to a central location, FL happens through the exchange of model parameters and the clients' data never leaves their premises. The training process unfolds across iterative communication rounds. In each round, clients fit the model to their local data and then share the updated parameters to facilitate knowledge exchange. The global model results from the aggregation of the trained parameters.

Privacy in FL is achieved by adhering to the *Principle of Data Minimization* [35–37]: only the data necessary for a specific computation is collected, *i.e.*, the model parameters (*focused collection*); the user's information is processed immediately upon receipt (*early aggregation*) and then discarded (*minimal retention*). In addition, users are informed and give their *consent* regarding the use of their data, ensuring *transparency*. Lastly, FL aims to uphold the *Principle of Data Anonymization*, where the model's final output does not disclose any sensitive information and aggregate statistics (*e.g.*, model parameters) do not significantly vary based on any individual user's data [38].

The following sections first outline the difference between FL and centralized learning (Section 3.1.1), and then explore the specifics of federated settings, considering the underlying framework architecture (Section 3.1.2), the nature of the local datasets (Section 3.1.3), and the characteristics of the clients (Section 3.1.4).

3.1.1 Federated *vs*. Centralized Learning

Federated and centralized learning differ primarily in how data is managed and processed. In centralized learning, data from various sources is collected and stored in a central server, where a global model is trained using this aggregated data. This approach, while straightforward, raises significant privacy and security concerns when sensitive data is transferred and stored in a single location. Additionally, the centralized storage location can become a single point of failure and is vulnerable to



(a) Local learning (b) Centralized learning (c) Centralized FL (d) Distributed FL

Fig. 3.1 Comparison of learning scenarios. Colors indicate different nodes with their own local data distribution. *(a)* **Local learning**: each node trains a model using only its data. *(b)* **Centralized learning**: the data is transferred (*blue arrows*) from the edge nodes to a central storage. The model is trained by accessing the overall data distribution and then sent back (*orange arrows*). *(c)* **Centralized federated learning**: the server shares the global model with the clients, which train it using their local datasets and send back the updated parameters. The server aggregates the updates. *(d)* **Distributed federated learning**: clients share model parameters, without the orchestration of a central server.

malicious attacks. In contrast, FL keeps the data localized on client devices. Each client trains a model on its own data and only shares the model updates rather than the raw data with a central server or other clients (see Section 3.1.2). This decentralized approach protects users' privacy by ensuring that sensitive information remains on local devices, and leverages edge computational resources, including memory for storage and CPU/GPU accelerators, which would otherwise remain underutilized. The FL approach introduces its own set of challenges in terms of coordination and communication, as discussed in Section 3.3. A potential alternative to both centralized and federated learning is to keep both data and training computation local, avoiding any additional communication. However, this method, known as *local learning*, restricts training to a limited number of data points, ultimately leading to overfitting [247]. Figures 3.1a to 3.1c summarize this comparison.

Lastly, it is important to note that the performance of a model trained in a centralized setting defines the **upper bound** for FL performance, as shown in [248, 7].

3.1.2 Centralized *vs*. Peer-to-Peer Federated Learning

Federated Learning can operate under either a client-server or a decentralized architecture, depending on the presence of a central orchestrating server [6].

In Centralized FL (CFL), a central server orchestrates the training process [34, 246]. Once the task to be solved is defined, the server initializes the global model using either pre-trained parameters or random ones [249–251]. During training, the server maintains an updated copy of the global model. At each round, it selects a subset of available clients to receive the model parameters and train them on their local data. The clients then send their updated models back to the server, which aggregates them according to a predefined rule. This process repeats for multiple rounds, continuously refining the global model. This approach ensures synchronization, implying that clients selected in the same round receive the same model, and all clients contribute to the same model. In addition, the server is assumed to be *honest-but-curious* [252], meaning it faithfully trains the federated model according to the principle of data minimization and does not maliciously alter the model architecture, send fake parameters to clients, or modify the FL algorithm. However, it could potentially use the legitimately received models to infer additional information, *e.g.*, statistical properties of the training data, or the users' geographical position. This approach also positions the server as the central player in the algorithm, making it a single point of failure. As discussed in [6, 253], while large companies or organizations can fulfill this role in certain application scenarios, a dependable and powerful central server may not always be readily available or desirable. Additionally, as the number of clients increases, the server can become a significant bottleneck [254].

An alternative to the client-server architecture is fully decentralized federated learning (DFL), where there are no central orchestrators, and communication occurs only between individual clients using a peer-to-peer (P2P) approach [255, 256, 247]. Relevant DFL frameworks are P2P FL [257], server-less FL [258] and device-to-device FL [259]. Communication between clients is typically modeled as a graph, with clients as nodes and edges representing communication channels. The underlying graph typically has a small maximum degree, allowing communication only between neighboring clients. Neighbors can be identified via geographical proximity, clients similarity, communication protocols, *etc*. [247]. In contrast, the client-server architecture can be thought of as a star graph, where the server communicates with all clients, while clients communicate solely with the server (Figures 3.1c and 3.1d). In DFL, each round involves exchanges between a client and its neighbors, with model updates occurring locally by aggregating both local and received parameters. Consequently, each client maintains a different version of the global model, *i.e.*, there is no longer a global state. Nonetheless, each local model gradually converges towards a consensus, ultimately reaching the desired solution. The primary advantages of DFL are its reduced communication costs, as it eliminates the need for intermediate exchanges with a central server [260]. However, unlike the CFL scenario, no assumptions about security and privacy can be made regarding the clients. In DFL, clients may be malicious and could attempt to compromise the privacy of other users by exploiting the received trained parameters, or they may try to poison the data or the updates sent [6]. Possible solutions are client-level differential privacy [261], achieved by adding random Gaussian noise to the model parameters, and secure aggregation protocols [262]. Another challenge is the practical applicability of DFL. Key questions include how to define the learning task without a central authority, and how to handle unreliable networks and clients.

This thesis focuses on the more-studied CFL. For a comprehensive review of the state of the art on DFL, the reader is referred to [6, 263, 247].

3.1.3 Horizontal *vs*. Vertical Federated Learning

As initially introduced by Yang *et al.* [264], FL can be categorized into *horizontal*, *vertical* and *transfer* learning based on **data partitioning**.

Given a training dataset $\mathscr{D} = (\mathscr{X}, \mathscr{Y}, \mathscr{F})$, where $\mathscr{X}$, $\mathscr{Y}$ and $\mathscr{F}$ are the input, output and feature spaces respectively, in **horizontal** or sample-based FL, devices share the same feature space but different samples, *i.e.*, $\mathscr{F}_i = \mathscr{F}_j$, $\mathscr{Y}_i = \mathscr{Y}_j$ but $\mathscr{X}_i \neq \mathscr{X}_j$, where i and j index the local datasets of two distinct clients. For example, two local hospitals likely have separate group of users, *i.e.*, different samples, but propose similar medical analyses, leading to the same feature space. Another example is Google's keyboard, GBoard [41], which provides next-word prediction to users worldwide. Horizontal FL is the most common scenario in state-of-the-art research on FL. By contrast, in **vertical** or feature-based FL, devices share the same sample space but different feature space, *i.e.*, $\mathscr{F}_i \neq \mathscr{F}_j$, $\mathscr{Y}_i \neq \mathscr{Y}_j$ and $\mathscr{X}_i = \mathscr{X}_j$ $\forall i, j$. For instance, a hospital and a bank in the same city likely share the same set of users but offer significantly different services, resulting in separate feature spaces. Lastly, federated **transfer** learning applies to scenarios where there is no overlap in either the sample or the feature space: $\mathscr{F}_i \neq \mathscr{F}_j$, $\mathscr{Y}_i \neq \mathscr{Y}_j$ and $\mathscr{X}_i \neq \mathscr{X}_j$ $\forall i, j$. A hospital in Italy and an e-commerce company in the U.S. likely do not share the same

Table 3.1 **Cross-silo *vs*. Cross-device FL *vs*. Distributed Learning**. Source: [6].

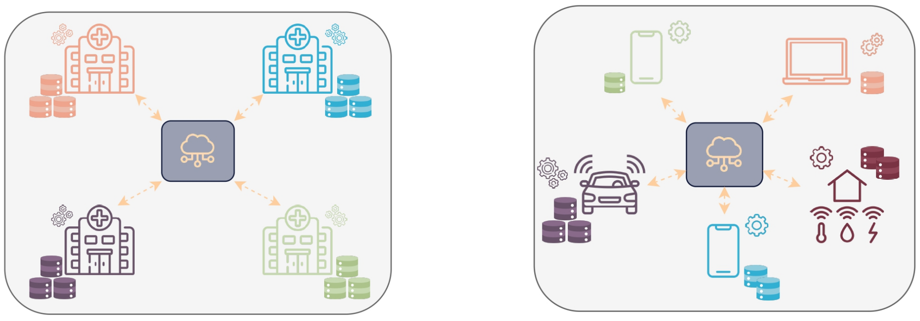
	Distributed Learning	**Cross-silo FL**	**Cross-device FL**
Clients	Compute nodes in a cluster or datacenter	Silos (*e.g.*, hospitals, companies, banks)	Edge devices (*e.g.*, smartphones, IoT sensors, autonomous vehicles)
Distribution scale	$\approx 1-1000$ clients	$\approx 2-100$ clients	Up to 10^{10} clients
Data distribution	Data centrally stored, arbitrarily shuffled among clients, usually *i.i.d.*	Data autonomously collected from users and likely **non-*i.i.d.***. Data remains decentralized and is subject to **privacy** constraints.	
Clients availability	Almost always available		Only a fraction of clients available. Influencing factors: geographical position, user habits.
Bottleneck	Computation	Communication or computation	Communication
Data partition	Arbitrary	Vertical or horizontal	Horizontal

set of users. However, there might be a small overlap in their feature spaces. In such cases, transfer learning techniques [144] can be employed to *transfer* knowledge from one domain to the other, facilitating the learning of a common representation.

3.1.4 Cross-silo *vs*. Cross-device Federated Learning

The final FL taxonomy pertains to the nature and number of clients involved in the training process, distinguishing between cross-silo and cross-device FL [6], summarized in Figure 3.2.

As the name suggests, in **cross-silo FL**, clients are large entities, or *silos*, such as hospitals, banks, and companies, that aim to learn a common model without sharing their data. For instance, hospitals might be interested in developing better diagnostic models [175], while banks might focus on enhancing fraud detection systems [265]. In general, a central server orchestrates the training process between approximately $2-100$ clients, *e.g.*, all hospitals within a single region, as summarized in Table 3.1. Silos are considered reliable and likely to be almost always available, allowing them to potentially participate in every training round. Given that silos typically possess powerful computational resources and stable communication networks, the main bottleneck can vary depending on various factors, such as network congestion, geographical locations, and weather conditions.



(a) Cross-silo FL (b) Cross-device FL

Fig. 3.2 Cross-silo *vs*. cross-device FL. *(a)* **Cross-silo FL**: training occurs between silos, such as hospitals, using their private data (*e.g.*, patient medical records). Each silo has its own data distribution, with typically large datasets and substantial computational resources. *(b)* **Cross-device FL**: clients are typically personal devices, such as smartphones, laptops, IoT sensors, or autonomous vehicles. They are characterized by data heterogeneity, varying and limited computational capacities, and skewed data quantities.

Cross-device FL, on the other hand, potentially involves billions of edge devices, including smartphones [41, 266, 44], IoT sensors [267, 268], and autonomous vehicles [269–272]. In this setting, clients may become unavailable due to various reasons, such as communication network instability, lack of internet access, battery power, time zone differences, and user habits. Consequently, only a fraction of clients can participate in each training round. Furthermore, although edge devices typically have limited computational resources, the primary bottleneck is communication, due to network instability and congestion.

In both cross-device and cross-silo FL, each client autonomously collects its own data, resulting in non-independent and identically distributed (non-*i.i.d.*) data that varies in terms of both distribution and quantity.

To better highlight the properties of federated settings, Table 3.1 also compares cross-silo and cross-device FL with distributed learning. The main differences lie in data distribution and access: in distributed learning, data is centrally stored and arbitrarily shuffled among the computing nodes. This means that the data distribution across clients is likely *i.i.d*. and, most importantly, the central orchestrator has access to the raw data. This contrasts with the key characteristics of FL, where data is naturally collected, typically non-*i.i.d.*, and remains inaccessible to external actors for privacy reasons.

This thesis focuses on cross-*device* FL, specifically aiming to develop a model that can generalize effectively when dealing with non-*i.i.d.* local data distributions.

3.2 Problem Statement

In FL, a central server communicates with a set of clients $\mathscr{C}$ across T rounds. The goal is to learn a global model $f(\boldsymbol{w}) : \mathscr{X} \to \mathscr{Y}$ parametrized by $\boldsymbol{w} \in \mathbb{R}^d$, where $\mathscr{X}$ and $\mathscr{Y}$ are the input and the output spaces respectively. Each client $k \in \mathscr{C}$ has access to a local dataset $\mathscr{D}_k$ containing N_k pairs of samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_k}$ with $\boldsymbol{x}_i \in \mathscr{X}$ and $y_i \in \mathscr{Y}$. In realistic heterogeneous settings, clients usually hold different data distributions, *i.e.*, $P_i \neq P_j$, and number of data points, *i.e.*, $N_i \neq N_j \forall i \neq j \in \mathscr{C}$.

The global FL objective is:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \left\{ f(\boldsymbol{w}) = \frac{1}{K} \sum_{k \in \mathscr{C}} f_k(\boldsymbol{w}) \right\}, \tag{3.1}$$

$$f_k(\boldsymbol{w}) \triangleq \mathbb{E}_{\xi_k \sim D_k} \ell(\boldsymbol{w}, \xi_k), \tag{3.2}$$

where $K \triangleq |\mathscr{C}|$ is the total number of clients, $f(\boldsymbol{w})$ is the empirical loss on the overall dataset $\mathscr{D} = \sum_k \mathscr{D}_k$, f_k is the empirical loss of the k-th client, ℓ is the loss function on the k-th client's data $\mathscr{D}_k$ (*e.g.*, cross-entropy loss, ℓ^{ce}) and $\xi_k = (\boldsymbol{x}_i, y_i)$ is the data sample randomly drawn from the local data distribution P_k.

The training process is a two-phase optimization approach within each round $t \in [T]$. First, due to potential client unavailability, a subset of selected clients $\mathscr{C}^t \subset \mathscr{C}$ trains the received global model parameters $\boldsymbol{w}^t$ using their local optimizer ClientOpt (*e.g.*, SGD). The local optimization happens for E epochs, using a local learning rate η_l, resulting in $\boldsymbol{w}_k^t$. Then, the server aggregates their updates with a server optimizer, ServerOpt (*e.g.*, SGD, Adam [273], AdaGrad [274]). Reddi *et al.* [275] propose FedOpt as a general framework to solve the FL optimization objective, formalized as

$$\Delta_{\boldsymbol{w}}^t \triangleq \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} (\boldsymbol{w}^t - \boldsymbol{w}_k^t), \tag{3.3}$$

$$\boldsymbol{w}^{t+1} \leftarrow \textsc{ServerOpt}(\boldsymbol{w}^t, \Delta_{\boldsymbol{w}}^t, \eta_g, t), \tag{3.4}$$

Algorithm 2 Federated Optimization (FedOpt)

```
Require: Number of clients $K$, number of communication rounds $T$, local mini-
    batch size $B$, number of local epochs $E$, local learning rate $\eta_l$, local optimizer
    ClientOpt, server optimizer ServerOpt, global learning rate $\eta_g$
 1: Initialize global model parameters $\boldsymbol{w}^0$
 2: for each round $t = 1, \ldots, T$ do
 3:     Randomly select a fraction of clients $\mathscr{C}^t \subseteq \mathscr{C}$
 4:     Send the current global model $\boldsymbol{w}^t$ to each client in $\mathscr{C}^t$
 5:     for each client $k \in \mathscr{C}^t$ in parallel do
 6:         $\mathbf{w}_k^t \leftarrow \mathbf{w}^t$                          ▷ Initialize local model
 7:         for each local epoch $e = 1, \ldots, E$ do
 8:             for each batch $b$ of size $B$ do
 9:                 Compute the local gradient $g_{b,k}^t = \nabla_{\boldsymbol{w}} \ell(\mathbf{w}_k^t; b)$
10:                 $\mathbf{w}_k^t \leftarrow$ ClientOpt$(\boldsymbol{w}_k^t, g_{b,k}^t, \eta_l, b)$      ▷ Client optimizer step
11:             end for
12:         end for
13:         Send updated local model $\mathbf{w}_k^t$ to the server
14:     end for
15:     $\Delta_{\boldsymbol{w}}^t = \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} (\boldsymbol{w}^t - \boldsymbol{w}_k^t)$          ▷ Compute global pseudo-gradient
16:     $\mathbf{w}^{t+1} \leftarrow$ ServerOpt$(\boldsymbol{w}^t, \Delta_{\mathbf{w}}^t, \eta_g, t)$           ▷ Server optimizer step
17: end for
18: Output: Global model parameters $\mathbf{w}^T$
```

where $\Delta_{\mathbf{w}}^t$ is the *global pseudo-gradient* at round t, $N = \sum_{k \in \mathscr{C}^t} N_k$ the total number of images seen during the current round and η_g the server learning rate. Algorithm 2 summarizes FedOpt.

The de-facto standard server-side optimization algorithm in FL is FedAvg [34], that computes $\boldsymbol{w}^{t+1}$ as a weighted average of the clients' updates, corresponding to one SGD step on the pseudo-gradient $\Delta_{\mathbf{w}}^t$ with learning rate $\eta_g = 1$:

$$\boldsymbol{w}^{t+1} \leftarrow \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} \boldsymbol{w}_k^t \tag{3.5}$$

$$= \boldsymbol{w}^t - \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} (\boldsymbol{w}^t - \boldsymbol{w}_k^t) \tag{3.6}$$

$$= \boldsymbol{w}_t - \Delta_{\boldsymbol{w}}^t. \tag{3.7}$$

FedAvg is summarized in Algorithm 3.

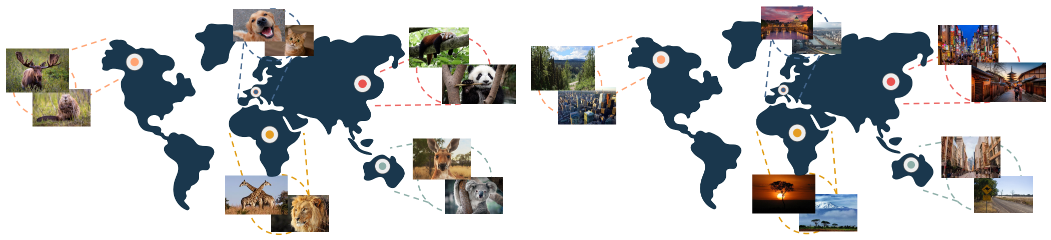
Algorithm 3 Federated Averaging (FedAvg)

```
Require: Number of clients $K$, number of communication rounds $T$, local minibatch
    size $B$, number of local epochs $E$, local learning rate $\eta_l$
 1: Initialize global model parameters $\boldsymbol{w}^0$
 2: for each round $t = 1, \dots, T$ do
 3:     Randomly select a fraction of clients $\mathscr{C}^t \subseteq \mathscr{C}$
 4:     Send the current global model $\boldsymbol{w}^t$ to each client in $\mathscr{C}^t$
 5:     for each client $k \in \mathscr{C}^t$ in parallel do
 6:         $\mathbf{w}_k^t \leftarrow \mathbf{w}^t$                                    ▷ Initialize local model
 7:         for each local epoch $e = 1, \dots, E$ do
 8:             for each batch $b$ of size $B$ do
 9:                 $\mathbf{w}_k^t \leftarrow \mathbf{w}_k^t - \eta_l \nabla_{\boldsymbol{w}} \ell(\mathbf{w}_k^t; b)$
10:             end for
11:         end for
12:         Send updated local model $\mathbf{w}_k^t$ to the server
13:     end for
14:     $\mathbf{w}^{t+1} \leftarrow \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} \mathbf{w}_k^t$                 ▷ Aggregate local models
15: end for
16: Output: Global model parameters $\mathbf{w}^T$
```

A Game-Theoretic Approach to Federated Learning. The FL problem (Equation (3.2)) can be seen as a game between multiple agents, *i.e.*, the clients, aiming to learn a model with low expected error on their own distribution. Instead of simply performing local learning, the subset of selected agents $\mathscr{C}^t$ forms a *coalition*, where their local estimates of the model parameters are aggregated together, *e.g.*, as in Equation (3.5). The goal of the coalition is to minimize the weighted sum of errors across players [276].

3.3 Challenges in the Real World: Literature Review

Deploying FL algorithms in real-world scenarios introduces a unique set of challenges. Key issues include managing non-*i.i.d.* data distribution and unbalanced local dataset sizes across clients, accounting for devices with varying and limited computational resources, ensuring efficient communication over potentially unreliable networks, and maintaining data privacy and security [277, 6]. Addressing these challenges is crucial for the successful deployment and scalability of FL systems in



(a) Animals around the world (b) Landscapes around the world

Fig. 3.3 Example of **heterogeneous data collection**. Local data distribution is significantly influenced by factors such as geographical location and user preferences. *(a)* Pictures of animals vary across the world. For instance, Australian users may take pictures of kangaroos and koalas, while moose may appear in photos taken in Canada. *(b)* Landscape photos can depict urban or natural scenery and may be captured at different times of the day (*e.g.*, nighttime *vs*. daytime), representing various *domains*.

diverse real-world applications. The following sections will explore each of these challenges in detail and discuss state-of-the-art solutions.

3.3.1 Statistical Heterogeneity

In FL, clients collect data autonomously, and this process is often influenced by factors such as user habits and geographical location [6, 278]. For instance, users may prefer photographing different subjects, at various times of the day, or in varying quantities (Figure 3.3). This introduces an inherent diversity of data distributions across clients, *i.e.*, local datasets are non-*i.i.d.* w.r.t. the underlying global distribution. This occurrence, known as *statistical heterogeneity*, poses a major challenge in FL scenarios.

Given two clients i and j in $\mathscr{C}$, their local data distributions $P_i(x_i, y_i)$ and $P_j(x_j, y_j)$ with $(x_i, y_i) \sim \mathscr{D}_i$ and $(x_j, y_j) \sim \mathscr{D}_j$ can be rewritten as $P(y|x)P(x)$ and $P(x|y)P(y)$ [6]. Depending on the varying factor, the following cases can be identified:

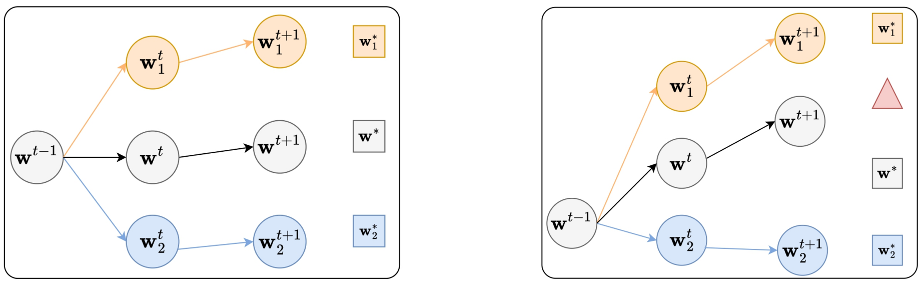
- **Label skew**: clients have access to different classes. $P(x|y)$ is shared, while $P_i(y)$ may differ from $P_j(y)$. For instance, data collection highly depends on geographical position (*e.g.*, kangaroos are only found in Australia, as shown in Figure 3.3a).

- **Feature shift**: clients hold different feature distributions. $P(x)$ may vary even if they share the same $P(y|x)$. For example, users writing the same word may have diverse handwriting styles.
- **Domain shift**: clients share the same classes but have different features, *i.e.*, $P(x|y)$ may vary even if $P(y)$ is shared. For instance, the concept of a "house" in the United States and in Japan is associated with very different buildings, while still having the same label. Other examples include shifts in weather or light conditions (Figure 3.3b).
- **Concept shift**: same features but different labels. The conditional distribution $P_i(y|x) \neq P_j(y|x)$, but $P_i(x) = P_j(x)$. Due to personal preferences, the same feature vectors can be mapped to distinct labels.
- **Quantity skew**: the sizes of local datasets differ, *i.e.*, $|D_i| \neq |D_j|$, with potentially $|D_i| \ggg |D_j|$ or vice versa.

Formally, each client $k \in \mathscr{C}$ solves the optimization problem $\arg\min_{\boldsymbol{w}_k} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}_k}[\ell(f(\boldsymbol{w}_k(\boldsymbol{x}),y)]$. This indicates that the clients' objectives vary significantly and depend heavily on their data distributions. In heterogeneous scenarios, the local data distributions are statistically different, causing local optimizations to converge towards different minima in the loss landscape [279, 45, 280]. This leads to a phenomenon known as ***client drift*** [279, 280], which quantifies the difference in expectation between the clients' convergence points and the global current one. Figure 3.4 compares the optimization paths in scenarios with and without client drift. Formally, at round t, the client drift is defined as

$$\varepsilon^t \triangleq \frac{1}{E \cdot K} \sum_{e=1}^{E} \sum_{k=1}^{K} \mathbb{E}[\|\, \boldsymbol{w}_{k,e}^t - \boldsymbol{w}^t \,\|], \tag{3.8}$$

where $\boldsymbol{w}_{k,e}^t$ is the k-th client local model after e epochs. Due to the client drift, the local optimization paths drift from the global convergence point, resulting in slow and unstable convergence [279, 281, 282, 280, 7, 283, 8]. The following sections discuss existing approaches to address this issue. For detailed surveys on methods for statistical heterogeneity in FL, the reader is referred to [284–287].



(a) *I.i.d.* setting

(b) Non-*i.i.d.* setting

Fig. 3.4 **Client drift** in *i.i.d.* *(a)* and non-*i.i.d.* settings *(b)*. Given a global model $\boldsymbol{w}^{t-1}$ (in grey) and two clients $k = 1$ and $k = 2$, at round t, the local updates $\boldsymbol{w}_k^t$ (in blue and orange) are aggregated to build the new global model $\boldsymbol{w}^t$, with $k \in \{1,2\}$. The same process occurs in the following round $t+1$. The convergence points are enclosed in squares. In *i.i.d.* settings (*left*), the global model correctly moves towards the global optimum $\boldsymbol{w}^*$ and no client drift occurs. In contrast, the non-*i.i.d.* data distribution (*right*) causes the global model to drift away from its optimum, converging instead at the points highlighted by the red triangle.

Client-side Approaches

Several approaches address client drift by directly enforcing regularization during local training. These include methods like FedProx [45], which introduces a term encouraging proximity between local and global parameters, and FEDDYN [288], employing the alternating direction method of multipliers (ADMM) [289] to align local and global convergence points. ADABEST [8] corrects local updates with an adaptive bias estimate. Differently, SCAFFOLD [280] leverage stochastic variance reduction to mitigate the client drift. Furthermore, some methods use momentum [157] to maintain a consistent global trajectory. This can be achieved either on the server side (*e.g.*, FEDAVGM [290]) or by incorporating global information into local training (*e.g.*, FEDCM [291], FEDACG [292], MIME [282]). FEDDC [283] instead maintains an auxiliary local variable to reduce the drift between local and global models.

A different line of works looks at statistical heterogeneity through the lens of Domain Generalization (FEDDG). Liu *et al.* [293] exploit frequency space to extract privacy-preserving information on the local distributions and learn a model robust to domain shifts in medical data. SILOBN [191] and FEDBN [193] learn domain-specific BN statistics. Cross-client style transfer [294] exchange the style across users to improve the model's robustness, and FEDKA [295] aligns feature distributions in the global space to learn domain-invariant features. A related research field is Federated Domain Adaptation (FEDDA). Differently from FEDDG, FEDDA has access to the target domain. Methods like [296, 297] align source and target features.

Server-side Approaches

Recent literature addresses the statistical heterogeneity issue also by looking at the server-side optimization. In terms of *model aggregation*, in addition to FEDOPT and FEDAVG introduced in Section 3.2, FAIRAVG [298] suggests updating Equation (3.5) by equally weighting all clients' updates, regardless of local dataset size, to ensure fair treatment of users. Conversely, a line of research demonstrates that incorporating the memory of previous updates through server momentum is effective in reducing bias towards recently observed clients [290, 282, 299], or in reducing the client drift by locally guiding the clients' updates [300, 291, 301, 302].

To improve the global model robustness, the literature additionally proposes to exploit *publicly available data* on the server side. Given that the trained local model under-represents patterns from missing classes, Zhao *et al.* [279] demonstrate that sharing a small set of public data among clients leads to notable improvements. Additionally, Li *et al.* [303] utilize public data for knowledge distillation, while Huang *et al.* [304] leverage unlabeled public data to learn a global representation invariant to domain shifts. Other studies [201, 305] suggest that the model's classifier layer is particularly affected by non-*i.i.d.* data distributions. Stabilizing this layer can significantly bridge the performance gap between non-*i.i.d.* and *i.i.d.* scenarios, improving overall model robustness and generalization.

As discussed in Section 3.2, FEDOPT demonstrates that FL is a two-level optimization process, with the "inner" training loop occurring on the client side and the "outer" loop on the server side. Building on this insight, works like [306–311] approach the FL task using Model Agnostic Meta-Learning (MAML) [312] techniques. Another line of research instead builds a parallelism between FL and Multi-Task Learning, seeing client with its own data distribution as a different task [313–316].

Another distinction can be made in terms of *orchestration* of the training process. For instance, FEDVC [278] creates virtual clients to reduce the impact of quantity skewness. In [317–320], the server clusters similar clients together to learn a cluster-specific model and reduce the impact of data heterogeneity. For examples, similarity can be expressed in terms of data distributions, available resources and geographical positions. Conversely, the *anti-clustering* [321] techniques aim to group together dissimilar elements. An example is FEDGSP [322], which dynamically expands the number of groups made of different clients in each round to enhance parallelism.

Personalized FL

Personalized Federated Learning (PFL) [309] argues that while standard FL enables knowledge sharing across multiple entities, the learned global model outputs the same features for all users and does not adapt to individual use cases. To overcome this limitation and maximize user experience, PFL aims to learn a common base model that each user can easily adapt to their specific task with minimal iterations, resulting in a personalized local model. PFL is applicable to a variety of tasks and settings. For instance, Apple and Google use PFL with their voice assistants - Siri and Hey Google respectively - to recognize individual users' voices [323]. Similarly, works such as [324] leverage PFL for speaker recognition. In [325], PFL is used for health monitoring tailored to individual user characteristics, and FEDHEALTH [326] proposes a framework for wearable healthcare. Detailed overviews of PFL can be found in [327, 328].

In contrast to PFL, this thesis aims to learn a *global* model capable of addressing the overall underlying distribution while ensuring local convergence.

3.3.2 System Heterogeneity

In realistic FL scenarios, edge devices exhibit a wide range of computational resources (*e.g.*, storage, processing power) due to *system heterogeneity* [44, 329]. In cross-device FL, such distinction is further underlined by limitations in battery life, varying device availability and *limited* resources. This heterogeneity impacts critical decisions like model complexity and local training volume, ultimately affecting convergence speed. Larger models, while offering superior performance, demand more resources for training, making them less suitable for resource-constrained edge devices. Consequently, FL in such settings often opts for smaller models, sacrificing some performance for efficient training. In addition, the limited availability of resources also restricts the training intensity. Local training on these devices is typically limited to a small number of epochs (often $E \leq 5$) to ensure efficient utilization of battery and processing power [45].

Adding to the challenge of limited resources is the restricted availability of edge devices. To avoid battery drain, training on these devices typically occurs only when they are plugged in and not in use. This often coincides with nighttime for personal devices like smartphones. As a consequence, client availability becomes dependent on the time zone, adding another layer of complexity to FL deployments in geographically distributed settings. Furthermore, the limitation of local training epochs due to resource constraints and the selection of only a subset of clients per round result in needing a larger number of communication rounds to achieve a target performance, ultimately slowing down convergence. Lastly, the varying clients' computational capabilities and availability results in different latency in reporting back to the server. These devices are referred to as *stragglers* [45]. To address these issues, FEDPROX [45] introduces a proximal term to account for device heterogeneity. FEDGKT [330] allows each client to define its own model architecture and uses knowledge transfer techniques to build a common server model. In Split-Mix FL [331], clients can customize a base sub-network to meet their specific needs.

3.3.3 Communication Efficiency

Communication is the main bottleneck in cross-device FL due to network congestion, unstable internet access, and potential client unavailability [6, 277, 332]. Consequently, significant attention has been given to minimizing both the communication bandwidth (*i.e.*, the number of bytes transmitted over the network) and the number of rounds necessary to reach target performance by improving convergence speed. However, as discussed in Section 2.1.1, although reducing the model size is a straightforward solution, it can lead to sub-optimal results (*e.g.*, underfitting). Therefore, this issue involves a trade-off between minimizing communication and achieving optimal performance, as theorized by [333–335].

State-of-the-art approaches often use sparsification [336, 337] and quantization [338–342] to reduce the size of model parameters to a smaller number of bits, with minimal impact on overall accuracy [246]. Alternatively, [343, 344] propose client selection techniques for improving training efficiency, and [345, 346] instead employ knowledge distillation.

Building upon existing works such as [280, 282, 288], this thesis aims to accelerate the convergence of FL algorithms to reduce communication costs.

3.3.4 Privacy Concerns

The federated framework involves multiple actors with distinct roles. Edge devices, acting as distributed data sources, contribute by collecting data and performing local model training. In contrast, the central server orchestrates the training process, aggregates these local updates and disseminates the consolidated knowledge back to participating devices, enabling both existing and newly joined users to benefit from the collective learning. Ideally, each actor would operate with a principle of least privilege, meaning they would limit their activities to acquiring and processing only the information necessary to fulfill the designated role. However, in realistic settings, potential vulnerabilities and threats to the system can arise [6, 277]. Ultimately, aiming to protect the users' privacy, it all comes down to answering the question: "How much can each involved party be trusted?".

Security Challenges in Federated Learning

A critical objective in FL is to guarantee the algorithm's resilience against malicious actors. However, current FL approaches have inherent vulnerabilities [347, 348]. One common threat is posed by *inference attacks*, where adversaries attempt to infer sensitive information from the model updates exchanged between clients and the central server. For instance, *model inversion attacks* aim to reconstruct the original data from the gradients or model parameters, as demonstrated by [349]. Hitaj *et al.* [350] leverage a Generative Adversarial Network (GAN) [351] to reconstruct user data, while [352, 353] use GANs for data reconstruction or inferring client characteristics. Similarly, *membership inference attacks*, highlighted in [354], seek to determine whether a specific data point was part of the training dataset, potentially exposing sensitive user information.

Additionally, *poisoning attacks*, as discussed by [355], involve adversaries injecting malicious data into the training process to corrupt the global model, hindering its ability to learn effectively, or embed hidden backdoors. Another concern is the risk of *gradient leakage* [356], where the gradients shared during the training process can inadvertently reveal private information. Malicious servers instead could implement *label and feature fishing* attacks by strategically altering global model parameters [357].

Defense Mechanisms

A first step in protecting the privacy of all involved parties is encryption, ensuring that only users with physical access to the device can retrieve the information exchanged over the network [358, 359].

Common defense techniques include *differential privacy* (DP) [360, 220, 361], which adds random Gaussian noise to the local model parameters before transmission, and fragmenting local updates before sending them to the server [362]. In scenarios where the server can be trusted, DP can be applied to the model before it is shared with the external world (*centralized differential privacy*). However, in typical settings, clients do not trust the server, and the noise added locally accumulates during aggregation, leading to a loss in performance.

Other approaches aim to detect malicious users, assuming that the main features from honest users follow a similar distribution [363], and excluding the furthest model from the mean in the server aggregation [364].

For a comprehensive discussion on FL threats, attacks and defenses please refer to [364, 365].

3.3.5 Federated Vision Applications

FL has attracted significant interest from the machine learning community since its introduction in 2017 [34]. However, its application to computer vision tasks has lagged behind the focus on algorithm optimization and convergence [281, 288, 275]. While small-scale image classification has been explored in various works [366], more complex tasks have been largely neglected. This disparity underscores the need to bridge the gap between theoretical advancements and real-world deployments of FL in computer vision.

The medical field presents a promising avenue for FL applications. Researchers are actively developing specialized techniques that prioritize strong patient privacy protections [367–370] while remaining robust to domain shifts [293, 371]. Examples of successful applications include disease classification [372, 373, 191, 374], MRI (Magnetic Resonance Imaging) reconstruction [375–377], and image segmentation for disease detection [293, 378, 366]. Notably, during the COVID-19 pandemic, FL techniques were successfully applied to chest X-ray and computed tomography scan analysis for virus detection [379, 267, 380].

One significant challenge for FL in computer vision is the lack of readily available benchmarks and large-scale federated datasets. Hsu *et al.* [278] addressed this by introducing federated versions of Google Landmarks v2 [381] and iNaturalist 2017 [382], namely LANDMARKS-USER-160K and INATURALIST-USER-120K respectively, enabling large-scale federated classification tasks. Luo *et al.* [383] introduced the Street-5 and Street-20 datasets for object detection, consisting of images taken from street cameras.

Recent advancements are trying to fill the gap in complex computer vision tasks. Researchers have proposed algorithms for deploying both global and personalized models for object detection [384], while works like FEDMARGIN [385], Federated Multi-target Domain Adaptation (FMTDA) [297] and FEDSEG [386] focus on semantic segmentation. In particular, FMTDA introduces the challenge of learning from clients with unlabeled local target datasets while leveraging a public labeled source dataset available on the server side. Frameworks like FEDVISION [46] further bridge the gap by providing platforms for deploying FL applications in various settings, *e.g.*, smart cities.

While face recognition has been explored in FL [387–389], ethical considerations and privacy regulations (*e.g.*, GDPR [35]) necessitate careful evaluation. Training data can potentially be reconstructed from the trained model (Section 3.3.4), raising privacy concerns, especially when dealing with sensitive data like facial images.

3.4 Datasets

This Section details the federated datasets used in the proposed experimental evaluations, distinguished by task, *i.e.*, image classification (Section 3.4.1), semantic segmentation (Section 3.4.2) and visual place recognition (Section 3.4.3).

3.4.1 Image Classification

Most of the experimental results presented in this thesis focus on image classification. Table 3.2 summarizes the key information about the main datasets, with details provided in the following sections.

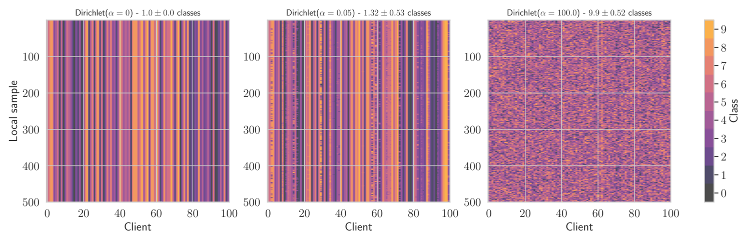
CIFAR Datasets

Task. CIFAR10 and CIFAR100 [390] are image classification datasets with 10 and 100 classes respectively. The labels include (but are not limited to) animals, vegetables, fruits, plants and vehicles.

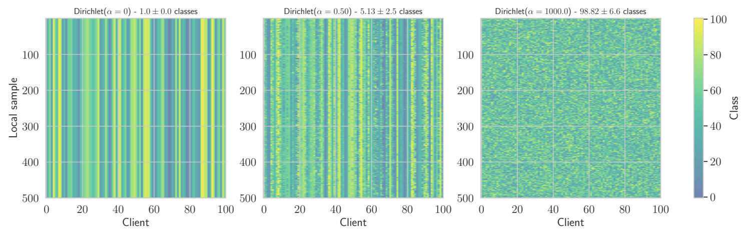
Centralized. Each dataset is made of $50,000$ training and $10,000$ test images, evenly distributed across the classes. CIFAR100 additionally groups the 100 classes into 20 *coarse* ones. For instance, the labels “sunflowers”, “roses” and “tulips” belong to the superclass “flowers”.

Federated-LDA. Following the adaptation proposed by [290], both datasets are split evenly among 100 clients, thus each of them has access to 500 data samples. This partitioning is performed according to a Latent Dirichlet Allocation (LDA) on the labels. In practice, each local dataset follows a multinomial distribution drawn according to a symmetric Dirichlet distribution with concentration parameter α. The higher the value of this parameter is, the more the local datasets resemble a homogeneous scenario, in the limit case $\alpha = 0$ each client has access to one only class of images. In this thesis, $\alpha \in \{0, 0.05, 100\}$ in CIFAR10 and $\alpha \in \{0, 0.5, 1000\}$ in CIFAR100. Figures 3.5a and 3.5b show how data is distributed across clients in all the experimental settings for these two datasets.

Federated-PAM. Proposed by [275], CIFAR100-PAM reflects the “coarse" and “fine" label structure of the dataset for a more realistic partition. The dataset is split among 500 clients - with 100 images each - following the Pachinko Allocation Method (PAM) [391], on the result of which LDA is applied.



(a) CIFAR10



(b) CIFAR100

Fig. 3.5 CIFAR10 *(a)* and CIFAR100 *(b)* data distribution across clients with the heterogeneity degree determined by α. The average number of classes seen by each client is reported on top of each chart.

Data pre-processing. The 32×32 input images are pre-processed following the standard pipeline: the training images are randomly cropped applying padding 4 with final size 32×32, randomly horizontally flipped with probability 0.5 and finally the pixel values are normalized with the dataset's mean and standard deviation. Normalization is applied to test images as well.

Corrupted CIFAR. CIFAR10-C and CIFAR100-C are the corrupted versions of the CIFAR datasets. As part of the benchmark proposed by [196], they are used for testing the image classifiers' robustness. The $10k$ test images are modified according to a given *corruption* and a corresponding level of *severity*, resulting in 19 possible corruptions (brightness, contrast, elastic blur, elastic transform, fog, frost, Gaussian blur, Gaussian noise, glass blur, impulse noise, JPEG compression, motion blur, pixelate, saturate, short noise, snow, spatter, speckle noise, zoom blur), with severity ranging from 1 (low) to 5 (high).

Google Landmarks v2

Task. Google Landmarks v2 [381] is an image classification dataset over $2,028$ distinct natural landmarks.

Table 3.2 Image classification datasets

Dataset	Clients	Size imbalance	Training samples	Classes
CIFAR10	100	✗	50,000	10
CIFAR100	100	✗	50,000	100
CIFAR100-PAM	500	✗	50,000	100
LANDMARKS-USER-160K	1,262	✓	164,172	1,028
FEMNIST	3,550	✓	805,263	62
CELEBA	9,343	✓	200,288	2

Centralized. The dataset contains approximately 5 milion images, depicting pictures of landscapes taken around the world by various contributors.

Federated. The LANDMARKS-USER-160K dataset [278] is the federated adaptation of Google Landmarks v2 and contains 164,172 images, distributed among 1262 clients, accounting for authorship information. Each contributor to the dataset was required to provide at least 30 images, capturing a minimum of 5 distinct landmarks. Each location was visited by 10 or more contributors and is depicted in at least 30 images. The authors in the test set do not overlap with the ones appearing in the training split.

Data pre-processing. The data pre-processing pipeline adheres to the standard implementation used for training models on ImageNet. The input images are resized and cropped to 224×224 with random scale and aspect ratio as described in [392]. The data augmentation pipeline used for the experiments can be found here[1].

FEMNIST

Task. EMNIST (Extended MNIST) [393] is the extended version of MNIST [112], containing handwritten characters (A-Z, a-z) and digits (from 0 to 9) for a total of 62 classes.

Centralized. The dataset contains 805,263 black and white images of shape 28×28.

[1]https://github.com/google/flax/blob/571018d16b42ce0a0387515e96ba07130cbf79b9/examples/imagenet/input_pipeline.py#L90-L108

Federated. FEMNIST (Federated Extended MNIST) was first introduced as part of the LEAF benchmark [394]. The partitioning between clients preserves the authorship information, resulting in a total of $3,500$ clients with an average of 226 samples each. The number of local samples significantly varies across clients.

CelebA

Task. CelebA [395] is a dataset containing pictures of celebrities' faces, each characterized by 40 attributes such as gender, presence of glasses or beard, and smile. The task involves binary classification, aiming to determine whether the person in the picture is smiling or not.

Centralized. The dataset contains $202,599$ face images for a total of $10,177$ identities.

Federated. Following [394], federated CelebA partitions the dataset based on the depicted celebrity, excluding those with fewer than 5 images. This results in 9,343 users, each with access to an average of 21 samples.

3.4.2 Semantic Segmentation

This Section details the semantic segmentation datasets for autonomous driving. Their federated adaptation is one of the contributions of this thesis and is discussed in details in Chapter 6.

Cityscapes

Task. Cityscapes [396] is among the most used datasets for semantic segmentation. It accounts for 19 classes and provides street-view images from 50 cities in Central Europe. Examples of extracted images and corresponding segmentation maps can be found in Figure 2.2.

Centralized. The training set contains 2,975 real photos taken in the streets of 50 different cities with good weather conditions. There are 500 test images.

Data pre-processing. The images are randomly scaled in the range (0.5, 1.5) and cropped to a 512×1024 shape.

Mapillary Vistas

Task. The Mapillary Vistas dataset [397] collects geo-localized street-view images from all around the world with 19 semantic classes.

Centralized. The dataset contains 17,969 training and 2,000 test images.

IDDA

Task. IDDA [174] is a synthetic dataset for semantic segmentation, simulating images captured by autonomous vehicles, and includes 16 semantic classes.

Centralized. The dataset contains $1,006,800$ 1920×1080 images. The driving conditions are characterized by 7 towns (ranging from urban to rural environments), 5 viewpoints (simulating different vehicles), and 3 weather conditions (noon, sunset, and rainy scenarios), resulting in a total of 105 domains.

Data pre-processing. The images are randomly scaled in the range (0.5, 2.0) and cropped to a 512×928 shape.

CrossCity

Task. CrossCity [398] collects driving scenes from Rome, Rio, Tokyo, and Taipei, with 13 semantic classes.

Centralized. The dataset is made of 12,800 training and 400 test images.

GTA5

Task. GTA5 [399] is a synthetic dataset of highly realistic road scenes of typical US-like urban and suburban environments, rendered from the videogame Grand

Theft Auto 5. Each image has pixel-level semantic annotations over 19 classes, which are compatible with the ones of Cityscapes.

Centralized. The training set contains 24,966 images.

3.4.3 Visual Place Recognition

Mapillary Street-Level-Sequences

Task. The Mapillary Street-Level Sequences (MSLS) dataset [400] is geographically distributed across 30 cities worldwide, including Amsterdam, Manila, San Francisco, and Copenhagen, covering over 4,228 km. The images are captured by various users, featuring a diverse range of cameras, weather conditions, times of day, and scenarios spanning both urban and rural environments.

Centralized. The dataset contains approximately 1.68 million images and is divided into non-overlapping training, validation, and test sets, with each set comprising distinct cities: Amsterdam and Manila for validation, San Francisco and Copenhagen for testing, and the remaining cities for training. Each subset is further split into databases and queries, where queries represent images to be localized, and databases serve as the system's prior knowledge of the area.

Chapter 4

Generalization through the Lens of the Loss Landscape

A mathematician who can only generalise is like a monkey who can only climb up a tree, and a mathematician who can only specialise is like a monkey who can only climb down a tree. In fact neither the up monkey nor the down monkey is a viable creature

GEORGE PÓLYA

This Chapter introduces the first contribution of this thesis. The issue of lack of generalization in heterogeneous federated learning is addressed through the lens of the geometry of the loss surface. After delineating the contributions (Section 4.1), Section 4.2 studies the models behavior in heterogeneous FL, placing particular emphasis on the impact of label skew on model performance, convergence speed and stationary points. Section 4.3 introduces FEDSAM, a novel method that applies Sharpness-aware Minimization and Stochastic Weight Averaging in federated learning to improve generalization by promoting convergence towards flatter minima. Sections 4.4 and 4.5 study the limitations of FEDSAM and propose alternative solutions. The contributions are summarized in Section 4.6.

4.1 Introduction

A fundamental challenge in cross-device heterogeneous federated learning is the lack of generalization. This arises due to heavily skewed local distributions, causing local models to drift from the convergence minimum in the global optimization path, ultimately resulting in poor generalization to the overall data distribution (Chapter 3). This thesis posits that, in addition to client drift, local models tend to overfit their training data, which contributes to a global model that lacks generalization. This issue is exacerbated by local datasets that are limited in size and contain only a subset of the target task's classes. While this simplifies the learning process for local models, it hinders their ability to adapt to the full range of potential inputs.

Furthermore, no previous work has attempted to explain the (lack of) generalization in heterogeneous federated learning through the lens of the loss surface. To gain a comprehensive understanding of models' behavior in FL, this thesis asks: *Do models converge to sharp minima in heterogeneous FL? If so, can generalization be improved by promoting convergence towards flat minima?*

To enhance the generalization ability of the global model by learning better local models, this thesis proposes leveraging Sharpness-Aware Minimization (SAM) [213] during client-side training to reach flat regions in the loss landscape, and Stochastic Weight Averaging (SWA) [237] during server-side aggregation for increased robustness. This approach, named FEDSAM, has been shown to outperform the state of the art on various vision tasks (Section 4.3).

However, FEDSAM +SWA comes with a few limitations:

- Inconsistency between local and global loss landscapes: in heterogeneous FL, *local* flat minima may not directly translate to *global* flat minima.
- Increased computational cost: SAM requires two optimization steps at each training iteration, increasing the computational burden on resource-constrained devices.
- Limited efficacy of SWA: SWA can only be applied during the final stages of training, which restricts its overall impact and usability.

To address these limitations, FEDGLOSS targets the inconsistency between local and global sharpness by using SAM in the server-side optimization process (Section 4.4).

WIMA (Window-based Model Average) instead overcomes the SWA's limitations by introducing a window-based aggregation of global models, applicable from the initial training stages (Section 4.5). All these solutions are shown to outperform the state of the art on multiple vision tasks.

The ideas and results presented in this Chapter have led to the publication of the following works:

- Caldarola, D., Caputo, B., & Ciccone, M. (2022).
  Improving Generalization in Federated Learning by Seeking Flat Minima.
  In *European Conference on Computer Vision* (pp. 654-672). Springer Nature Switzerland (**ECCV 2022**).

- Caldarola, D., Caputo, B., & Ciccone, M. (2023).
  Window-based Model Averaging Improves Generalization in Heterogeneous Federated Learning.
  In *Proceedings of the IEEE/CVF International Conference on Computer Vision, Women in Computer Vision Workshop* (pp. 2263-2271). (**ICCVW 2023**).

The paper mentioned below is currently under review in a peer-reviewed conference:

- Caldarola, D., Cagnasso, P., Caputo, B., & Ciccone, M. (2024).
  Beyond Local Sharpness: Communication-Efficient Global Sharpness-aware Minimization for Federated Learning.

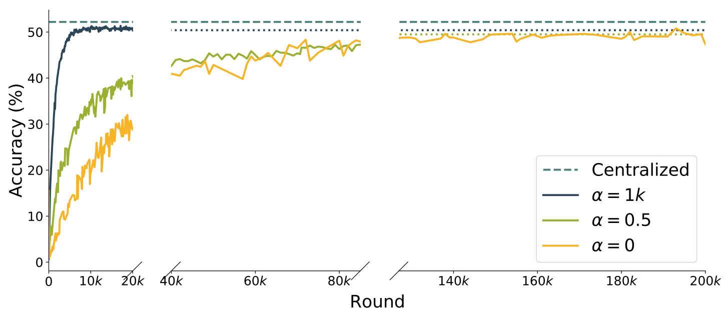


Fig. 4.1 CIFAR100 accuracy trends with varying data heterogeneity, compared to the centralized upper bound. Differently from homogeneous federated settings ($\alpha = 1k$), models trained in heterogeneous FL ($\alpha \in \{0, 0.5\}$) necessitates more rounds to converge.

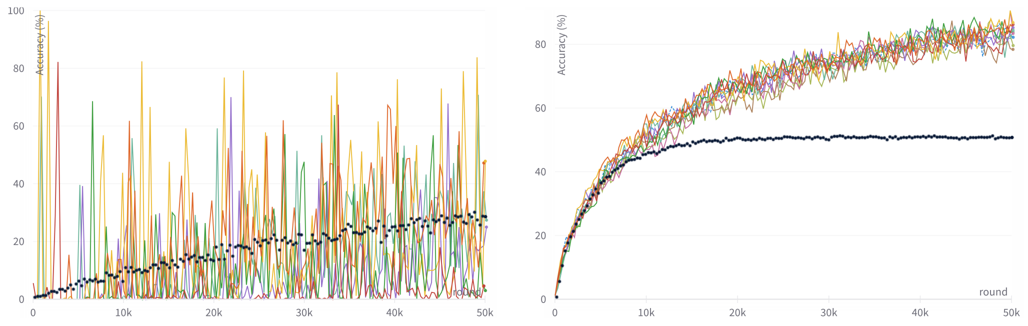
4.2 Where Heterogeneous Federated Learning Fails at Generalizing

To fully understand the behavior of a model trained in a heterogeneous FL scenario, this Section introduces a thorough empirical analysis to underline the impact of non-*i.i.d.* data on the convergence speed, stationary points in the loss landscape and on the training hyperparameters (*e.g.*, number of selected clients and number of local epochs).

Experimental Setup. The experimental setup replicates the one proposed by Hsu *et al.* [278], including both the dataset and network configuration. The chosen dataset is the federated CIFAR100, split following a Dirichlet distribution with concentration parameter α, as described in Section 3.4.1. For simulating a heterogeneous scenario, α values of 0 and 0.5 are chosen, while for a homogeneous scenario, α is set to 1000. The model undergoes training for $20,000$ rounds. The chosen architecture is a CNN similar to LeNet5 [112]. The de-facto standard FEDAVG is used for model aggregation, and SGD for client-side optimization. Further implementation details are provided in Chapter A.

4.2.1 Convergence Under Label Skew

Figure 4.1 illustrates the impact of label skew in FL. The centralized baseline is the reference upper bound, as discussed in Section 3.1.1. A comparison of learning



(a) $\alpha = 0$ (b) $\alpha = 1000$

Fig. 4.2 CIFAR100. Global model performance on local distributions with *(a)* $\alpha = 0$ and *(b)* $\alpha = 1000$ at $20k$ rounds. Each color represents a local distribution (*i.e.*, one class for $\alpha = 0$), while the dark line represents the performance on the global test set.

trends between homogeneous and heterogeneous federated settings reveals that the latter exhibits significantly noisier and more unstable trends, as well as a considerable performance gap. Despite this, it is observed that the heterogeneous distribution of the data does not entirely inhibit the model from achieving comparable performance; it merely requires a larger number of rounds to reach convergence. Specifically, with a round budget ten times larger, the model eventually converges. This indicates that learning is slowed down but not impaired in heterogeneous settings, suggesting there is room for improvement.

An analysis of the model's behavior reveals that shifts in client data distribution lead to significant fluctuations in learning. At each round, the model focuses on a subset of recently seen tasks and fails to generalize to previously learned ones. This phenomenon, known as *catastrophic forgetting* in neural networks [401], is typical in multitask learning [313, 402]. Figure 4.2 illustrates this by comparing the accuracy of the global model on the clients' data and the test set for $\alpha = 0$ and $\alpha = 1000$. In the first case, the model achieves very high performance on one class at each round but forgets others, with this behavior only slightly improving as training progresses (Figure 4.2a). Conversely, in the homogeneous scenario, the model behaves similarly across each client, achieving convergence more easily but leading to overfitting as the number of rounds increases (Figure 4.2b).

Fluctuations in model predictions can also be observed by examining its output features, $f_{\boldsymbol{w}}(\boldsymbol{x})\, \forall \boldsymbol{x} \in \mathscr{X}$. Figure 4.3 shows the L2-norm of the output features computed using the current global model parameters $\boldsymbol{w}^t$ at each round $t \in [T]$, given

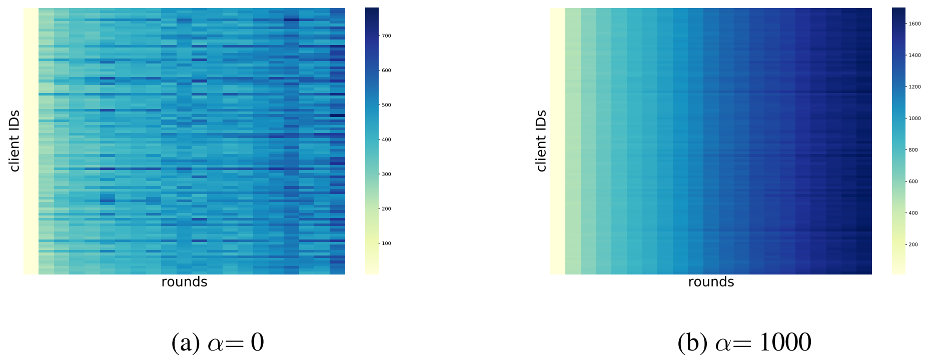
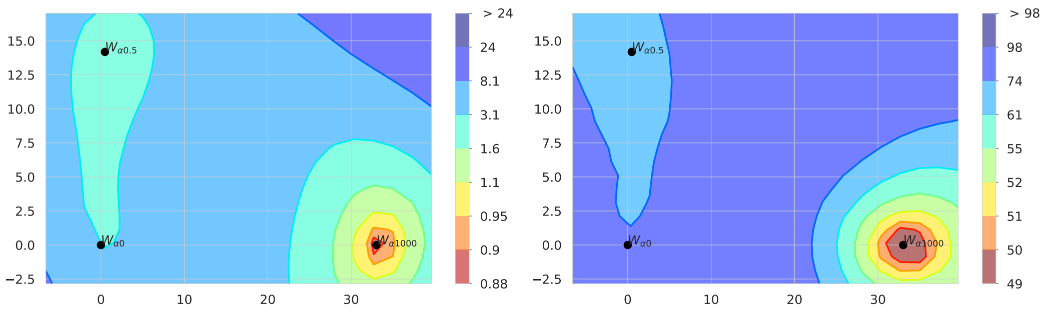


Fig. 4.3 CIFAR100. L2-norm of global classifier output features over successive rounds, using each client's local data as input. Models trained with FEDAVG. *(a)* With $\alpha = 0$, at each round, the model tends to focus on a different client's distribution, *i.e.*, a single class. *(b)* With $\alpha = 1000$, the model distributes attention evenly across all distributions.

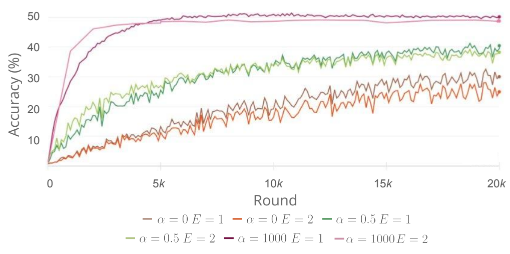


(a) Train loss surface (b) Test error surface

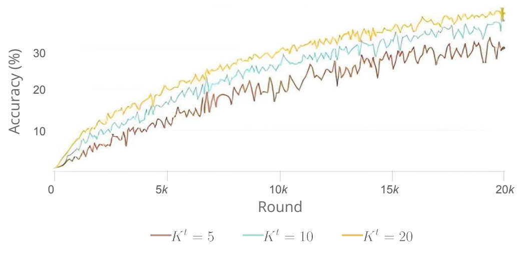
Fig. 4.4 CIFAR100. Global model convergence points in the *(a)* training loss and *(b)* test error surfaces when trained with varying data heterogeneity, with $\alpha \in \{0, 0.5, 1000\}$, after $20k$ rounds.

the local clients' data $\mathscr{D}_k$ for all $k \in \mathscr{C}$. A higher norm value corresponds to greater attention paid to that class by the network. The uniformity of the features obtained in the homogeneous setting (Figure 4.3b) contrasts with the chaotic distribution observed when $\alpha = 0$, where the values vary significantly over time without following a consistent trend (Figure 4.3a).

From the loss landscape perspective (Figure 4.4), given $T = 20k$ rounds, models trained with varying degrees of heterogeneity end up at different points, as anticipated by their performance in Figure 4.1. The model trained in an *i.i.d.* setting reaches a local minimum, while the others remain in high-loss regions due to slower learning trends. Furthermore, the shift between the training (Figure 4.4a) and testing surfaces



(a) Local epochs



(b) Number of selected clients

Fig. 4.5 CIFAR100. *(a)* Impact of local training epochs E in heterogeneous (α $\{0, 0.5\}$) and homogeneous FL (α= 1000). *(b)* Impact of selected clients at each round t, K^t, with α= 0.

(Figure 4.4b) indicates that the models trained in the heterogeneous setting are unable to generalize well to unseen data.

Impact of Training Hyperparameters

This section analyzes the impact of training hyperparameters under non-*i.i.d.* data distribution across clients, taking into account total rounds T, number of local epochs E and number of clients selected at each round $K^t \triangleq |\mathscr{C}^t|$.

As shown in Figure 4.1, the number of rounds required to reach convergence varies depending on the data distribution, with heterogeneous settings necessitating nearly 10 times more rounds to achieve the performance of the centralized baseline. Additionally, this plot highlights a fundamental insight: the increased number of rounds in heterogeneous FL implicitly indicates more communication exchanges, thereby underscoring the importance of communication efficiency in such settings.

Figure 4.5a examines the impact of the number of local training epochs in both homogeneous and heterogeneous federated learning, with $E \in \{1, 2\}$. Consistent with previous works [45, 281], a larger number of epochs results in increased client drift in heterogeneous settings, further slowing down training and degrading performance. Conversely, when α= 1000, increasing the number of epochs accelerates convergence and ultimately leads to overfitting to the overall distribution. Motivated by these results, $E = 1$ in the results presented in this thesis.

Lastly, increasing the number of clients selected at each round t positively impacts the training progress, as shown in Figure 4.5b. Involving more clients statistically increases the likelihood of selecting users with similar distributions, which leads to constructive interference during model aggregation. Additionally, merging more updates reduces noise, enhancing the overall stability and performance of the model. However, as discussed in Section 3.1, clients in cross-device federated learning are not always reliable, which negatively impacts their likelihood of participating in multiple rounds.

In conclusion, models trained in heterogeneous settings suffer from client drift and catastrophic forgetting, resulting in slower, unstable, and noisy learning trends. These negative effects are exacerbated by a larger number of training epochs but can be alleviated with higher client participation. Nonetheless, convergence can be achieved, making communication efficiency of utmost importance.

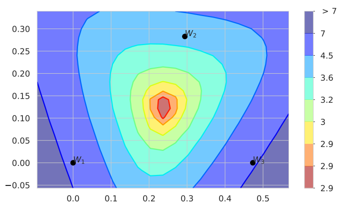
4.3 Improving Generalization in Federated Learning by Seeking Flat Minima

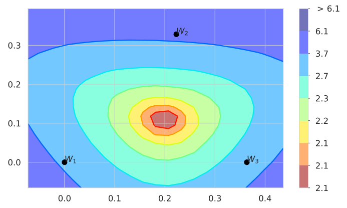
Reproduced with permission from Springer Nature: Caldarola, D., Caputo, B., & Ciccone, M. (2022). Improving generalization in federated learning by seeking flat minima. In European Conference on Computer Vision - ECCV (pp. 654-672). Springer Nature Switzerland.

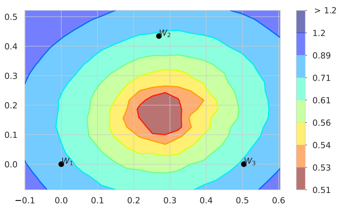
Models trained in federated settings often suffer from degraded performance and poor generalization, particularly in heterogeneous scenarios. This work investigates such behavior through the geometry of the loss and Hessian eigenspectrum, linking the model's lack of generalization capacity to the sharpness of the solution. Motivated by prior studies connecting the sharpness of the loss surface and the generalization gap, it is demonstrated that training clients locally with Sharpness-Aware Minimization (SAM) or its adaptive version (ASAM), and averaging stochastic weights (SWA) on the server side, can substantially improve generalization in federated learning and help bridge the gap with centralized models. By seeking parameters in neighborhoods with uniformly low loss, the model converges towards flatter minima, significantly improving generalization in both homogeneous and heterogeneous scenarios. The code is available at https://github.com/debcaldarola/fedsam.

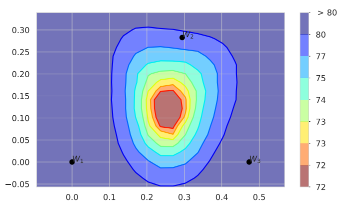
4.3.1 Motivation

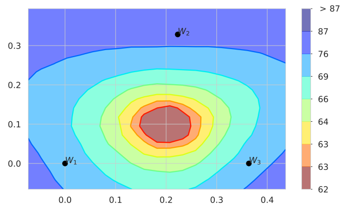
This work begins by analyzing the heterogeneous federated learning scenario to identify the causes of poor generalization in federated algorithms. Building upon the insights presented in Section 4.2, it is hypothesized that local models overfit their current distributions during training, an issue exacerbated by limited dataset sizes and accessible classes. This hypothesis is verified in Figure 4.6, which shows the positions of three local models $\boldsymbol{w}_1, \boldsymbol{w}_2$ and $\boldsymbol{w}_3$ after $20k$ rounds, before the server-side aggregation. The training loss surfaces (Figures 4.6a to 4.6c) reveal that local models are distant from the center of the minimum, with this distance increasing as data heterogeneity rises. This implies that, despite achieving 100% training accuracy on their local datasets, the models fail to generalize to the underlying global training distribution. This failure becomes even more evident on the test set (Figures 4.6d to 4.6f), where models trained with $\alpha = 0$ and $\alpha = 0.5$ are found in


(a) $\alpha = 0$ Train loss surface


(b) $\alpha = 0.5$ Train loss surface


(c) $\alpha = 1k$ Train loss surface


(d) $\alpha = 0$ Test error surface


(e) $\alpha = 0.5$ Test error surface

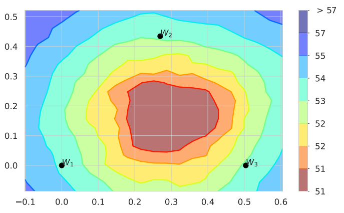
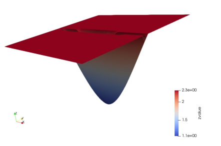
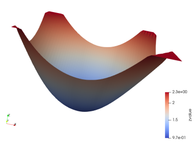
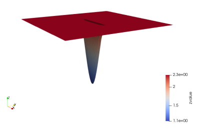

(f) $\alpha = 1k$ Test error surface

Fig. 4.6 CIFAR100 with $\alpha \in \{0, 0.5, 1k\}$. Train loss (*top*) and test error surfaces (*bottom*) of three local models resulting from 20k training rounds with FEDAVG.


(a) $\alpha = 0$ FEDAVG


(b) $\alpha = 0$ FEDASAM


(c) $\alpha = 1k$ FEDAVG

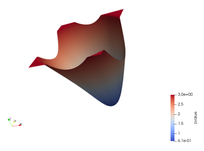
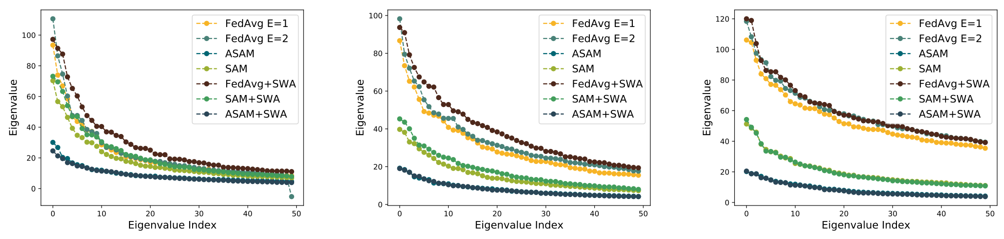

(d) $\alpha = 1k$ FEDASAM

Fig. 4.7 CIFAR100. 3D loss landscape visualizations of global models trained using FEDAVG with *(a-b)* $\alpha = 0$ and *(c-d)* $\alpha = 1k$, with FEDAVG or FEDASAM. FL models converge to sharp minima.

high-error regions. Thus, the resulting global model fails to generalize to the overall underlying distribution.

Inspired by recent trends in deep learning that connect the geometry of the loss surface to the generalization gap [52, 207–209, 228, 237], this work investigates the loss surface geometry of models trained in non-*i.i.d.* federated scenarios. The aim is to determine whether sharp minima contribute to the lack of generalization in heterogeneous FL. Following [208], the loss surfaces obtained from models trained with varying heterogeneity are plotted and reported in Figure 4.7. The results show that *FL models converge to sharp regions*, providing a plausible explanation for the observed lack of generalization.

Additionally, [52] characterizes flatness through the eigenvalues of the Hessian, where the dominant eigenvalue λ_1 evaluates the worst-case landscape curvature. A larger λ_1 indicates greater changes in loss in that direction and a steeper minimum



(a) $\alpha = 0$ (b) $\alpha = 0.5$ (c) $\alpha = 1000$

Fig. 4.8 CIFAR100. Hessian eigenspectra of the global model final parameters with $\alpha \in \{0, 0.5, 1000\}$.

Table 4.1 Hessian maximum eigenvalue λ_1 and the ratio λ_1/λ_5 as a proxy for sharpness. FEDAVG reaches sharp minima, while models trained with our sharpness-aware solutions combined with SWA converge to flat regions. CIFAR100.

Algorithm	λ_1		λ_1/λ_5	
	$\alpha = 0$	$\alpha = 1k$	$\alpha = 0$	$\alpha = 1k$
FEDAVG $E = 1$	93.46	106.14	2.00	1.31
FEDAVG $E = 2$	110.62	118.35	2.32	1.30
FEDSAM	70.29	51.28	**1.79**	1.48
FEDASAM	**30.11**	<u>**20.19**</u>	1.80	<u>**1.27**</u>
FEDAVG+SWA	97.24	120.02	1.49	1.39
FEDSAM+SWA	73.16	54.20	1.56	1.61
FEDASAM+SWA	<u>**24.57**</u>	**20.49**	<u>**1.51**</u>	**1.30**

(Section 2.4.5). The Hessian eigenspectrum (first 50 eigenvalues) is computed using the power iteration method and shown in Figure 4.8. Table 4.1 reports the values of λ_1 and the ratio λ_1/λ_5, commonly used as proxies for sharpness, as the heterogeneity varies. As expected, λ_1 is large in all settings when using FEDAVG, indicating that this method leads models towards sharp minima regardless of the data distribution, consistent with the observations from the loss landscapes. At the same time, the ratio λ_1/λ_5 is smaller when $\alpha = 1k$, implying a less sharp minimum and that larger *sharpness is correlated with data heterogeneity*.

These results suggest that explicitly searching for flatter minima can potentially improve global model generalization in FL.

4.3.2 Federated Sharpness-Aware Minimization

To expedite training and reduce the performance gap in the presence of non-*i.i.d.* data, efforts are directed towards improving the model's generalization ability. Common first-order optimizers (*e.g.*, SGD, Adam [273]) are typically non-robust to unseen data distributions [214], as they focus solely on minimizing the training loss $\mathscr{L}_{\mathscr{D}}$ without considering higher-order information related to generalization, such as curvature. This issue is exacerbated in federated learning due to inherent statistical heterogeneity, leading to sharp minima and poor generalization. This work hypothesizes that encouraging the local model to converge towards flatter neighborhoods may help bridge the generalization gap. To this end, methods from current literature that explicitly seek flat minima are introduced in the federated training: Sharpness-Aware Minimization (SAM) or its adaptive version (ASAM) on the client-side (FEDSAM and FEDASAM respectively), and Stochastic Weight Averaging (SWA) on the server-side (Figure 4.9). The overall process is delineated in Algorithm 4.

Intuitively, by minimizing the sharpness of the loss surface and the generalization gap, local models become more robust to unseen data distributions and, when averaged, contribute to a more solid central model.

Sharpness-aware Minimization in Client-side Optimization

FEDSAM aims to improve the generalization of the clients' models through convergence to flatter regions by using SAM in the local training. From Equations (2.29) and (3.2), its global objective becomes

$$\min_{\boldsymbol{w}} \left\{ f^{\text{SAM}}(\boldsymbol{w}) = \frac{1}{K} \sum_{k \in \mathscr{C}} f_k^{\text{SAM}}(\boldsymbol{w}) \right\}, f_k^{\text{SAM}}(\boldsymbol{w}) \triangleq \max_{\|\boldsymbol{\epsilon}_k\| \leq \rho} f_k(\boldsymbol{w} + \boldsymbol{\epsilon}_k), \tag{4.1}$$

with $\boldsymbol{\epsilon}_k$ being the local perturbation. The intuition behind this approach is that the server indirectly inherits sharpness-minimizing behavior from the clients, and the improved local models' generalization ability positively reflects on the global model.

FedSAM as a Two-Player Game Formulation. As discussed in Section 2.4.5, the robustness of deep learning models can be formalized as a two-player game between an adversary and the model. Similarly, SAM's min-max optimization problem (Equation (2.29)) reminds of a game between a "perturbator" and a model. The former aims to find a parameter perturbation $\boldsymbol{\epsilon}$ that maximizes the training loss, implicitly identifying sharp regions in the loss landscape, while the latter's objective is to minimize the maximum loss deriving from small perturbations to its parameters. This results in model parameters that effectively avoid sharp minima and flatten the loss surface. In FEDSAM instead, the robust formulation as a two-player game operates at two interconnected levels: local (client-specific) and global (federated) objectives. At the local level, each client plays the standard SAM game, where the local model (*Player 1*) minimizes the worst-case loss induced by its local perturbator (*Player 2*). Moreover, this **local two-player game** is nested within the global optimization process (Equation (4.1)), introducing an additional layer of interaction between the local and global objectives. In this setting, while the server does not actively engage in the sharpness-aware two-player game at the client level, it plays a critical role in integrating client updates to advance the global model. This introduces an additional **coordination game**, where clients' local two-player outcomes contribute to the global update, influencing the overall robustness and generalization. By balancing competing local sharpness-aware objectives, the server implicitly inherits the robustness from local SAM.

To underline the differences between SAM's and FEDSAM's games in centralized and federated learning respectively, Equations (2.29) and (4.1) are reported with unified notation:

$$\text{SAM:}\quad \min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}} \max_{\|\boldsymbol{\epsilon}\|\le\rho} \mathscr{L}\left(f_{\boldsymbol{w}+\boldsymbol{\epsilon}}(\boldsymbol{x}), y\right) \tag{4.2}$$

$$\text{Client-side SAM:}\quad \min_{\boldsymbol{w}_k} \mathbb{E}_{(\boldsymbol{x}_k,y_k)\sim\mathscr{D}_k} \max_{\|\boldsymbol{\epsilon}_k\|\le\rho} \mathscr{L}(f_{\boldsymbol{w}_k+\boldsymbol{\epsilon}_k}(\boldsymbol{x}_k), y_k) \tag{4.3}$$

$$\text{FedSAM:}\quad \min_{\boldsymbol{w}} \frac{1}{K} \sum_{k\in\mathscr{C}} \min_{\boldsymbol{w}_k} \mathbb{E}_{(\boldsymbol{x}_k,y_k)\sim\mathscr{D}_k} \max_{\|\boldsymbol{\epsilon}_k\|\le\rho} \mathscr{L}(f_{\boldsymbol{w}_k+\boldsymbol{\epsilon}_k}(\boldsymbol{x}_k), y_k). \tag{4.4}$$

In centralized SAM, the perturbation $\boldsymbol{\epsilon}$ is applied globally to the centralized model parameters $\boldsymbol{w}$ and is computed based on the entire training dataset $\mathscr{D}$. In FEDSAM instead, locally, each client k seeks to minimize its local sharpness-aware loss. Each local perturbator maximizes the local loss by finding the sharpest direction $\boldsymbol{\epsilon}_k$ for the client's data $\mathscr{D}_k$. The local model counteracts by adjusting $\boldsymbol{w}_k$ to minimize the effect of the worst-case $\boldsymbol{\epsilon}_k$. Since each $\boldsymbol{\epsilon}_k$ is tailored to the local distribution, their effect at the global level depends on how updates are aggregated. This is where the server comes into play, acting as a coordinator by aggregating the locally optimized models, implicitly inheriting the sharpness-minimizing behavior. Thus, at each round, clients alternate between playing the local game (SAM) and contributing updates to the

Algorithm 4 SAM/ASAM and SWA applied to FEDAVG

```
Require: Initial random model parameters $\boldsymbol{w}^0$, clients set $\mathscr{C}$, $T$ rounds, SWA learning rates $\gamma_1, \gamma_2$, neighborhood size $\rho > 0$, $\eta > 0$, batch size $B$, local epochs $E$, cycle length $c$
 1: for each round $t = 0$ to $T-1$ do
 2:     if $t = 0.75 * T$ then                                ▷ Apply SWA from 75% of training onwards
 3:         $\boldsymbol{w}_{\text{SWA}} \leftarrow \boldsymbol{w}^t$                                  ▷ Initialize SWA model
 4:     end if
 5:     if $t \geq 0.75 * T$ then
 6:         $\gamma = \gamma(t)$                                  ▷ SWA LR scheduling step (Eq. 2.37)
 7:     end if
 8:     Subsample a set $\mathscr{C}^t$ in $\mathscr{C}$ of clients
 9:     for each client $k$ in $\mathscr{C}^t$ in parallel do          ▷ Iterate over subset $\mathscr{C}^t$ of clients
10:         $\boldsymbol{w}_{k,0}^{t+1} \leftarrow \boldsymbol{w}^t$
11:         for each batch $b$ of size $B$ in $\mathscr{D}_k$ do
12:             Compute SGD gradient $\nabla_{\boldsymbol{w}} \mathscr{L}_{\mathscr{B}}(\boldsymbol{w}_{k,i}^t)$
13:             Compute $\hat{\epsilon}\left(\boldsymbol{w}_{k,i}^t\right) = \rho \frac{\nabla_{\boldsymbol{w}} \mathscr{L}_b\left(\boldsymbol{w}_{k,i}^t\right)}{\|\nabla_{\boldsymbol{w}} \mathscr{L}_b\left(\boldsymbol{w}_{k,i}^t\right)\|} =: \hat{\epsilon}(\boldsymbol{w})$     ▷ Solve Eq. 2.31
14:             $\boldsymbol{w}_{k,i+1}^t \leftarrow \boldsymbol{w}_{k,i}^t - \gamma \nabla_{\boldsymbol{w}} \mathscr{L}_{\mathscr{B}}(\boldsymbol{w}_{k,i}^t)|_{\boldsymbol{w}+\hat{\epsilon}(\boldsymbol{w})}$     ▷ Local update w/
    sharpness-aware gradient
15:         end for
16:         Send $\boldsymbol{w}_k^t$ to the server
17:     end for
18:     $\boldsymbol{w}^{t+1} \leftarrow \sum_{k \in \mathscr{C}^t} \frac{N_k}{N} \boldsymbol{w}_k^t$                         ▷ FEDAVG (Eq. 3.5)
19:     if $t \geq 0.75 * T$ and $\text{mod}(t, c) = 0$ then                  ▷ End of SWA cycle
20:         $n_{\text{models}} \leftarrow t/c$
21:         $\boldsymbol{w}_{\text{SWA}} \leftarrow \frac{\boldsymbol{w}_{\text{SWA}} \cdot n_{\text{models}} + \boldsymbol{w}^{t+1}}{n_{\text{models}} + 1}$               ▷ Update SWA average (Eq. 2.38)
22:     end if
23: end for
```

global objective. By aggregating these outcomes, the server effectively takes a meta-step in the larger federated optimization game.

Stochastic Weight Averaging for Robust Server-side Aggregation

To further enhance the robustness of the learning process, SWA is applied on the server-side. Adapting the scenario from [237] to the FL framework, the server

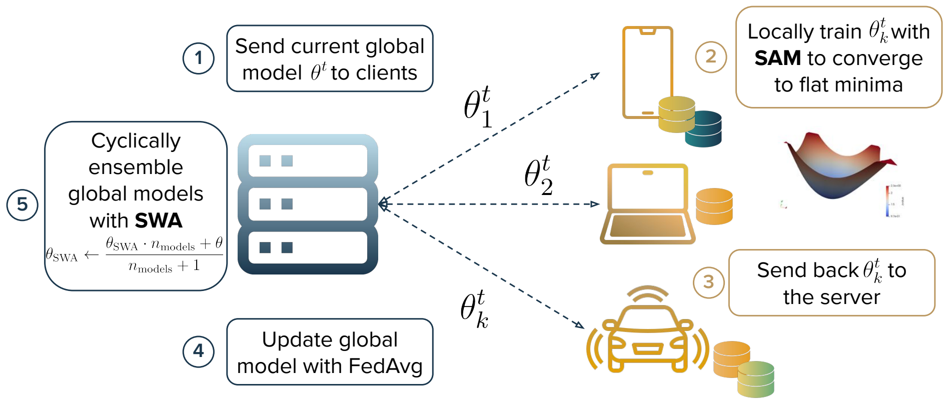

1 Send current global model θ^t to clients
2 Locally train θ_k^t with **SAM** to converge to flat minima
3 Send back θ_k^t to the server
4 Update global model with FedAvg
5 Cyclically ensemble global models with **SWA** $\theta_{\text{SWA}} \leftarrow \frac{\theta_{\text{SWA}} \cdot n_{\text{models}} + \theta}{n_{\text{models}} + 1}$
θ_1^t
θ_2^t
θ_k^t

Fig. 4.9 Overview of FEDSAM+SWA. At each round t, *(1)* the global model is sent to the clients, which *(2)* train it with SAM on their local data. *(3)* The model is sent back to the server, *(4)* where the updates are aggregated using FEDAVG and *(5)* then ensembled with previous models using SWA.

maintains two models, $f_{\boldsymbol{w}}$ and $f_{\boldsymbol{w}_{\text{SWA}}}$ (denoted as f and f_{SWA} for simplicity), from 75% of the training onwards. f follows the standard FEDAVG paradigm, while f_{SWA} is updated every c rounds.

4.3.3 Generalization and Convergence Speed-up with FedSAM

This section introduces the main results of the proposed FEDSAM, FEDASAM and their combination with SWA.

Federated Baseline

To determine the reference FL baseline with the optimal server-side optimizer according to Equation (3.4), SGD, Adam, and AdaGrad are evaluated cross both the heterogeneous ($\alpha = 0$) and homogeneous ($\alpha = 1k$) configurations of CIFAR100, involving five clients per round. According to the adaptive optimization strategies outlined in [275], the parameters β_1 and β_2 were set to 0 for AdaGrad, and to 0.9 and 0.99 for Adam, respectively. As demonstrated in Table 4.2, SGD with unitary learning rate, commonly referred to as FEDAVG, emerges as the superior choice, yielding satisfactory results in both scenarios, particularly in the heterogeneous configuration, becoming the reference in the following experiments.

Table 4.2 Comparison of different server-side optimizers with varying learning rate η_g on CIFAR100 @ 20k rounds. 5% clients participation. In bold the best results in terms of accuracy (%) on both α= 0 and α= 1k.

Optimizer	η_g	α**= 0**	α**= 1k**
SGD	1	**30.25**	**49.92**
	0.1	14.09	40.43
	0.01	2.67	11.35
	0.001	1.20	1.12
ADAM	1	1.00	51.73
	0.1	29.75	51.62
	0.01	13.72	40.12
	0.001	2.60	11.31
ADAGRAD	1	1.00	1.00
	0.1	1.77	46.74
	0.01	26.25	51.44
	0.001	9.70	32.01

Table 4.3 Results on CIFAR100 with $\alpha\in\{0, 0.5, 1000\}$ @ 20k rounds and CIFAR10 with $\alpha\in\{0, 0.05, 100\}$ @ 10k rounds, distinguished by clients participation at each round

	Algorithm	α**= 0**			α**= 0.5/0.05**			α**= 1000/100**		
		5cl	10cl	20cl	5cl	10cl	20cl	5cl	10cl	20cl
CIFAR100	FEDAVG $E = 1$	30.25	36.74	38.59	40.43	41.27	42.17	49.92	50.25	50.66
	FEDAVG $E = 2$	24.94	31.81	35.18	38.21	39.59	40.94	48.72	48.64	48.45
	FEDSAM	31.04	36.93	38.56	44.73	44.84	46.05	54.01	53.39	53.97
	FEDASAM	**36.04**	**39.76**	**40.81**	**45.61**	**46.58**	**47.78**	**54.81**	**54.97**	**54.50**
	FEDAVG+SWA	39.34	39.74	39.85	43.90	44.02	42.09	50.98	50.87	50.92
	FEDSAM+SWA	39.30	39.51	39.24	47.96	46.76	46.47	**53.90**	53.67	**54.36**
	FEDASAM+SWA	**42.01**	**42.64**	**41.62**	**49.17**	**48.72**	**48.27**	53.86	**54.79**	54.10
CIFAR10	FEDAVG $E = 1$	65.00	65.54	68.52	69.24	72.50	73.07	84.46	84.50	84.59
	FEDAVG $E = 2$	61.49	62.22	66.36	69.23	69.77	73.48	83.93	84.10	84.21
	FEDSAM	70.16	71.09	72.90	73.52	74.81	76.04	84.58	84.67	**84.82**
	FEDASAM	**73.66**	**74.10**	**76.09**	**75.61**	**76.22**	**76.98**	**84.77**	**84.72**	84.75
	FEDAVG+SWA	69.71	69.54	70.19	73.48	72.80	73.81	84.35	84.32	84.47
	FEDSAM+SWA	74.97	73.73	73.06	**76.61**	75.84	76.22	84.23	84.37	84.63
	FEDASAM+SWA	**76.44**	**75.51**	**76.36**	76.12	**76.16**	**76.86**	**84.88**	**84.80**	**84.79**

The Effective Search of Flat Minima in Federated Learning

Section 4.2 demonstrates that FL models in heterogeneous settings exhibit significant performance disparities when compared to homogeneous settings, especially under a fixed number of communication rounds. Table 4.3 shows these differences can reach up to 20 percentage points. This performance gap is partly attributed to clients' hyperspecialization to their local datasets, which hinders the global model's generalization across the training distribution. This phenomenon is further evidenced by the

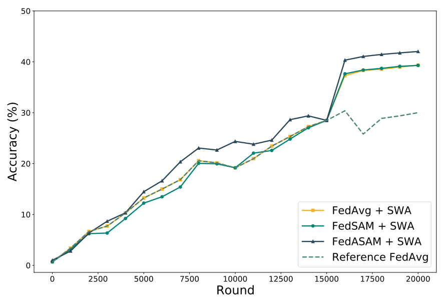
Table 4.4 Accuracy results (%) on Cifar100-PAM with ResNet18, with varying clients participation. The results underline the performance during training (@5k and @10k rounds), with and without Swa, and with the addition of strong data augmentations (mixup and cutout). Best results in bold.

Algorithm	Augmentation	10 clients			20 clients		
		@5k	@10k	w/ Swa	@5k	@10k	w/ Swa
FedAvg		46.60	47.03	52.70	46.51	45.83	50.28
FedSam		50.71	**53.10**	**55.44**	52.96	53.41	**54.67**
FedAsam		49.31	51.10	54.25	47.21	**53.50**	54.29
FedAvg $E = 2$		44.58	43.90	51.10	43.31	42.88	47.95
FedSam $E = 2$		**52.36**	52.04	55.23	51.41	51.35	53.41
FedAsam $E = 2$		49.03	49.33	53.01	**53.88**	52.94	54.18
FedAvg	mixup	43.47	49.25	56.71	50.33	49.89	55.74
FedSam	mixup	42.83	**51.92**	53.96	49.66	**55.77**	57.70
FedAsam	mixup	43.13	51.09	56.31	50.51	52.62	56.89
FedAvg $E = 2$	mixup	**44.76**	46.44	57.15	47.10	47.59	54.40
FedSam $E = 2$	mixup	42.17	51.04	56.54	**53.50**	54.75	**58.88**
FedAsam $E = 2$	mixup	**44.74**	50.14	**58.31**	49.87	50.87	55.86
FedAvg	cutout	48.64	48.59	55.40	47.00	46.96	51.70
FedSam	cutout	48.28	53.53	57.25	52.06	**54.37**	**56.70**
FedAsam	cutout	47.52	**52.13**	57.01	50.01	50.66	53.54
FedAvg $E = 2$	cutout	45.19	45.46	55.40	44.68	44.25	49.39
FedSam $E = 2$	cutout	**49.39**	51.88	**57.32**	**52.16**	52.37	55.45
FedAsam $E = 2$	cutout	48.99	50.09	55.77	48.48	48.77	52.00

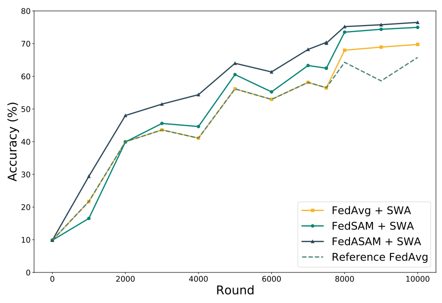
model's convergence to sharp minima, known to correlate with poor generalization. Table 4.3 reports the first results of various configurations of Sam, Asam, and their integration with Swa on the federated Cifar10 and Cifar100, with multiple heterogeneity levels, denoted by α, and client participation rate, K^t. Results on Cifar100-Pam are instead shown in Table 4.4.

Since both Sam and Asam involve alternating steps of gradient ascent and descent per iteration, they should be compared against FedAvg with two local epochs ($E = 2$). However, as also discussed in Section 4.2, a larger number of epochs usually results in degraded performance when $\alpha \rightarrow 0$. This trend is confirmed by the experiments in both Tables 4.3 and 4.4, thus establishing FedAvg with $E = 1$ as a more appropriate baseline due to its higher performance.

The mentioned results empirically demonstrate that optimizing for flat minima during both local training and server-side aggregation mitigates the issues arising in FL settings, particularly enhancing performance in heterogeneous environments. Experimental results underscore that incorporating Asam into FedAvg significantly enhances accuracies, showing improvements of $+6$ and $+8$ points for Cifar100 and



(a) CIFAR100

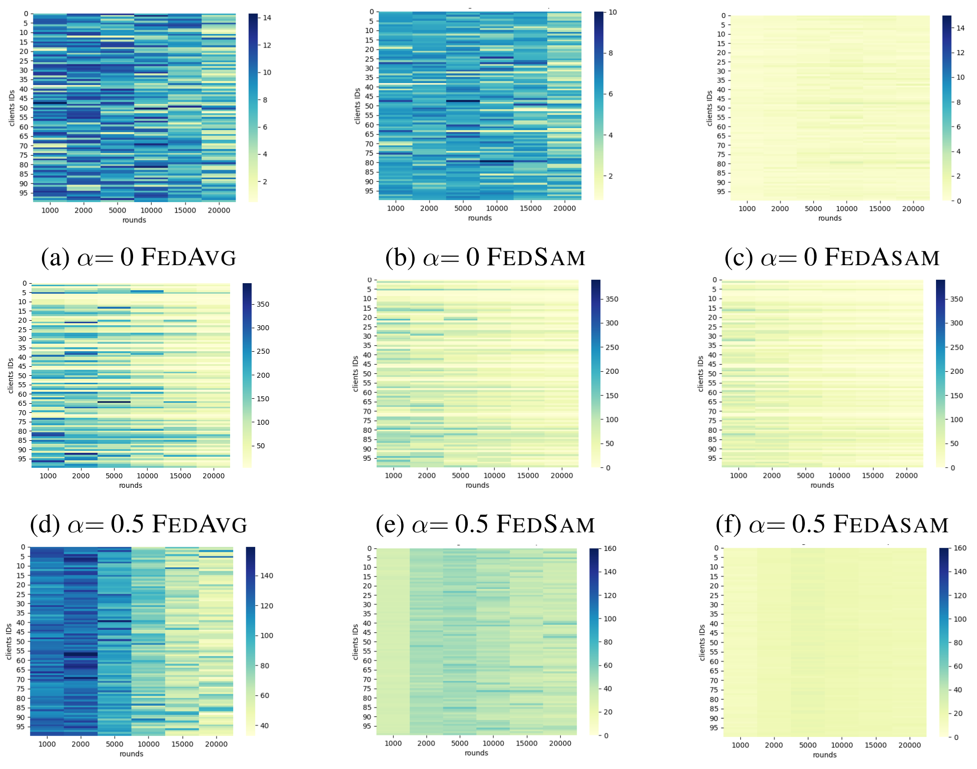


(b) CIFAR10

Fig. 4.10 Impact of SWA on the robustness of the model and training stability. The accuracy trends show the positive gap resulting from using SWA on top of FEDAVG or FEDASAM. Setting: $\alpha = 0$, 5% client participation.

CIFAR10 with the CNN, respectively, under the most challenging conditions ($\alpha = 0$ with 5% client participation). Further gains of $+12$ and $+11.5$ accuracy points are achieved when FEDASAM is combined with SWA. The stabilizing effect of SWA is particularly valuable in scenarios with lower client participation, which tend to exhibit more variability. Figure 4.10 highlights the improved stability and positive performance gains achieved by incorporating SWA, focusing on the most challenging scenario: $\alpha = 0$ and 5 clients per round on both CIFAR datasets. Ablation studies, detailed in Section 4.3.5, confirm that these improvements are primarily due to the averaging of stochastic weights (Equation (2.38)), rather than changes in the learning rate. The results with ResNet18 on CIFAR100-PAM instead reveal that SAM and SAM +SWA are more beneficial than ASAM in this specific setting.

Reaching Flatter Regions with ASAM and SWA. The analysis of the loss landscape and Hessian eigenspectrum is extended to models trained with FEDSAM, FEDASAM, and SWA. Both the loss surfaces (Figure 4.7) and the Hessian spectra (Figure 4.8) indicate that these methods facilitate convergence towards flatter minima. Specifically, examining the maximum eigenvalues, which represent the worst-case curvature, reveals a decrease in λ_1 from 93.5 with FEDAVG to 70.3 with FEDSAM and to 30.1 with FEDASAM in the most heterogeneous setting (Table 4.1). The result is further improved by combining FEDASAM and SWA, achieving $\lambda_1 = 24.6$. There is a strong correlation between the best λ_1 and the best ratio λ_1/λ_5. Even though the maximum eigenvalue resulting from FEDAVG+SWA and FEDSAM+SWA



(a) $\alpha = 0$ FEDAVG (b) $\alpha = 0$ FEDSAM (c) $\alpha = 0$ FEDASAM

(d) $\alpha = 0.5$ FEDAVG (e) $\alpha = 0.5$ FEDSAM (f) $\alpha = 0.5$ FEDASAM

(g) $\alpha = 1k$ FEDAVG (h) $\alpha = 1k$ FEDSAM (i) $\alpha = 1k$ FEDASAM

Fig. 4.11 Maximum Hessian eigenvalue λ_1^k computed for each client $k \in \mathscr{C}$ as rounds pass. CIFAR100. CNN.

is higher than their respective values without SWA, the corresponding lower ratio λ_1/λ_5 indicates that the majority of the spectrum lies in a lower curvature region, demonstrating the effectiveness of SWA.

Figure 4.11 instead compares the maximum Hessian eigenvalues obtained using the clients' local models. First, FEDAVG always leads to larger values, implying sharper local minima. Analyzing ASAM's behavior from each client's perspective, it is evident that flat minima are achieved from the beginning of the training for all values of α, positively impacting the model's performance.

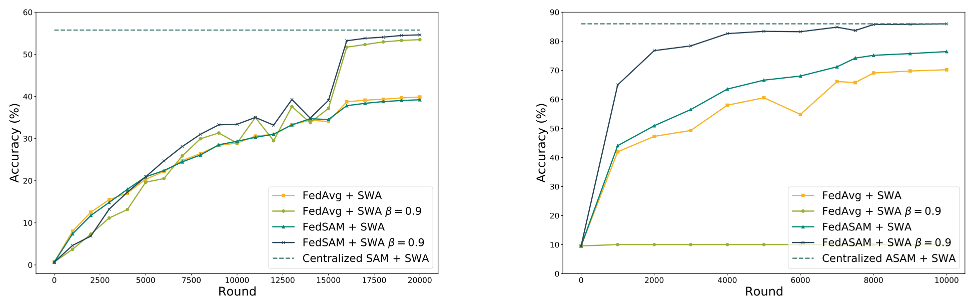
SAM is More Effective in FL than in Centralized Learning

Building upon the performance improvements observed with SAM, ASAM, and their combination with SWA, a question arises: *do these gains solely reflect the benefits observed in the centralized training scenario?* Table 4.5 addresses this

Table 4.5 Comparison of improvements (%) in centralized and federated scenarios ($\alpha \in \{0.5, 1k\}$, 5 clients) on CIFAR100, computed w.r.t. the reference at the bottom.

Algorithm	Accuracy				Absolute Improvement				Relative Improvement			
	Centr.	$\alpha=0$	$\alpha=0.5$	$\alpha=1k$	Centr.	$\alpha=0$	$\alpha=0.5$	$\alpha=1k$	Centr.	$\alpha=0$	$\alpha=0.5$	$\alpha=1k$
SAM	55.22	31.04	44.73	54.01	+3.02	+0.79	+4.30	+4.01	+5.79	+2.61	+10.64	+8.03
ASAM	55.66	36.04	45.61	54.81	+3.46	+5.79	+5.18	**+4.89**	+6.63	+19.14	+12.81	**+9.80**
SWA	52.72	39.34	43.90	50.98	+0.52	+9.09	+3.47	+1.06	+1.00	+30.05	+8.58	+2.12
SAM+SWA	55.75	39.30	47.96	53.90	+2.55	+9.05	+7.53	+3.98	+6.76	+29.92	+18.63	+7.97
ASAM+SWA	55.96	42.01	49.17	53.86	+3.76	**+11.76**	**+8.74**	+3.94	+7.20	**+38.88**	**+21.62**	+7.89
MIXUP	58.01	29.91	35.10	55.34	+5.81	-0.34	-5.33	+5.42	+11.13	-1.12	-13.18	+10.86
CUTOUT	55.30	24.24	37.72	53.48	+3.10	-6.01	-2.71	+3.56	+5.94	-19.87	-6.70	+7.13

Centralized: **52.20** - FEDAVG $\alpha=0$: **30.25**, $\alpha=0.5$: **40.43**, $\alpha=1k$: **49.92**



(a) CIFAR100 (b) CIFAR10

Fig. 4.12 SWA aids convergence for FEDAVGM/FEDSAM (CIFAR100, on the left) and FEDASAM (CIFAR10, on the right) in this highly heterogeneous case ($\alpha=0$, 20 clients per round). However, FEDAVGM +SWA fails on CIFAR10, while adding momentum to FEDASAM speeds up training.

by demonstrating that the positive performance gap achieved in the heterogeneous federated setting is larger than the one obtained in the centralized setting. This finding suggests that these approaches provide specific advantages within the federated learning context beyond what might be expected from a centralized training approach.

Reaching Convergence in Heterogeneous Federated Learning with FedSAM

Further experiments explore the impact of adding momentum ($\beta = 0.9$ in Equation (2.26)) to the server-side SGD optimization in the FEDAVG baseline, resulting in the FEDAVGM algorithm [290]. The combination of FEDAVGM with either SAM or ASAM and SWA achieves convergence even in highly heterogeneous scenarios

Table 4.6 Reaching convergence in heterogeneous scenarios with FEDAVGM (in bold). Results in terms of accuracy (%) computed with 20 clients per round.

	Algorithm	**Momentum**	**α= 0**	**α= 0.5/0.05**	**α= 1k/100**	**Momentum**	**α= 0**	**α= 0.5/0.05**	**α= 1k/100**
CIFAR100	FEDAVG	✗	38.59	42.17	50.66	✓	40.64	47.88	50.77
	FEDSAM	✗	38.56	46.05	53.97	✓	41.96	50.23	52.29
	FEDASAM	✗	40.81	47.78	54.50	✓	39.61	51.65	53.74
	FEDAVG+SWA	✗	39.85	42.09	50.92	✓	53.50	53.69	51.78
	FEDSAM+SWA	✗	39.24	46.47	54.36	✓	**54.63**	**54.93**	53.27
	FEDASAM+SWA	✗	41.62	48.27	54.10	✓	51.58	**56.19**	**54.72**
CIFAR10	FEDAVG	✗	68.52	73.07	84.59	✓	10.00	78.51	85.05
	FEDSAM	✗	72.90	76.04	84.82	✓	83.07	83.57	86.20
	FEDASAM	✗	76.09	76.98	84.75	✓	84.89	85.06	85.42
	FEDAVG+SWA	✗	70.19	73.81	84.47	✓	10.00	84.00	85.73
	FEDSAM+SWA	✗	73.06	76.22	84.63	✓	**85.65**	**85.98**	**86.79**
	FEDASAM+SWA	✗	76.36	76.86	84.79	✓	**85.98**	**86.03**	**86.00**

across both datasets (Table 4.6). However, FEDAVGM alone does not exhibit consistent convergence behavior. For example, in the CIFAR10 dataset with α= 0, the model remains stuck at random guess accuracy (10%) using FEDAVG. This issue is successfully addressed by applying our proposed method. Convergence behavior of the aforementioned training runs is illustrated in Figure 4.12.

Enabling the Use of Strong Data Augmentations in FL with FedSAM

The effectiveness of data augmentations in enhancing neural network performance and generalization is well-documented [181, 403, 404]. However, designing effective data augmentations often necessitates domain expertise and substantial computational resources, which may not be readily available in federated learning environments.

This work investigates the impact of commonly used strong data augmentations like MixUp [181] and Cutout [180] (Section 2.4.3) on FL tasks. Tables 4.3 and 4.7 present results obtained by applying these augmentations to the CIFAR100 (PAM and Dirichlet) and CIFAR10 datasets. Interestingly, the inclusion of MixUp and Cutout leads to performance degradation across all algorithms compared to the baseline without augmentation. This suggests that these specific augmentations may not be well-suited for the FL context and, in some cases, can hinder training progress. This behavior is also highlighted in Table 4.5.

Table 4.7 FEDAVG, SAM, ASAM and SWA with strong data augmentations (MixUp, Cutout)

	Algorithm	**SWA**	**Aug**	α= 0			α= 0.5/0.05			α= 1000/100		
				5*cl*	10*cl*	20*cl*	5*cl*	10*cl*	20*cl*	5*cl*	10*cl*	20*cl*
CIFAR100	FEDAVG	✗	MIXUP	29.91	33.67	35.67	35.10	37.80	39.34	55.34	**55.81**	**55.98**
	FEDSAM	✗		30.46	34.10	35.89	38.76	40.31	42.03	54.21	54.94	55.24
	FEDASAM	✗		34.04	36.82	36.97	40.71	42.24	**44.45**	49.75	49.87	49.68
	FEDAVG	✓		35.56	36.07	36.08	39.21	39.22	38.31	**55.43**	55.37	55.39
	FEDSAM	✓		35.62	36.25	35.66	42.13	41.95	42.03	52.9	53.14	53.48
	FEDASAM	✓		**40.08**	**38.74**	**37.47**	**44.53**	**43.97**	44.22	46.97	47.24	46.93
	FEDAVG	✗	CUTOUT	24.24	31.55	32.44	37.72	38.45	39.48	53.48	53.83	52.90
	FEDSAM	✗		23.51	30.92	33.12	40.33	40.31	42.58	**54.27**	**54.75**	**54.76**
	FEDASAM	✗		30.05	33.62	34.51	41.86	41.84	43.33	51.88	51.78	53.03
	FEDAVG	✓		33.65	34.40	35.03	40.43	40.12	39.32	53.87	54.09	52.75
	FEDSAM	✓		34.00	34.08	34.26	43.09	42.81	42.85	53.78	54.28	53.93
	FEDASAM	✓		**39.30**	**37.46**	**36.27**	**44.76**	**43.48**	**43.95**	50.00	49.65	50.81
CIFAR10	FEDAVG	✗	MIXUP	62.26	63.61	65.54	65.63	68.44	68.21	**82.38**	**84.46**	**83.58**
	FEDSAM	✗		67.35	69.32	69.78	70.34	72.98	72.54	81.88	82.24	82.25
	FEDASAM	✗		70.61	71.31	71.62	72.19	**72.84**	**72.72**	82.36	82.75	83.08
	FEDAVG	✓		66.31	66.89	66.26	69.79	69.12	68.80	82.27	82.88	82.67
	FEDSAM	✓		**72.42**	70.65	69.75	**73.36**	72.29	72.44	81.04	81.18	81.15
	FEDASAM	✓		72.37	**72.40**	**71.89**	72.54	72.36	72.32	81.86	81.70	81.92
	FEDAVG	✗	CUTOUT	61.12	64.47	64.20	66.45	69.09	68.99	**83.77**	83.91	**84.31**
	FEDSAM	✗		63.69	66.30	67.25	67.66	71.39	70.67	83.03	83.84	83.49
	FEDASAM	✗		68.50	69.26	69.75	69.23	**71.91**	**71.28**	83.73	**84.10**	84.00
	FEDAVG	✓		65.54	65.60	65.79	69.94	69.55	69.63	83.35	83.39	83.64
	FEDSAM	✓		69.40	68.45	67.36	71.36	71.56	70.99	82.61	82.75	82.52
	FEDASAM	✓		**71.30**	**71.12**	**70.91**	**72.79**	71.76	71.09	83.06	83.31	83.11

However, when combined with the proposed methods, performance improvements can be observed in heterogeneous scenarios compared to the baseline with data augmentation (FEDAVG + data augmentation). Furthermore, the incorporation of SWA yields a significant performance boost, suggesting its potential to facilitate the effective use of data augmentation techniques in FL.

Comparison with the State of the Art

Here, FEDSAM and FEDASAM are compared with several state-of-the-art (SOTA) FL algorithms, including FEDPROX [45], SCAFFOLD [280], FEDAVGM [290], FEDDYN [288], and ADABEST [8]. The comparison is conducted both for the baseline algorithms and for their combinations with SAM, ASAM, and SWA (Table 4.8).

FEDPROX introduces a proximal term to the local objective, but as expected from previous research [405, 8], it does not lead to significant improvement. SCAFFOLD employs control variates to mitigate client drift, exchanging parameters twice per

Table 4.8 SOTA comparison on CIFAR10 and CIFAR100 (centralized performance underlined)

	Algorithm	**w/o SWA**				**w/ SWA**			
		$\alpha = 0$		$\alpha = 0.05/0.5$		$\alpha = 0$		$\alpha = 0.05/0.5$	
		5*cl*	20*cl*	5*cl*	20*cl*	5*cl*	20*cl*	5*cl*	20*cl*
CIFAR10	FEDAVG	65.00	68.52	69.24	73.07	69.71	70.19	73.48	73.81
	FEDSAM	70.16	72.90	73.52	76.04	74.97	73.06	76.61	76.22
	FEDASAM	**73.66**	**76.09**	**75.61**	76.98	76.44	**76.36**	76.12	76.86
	FEDAVGM	10.00	10.00	10.00	**78.51**	10.00	10.00	10.00	$\underline{\mathbf{84.00}}$
	FEDPROX	62.72	68.44	68.38	73.02	70.56	70.08	74.27	73.67
	SCAFFOLD	32.25	15.56	54.46	44.76	11.98	10.00	33.25	24.11
	FEDDYN	67.69	73.81	71.36	75.20	77.00	74.00	77.99	75.12
	ADABEST	66.77	72.29	69.84	75.89	**78.94**	76.12	**80.35**	79.35
	FEDAVGM +ASAM	77.30	$\underline{\mathbf{84.89}}$	77.06	$\underline{\mathbf{84.92}}$	80.88	$\underline{85.98}$	78.29	$\underline{\mathbf{86.03}}$
	FEDPROX +ASAM	73.74	75.76	75.32	77.03	76.89	75.92	76.65	76.95
	SCAFFOLD +ASAM	**77.78**	77.93	77.59	77.80	75.66	75.30	75.32	75.29
	FEDDYN +SAM	77.38	81.00	**79.18**	81.70	**83.81**	$\underline{\mathbf{86.07}}$	**83.18**	$\underline{85.57}$
	ADABEST +ASAM	77.48	78.43	78.41	79.72	82.00	80.80	81.87	80.81
CIFAR100	FEDAVG	30.25	38.59	40.43	42.17	39.34	39.85	43.90	42.09
	FEDSAM	31.04	38.56	44.73	46.05	39.30	39.24	47.96	46.47
	FEDASAM	**36.04**	**40.81**	**45.61**	**47.78**	42.01	41.62	**49.17**	48.27
	FEDAVGM	1.00	40.64	4.60	**47.88**	1.00	$\underline{\mathbf{53.50}}$	4.60	$\underline{\mathbf{53.69}}$
	FEDPROX	31.20	38.59	39.53	42.17	39.06	39.68	43.98	41.84
	SCAFFOLD	1.00	1.00	33.26	1.00	1.00	1.00	5.76	1.00
	FEDDYN	1.00	1.40	22.03	24.75	1.00	1.40	8.27	35.15
	ADABEST	29.90	39.11	36.93	43.25	**44.48**	44.21	48.20	44.51
	FEDAVGM + ASAM	1.00	39.61	4.60	$\underline{\mathbf{51.65}}$	1.00	$\underline{\mathbf{51.58}}$	4.60	$\underline{\mathbf{56.19}}$
	FEDPROX +ASAM	36.10	40.91	44.81	48.17	43.90	42.06	48.66	48.19
	SCAFFOLD +ASAM	**43.65**	42.61	**46.50**	46.76	40.63	39.07	44.87	44.28
	FEDDYN + ASAM	22.16	23.51	38.43	38.60	17.51	19.22	38.60	31.06
	ADABEST + ASAM	39.75	**45.00**	45.25	49.56	$\underline{\mathbf{51.75}}$	47.42	$\underline{\mathbf{51.89}}$	$\underline{51.47}$

round. While achieving comparable performance to FEDAVG in the homogeneous setting (84.5% on CIFAR10 and 51.9% on CIFAR100), its effectiveness suffers significantly under data heterogeneity. Similar behavior is observed for FEDAVGM.

FEDDYN dynamically aligns local and global stationary points. However, as noted in [8], it is susceptible to parameter explosion. Although achieving good results on the simpler CIFAR10 dataset, it requires heavy gradient clipping and cannot complete training on CIFAR100. ADABEST was proposed as a solution, surpassing FEDAVG by a small margin.

The results demonstrate the consistent effectiveness of FEDASAM compared to the SOTA baselines. It improves accuracy by approximately 6% points over the best SOTA on both datasets. Furthermore, incorporating ASAM leads to notable performance gains for all FL algorithms. In particular, FEDAVGM and SCAFFOLD are enabled to train successfully in most highly heterogeneous settings. FEDDYN achieves nearly doubled accuracy on CIFAR100, even with the limitations imposed by gradient clipping. Finally, the best results are achieved with FEDASAM+SWA, which

Table 4.9 Accuracy Results (%) on LANDMARKS-USER-160K with FEDSAM, FEDASAM and SWA

	@5k **rounds**	**w/ SWA @75%**	**w/ SWA @100%**
FEDAVG	61.91	66.05	67.52
FEDSAM	63.72	67.11	68.12
FEDASAM	64.23	67.17	**68.32**
Centralized		74.03	

stabilizes the learning process and allows models to converge close to centralized performance with $\alpha = 0$.

4.3.4 Results in Real-World Vision Scenarios

This section explores the applicability of the proposed method in real-world scenarios beyond the standard small-scale classification tasks. The vision tasks of interest are large-scale classification, semantic segmentation (SS) for autonomous driving and domain generalization (DG).

Large-scale classification

To assess the performance of the proposed methods (SAM, ASAM, and SWA) in a more realistic setting, the analysis is extended to the challenging LANDMARKS-USER-160K [278]. This dataset incorporates real-world complexities such as non-*i.i.d.* distribution of labels across clients and dataset size imbalance. For additional information, refer to Section 3.4.1. The results, presented in Table 4.9, confirm the significant benefits of applying client-side sharpness-aware optimizers (SAM and ASAM) in these challenging scenarios. These improvements are further amplified when combined with server-side weight averaging through SWA, especially if applied at the end of training. This combination achieves final accuracy improvements of up to 7% compared to the baseline.

Semantic Segmentation for Autonomous Driving

Semantic segmentation plays a critical role in autonomous driving applications, requiring accurate identification and localization of objects within a scene. Given

Table 4.10 Federated SS on Cityscapes and IDDA. Results in mIoU (%) @ 1.5k rounds

		IDDA				Cityscapes
Algorithm	**Uniform**	**Country**		**Rainy**		**mIoU**
		seen	*unseen*	*seen*	*unseen*	
FedAvg	✓	63.31	48.60	**65.16**	27.38	43.61
FedSam	✓	**64.22**	**49.74**	64.81	30.00	44.58
FedAsam	✓	62.74	48.73	64.74	**31.32**	**45.86**
FedAvg+Swa	✓	**63.91**	43.28	63.24	47.72	45.64
FedSam+Swa	✓	62.26	**46.26**	**63.69**	48.40	45.29
FedAsam+Swa	✓	60.78	44.23	63.18	**51.76**	**45.69**
FedAvg	✗	42.06	36.04	39.50	24.59	38.65
FedSam	✗	43.28	**37.83**	39.65	29.27	41.22
FedAsam	✗	**43.67**	36.11	**41.68**	**30.07**	**42.27**
FedAvg+Swa	✗	37.16	37.48	37.06	42.33	42.48
FedSam+Swa	✗	44.26	**40.45**	38.15	45.25	**43.42**
FedAsam+Swa	✗	**45.23**	39.72	**42.09**	**45.40**	43.02
SiloBN	✗	45.86	32.77	48.09	39.67	45.96
SiloBN + SAM	✗	**46.88**	33.71	48.22	40.08	49.10
SiloBN + ASAM	✗	46.57	**35.22**	**48.33**	**40.76**	**49.75**

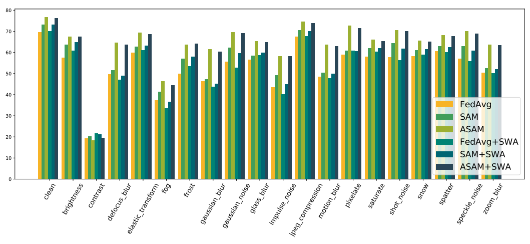
the privacy concerns associated with data collected by self-driving cars, FL emerges as a promising approach for training SS models while preserving data privacy [271].

This work leverages FedDrive [271], a benchmark specifically designed for evaluating autonomous driving tasks in FL settings. The evaluation employs two datasets: Cityscapes and IDDA, under both uniform (similar data distributions across clients) and heterogeneous (diverse data distributions) scenarios. Cityscapes' test set includes unseen cities, challenging the model's ability to generalize to novel environments. The IDDA dataset is further divided into two sub-scenarios to assess generalization under both semantic and appearance shifts. The former involves images captured in the countryside, and the latter focuses on rainy conditions. The model is evaluated on both previously encountered and unseen domains.

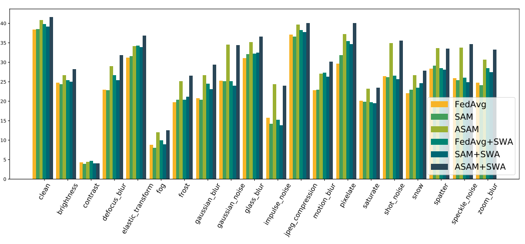
The results presented in Table 4.10 demonstrate that ASAM achieves the best performance on both Cityscapes and the heterogeneous IDDA setting. To further enhance performance, the research combines ASAM with SWA and SiloBN [191]. SiloBN maintains local Batch Normalization statistics on each client while sharing learnable parameters across domains, promoting better adaptation to unseen data. This combination achieves the state-of-the-art performance in this evaluation.

Domain Generalization

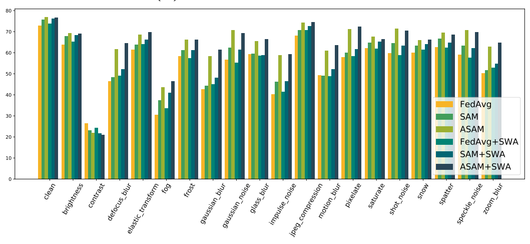
To evaluate the generalization performance of models trained with SAM, ASAM, and SWA, they are tested on corrupted versions of the CIFAR datasets, namely CIFAR10-C and CIFAR100-C. The test images are subjected to 19 different corruptions,



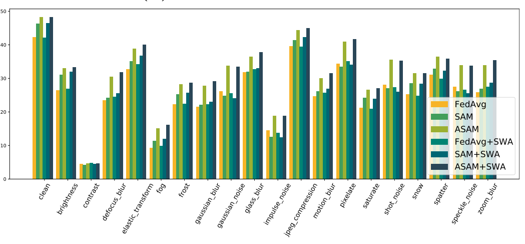
(a) CIFAR10-C $\alpha = 0$



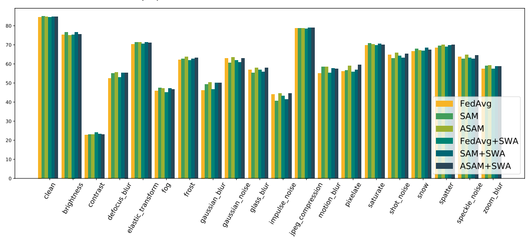
(b) CIFAR100-C $\alpha = 0$



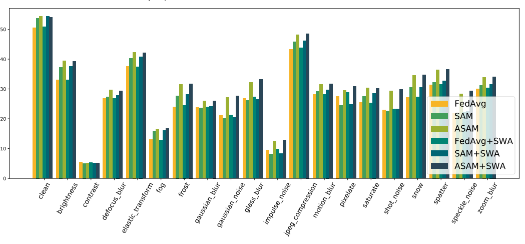
(c) CIFAR10-C $\alpha = 0.05$



(d) CIFAR100-C $\alpha = 0.5$



(e) CIFAR10-C $\alpha = 100$



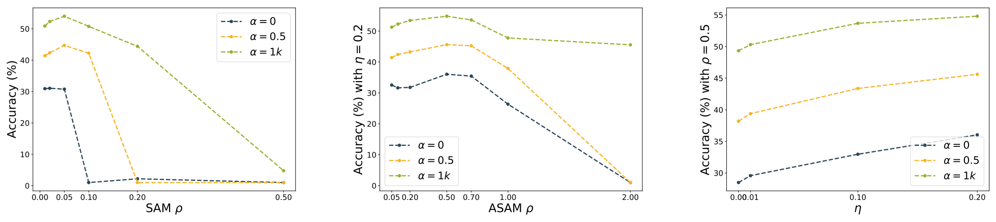
(f) CIFAR100-C $\alpha = 1000$

Fig. 4.13 Domain generalization in FL with FEDSAM, FEDASAM and SWA. Results with 20 clients, severity level 5 on CIFAR10-C (*left*) and CIFAR100-C (*right*), under various heterogeneity (α).

each with 5 severity levels. Results for the highest severity level are presented in Section 4.3.4 and further support the effectiveness of seeking flat minima in FL.

4.3.5 Ablation Studies

Sensitivity to ρ. This analysis examines the sensitivity of FEDASAM and FEDSAM to their hyperparameters using the CIFAR100 dataset with 5% client participation as a reference. The analysis focuses on the neighborhood size ρ and ASAM's stability-adaptivity trade-off parameter η. High ρ values in FEDSAM lead to a rapid performance decline (Figure 4.14a), suggesting better performance with smaller neighborhoods. Conversely, FEDASAM exhibits greater robustness to ρ, maintaining performance even with larger values (up to 0.5) in Figure 4.14b. Instead, as η



(a) SAM ρ (b) ASAM ρ (c) ASAM η

Fig. 4.14 FEDSAM and FEDASAM sensibility to hyperparameters

increases in Figure 4.14c, performance improves linearly, indicating a more adaptive approach potentially leading to better results.

SWA components. SWA is sensible to two main parameters: the cycle length and the starting round. SWA's learning rate decreases from γ_1 to γ_2 over a cycle of length c (Equation (2.37)). To understand the main component affecting the robustness proper of SWA the most, Table 4.11 compares SWA with constant ($c = 1$) and cycling learning rate ($c > 1$), together with the standard FEDAVG changing the clients' learning rate based on $c > 1$. The results indicate that the server-side weight averaging plays a more significant role in achieving stability and improved performance. While the cyclical learning rate offers some benefit (better performance compared to constant learning rate), its impact is less pronounced. This finding suggests that the weight averaging is the core contributor to SWA's effectiveness. However, the cyclical learning rate is still preferred in further experiments due to its slight performance improvement in challenging scenarios (low α and low client participation) observed in the CIFAR100 dataset.

Table 4.12 investigates the impact of the starting round, *i.e.*, the amount of pre-training needed before the deployment of SWA. The table shows the performance difference when applying SWA from various points in the training process (starting at 5%, 25%, 50%, and 75% of total training) to FEDAVG with 5 clients per round. The results suggest longer pre-training periods lead to greater effectiveness of SWA.

4.3.6 Discussion

In conclusion, this work investigated the performance degradation and training slowdown observed in heterogeneous FL due to the limited generalization ability of the learned global model. Inspired by recent research linking convergence to flat

Table 4.11 SWA ablation study: comparison between cyclical ($c > 1$) and constant learning rate ($c = 1$) and contribution given by averaging stochastic weights. Highlighted in bold the best result for each combination (algorithm, α, participating clients).

Dataset	**Algorithm**	**Weights Avg**	c	$\alpha = 0$			$\alpha = 0.5/0.05$			$\alpha = 1k/100$		
				5*cl*	10*cl*	20*cl*	5*cl*	10*cl*	20*cl*	5*cl*	10*cl*	20*cl*
	FEDAVG			**39.34**	39.74	39.85	**43.90**	**44.02**	42.09	50.98	50.87	50.92
	FEDSAM	✓	20	**39.30**	**39.51**	39.24	**47.96**	**46.76**	**46.47**	53.90	53.67	**54.36**
	FEDASAM			42.01	**42.64**	41.62	**49.17**	**48.72**	48.27	53.86	54.79	54.10
	FEDAVG			38.86	**39.82**	**40.19**	43.86	43.93	**42.67**	**51.33**	**51.05**	**51.11**
CIFAR100	FEDSAM	✓	1	38.58	39.20	**39.37**	47.29	46.34	46.40	53.88	**53.70**	**54.36**
	FEDASAM			**42.50**	42.40	**41.76**	48.67	48.50	47.95	54.16	**55.07**	54.19
	FEDAVG			30.68	34.86	37.42	40.34	42.40	41.89	50.06	50.21	50.81
	FEDSAM	✗	20	31.51	35.87	37.81	44.08	45.80	46.43	53.76	53.46	54.28
	FEDASAM			36.85	39.76	41.03	46.34	48.06	**48.38**	54.21	55.06	54.22
	FEDAVG			30.25	36.74	38.59	40.43	41.27	42.17	49.92	50.25	50.66
	FEDSAM	✗	1	31.04	36.93	38.56	44.73	44.84	46.05	**54.01**	53.39	53.97
	FEDASAM			36.04	39.76	40.81	45.61	46.58	47.78	**54.81**	54.97	**54.50**
	FEDAVG			69.71	69.54	70.19	73.48	72.80	**73.81**	84.35	84.32	84.47
	FEDSAM	✓	10	74.97	73.73	73.06	76.61	75.84	76.22	84.23	84.37	84.63
	FEDASAM			76.44	**75.51**	76.36	76.12	76.16	76.86	84.88	84.80	**84.79**
	FEDAVG			**69.88**	**69.83**	**70.72**	**73.91**	**73.12**	73.07	**84.90**	84.47	**84.67**
CIFAR10	FEDSAM	✓	1	**75.17**	**74.00**	**73.53**	**76.93**	**76.06**	**76.55**	**84.53**	84.54	84.77
	FEDASAM			**76.80**	75.48	**76.84**	**76.87**	**76.30**	**77.55**	**85.09**	**85.06**	84.73
	FEDAVG			61.41	63.96	67.39	67.17	69.88	72.19	84.18	84.15	84.45
	FEDSAM	✗	10	70.66	71.14	73.04	73.93	74.96	76.20	84.23	84.40	84.69
	FEDASAM			75.07	74.87	76.37	75.37	76.17	77.14	84.68	84.72	84.71
	FEDAVG			65.00	65.54	68.52	69.24	72.50	73.07	84.46	**84.50**	84.59
	FEDSAM	✗	1	70.16	71.09	72.90	73.52	74.81	76.04	84.58	**84.67**	**84.82**
	FEDASAM			73.66	74.10	76.09	75.61	76.22	76.98	84.77	84.72	84.75

minima in the loss landscape to generalization in deep learning, the model's behavior was analyzed through the lens of loss surface geometry, linking poor generalization in FL and convergence towards sharp minima. To address this gap, this work introduced Sharpness-Aware Minimization (SAM), its adaptive variant (ASAM), and Stochastic Weight Averaging (SWA) for FL, all of which encourage convergence towards flatter minima. The effectiveness of this approach was demonstrated on several real-world vision tasks (small and large scale image classification, semantic segmentation, domain generalization) and datasets.

Impact

Following the success of FEDSAM, several follow-up works explored new insights and enhancements. In Qu *et al.* [406], concurrent to FEDSAM, momentum is leveraged to improve the efficacy of SAM and convergence guarantees are provided. FEDSPEED [407] utilizes perturbed gradients as SAM, aiming to reduce local overfitting. FEDGAMMA [408] combines the stochastic variance reduction of SCAFFOLD with SAM. Moreover, Shi *et al.* [409] demonstrate FEDSAM's effectiveness in mitigating the negative effects of differential privacy, while DFEDSAM applies FEDSAM

Table 4.12 SWA ablation study: comparison in accuracy (%) between SWA starting rounds, with 5 clients per round. Server-side aggregation with FEDAVG.

Dataset	c	**Start round**	**Test Accuracy**		
			$\alpha = 0$	$\alpha = 0.5/0.05$	$\alpha = 1k/100$
CIFAR100	20	1000	24.53	34.52	49.38
		5000	30.66	39.71	**51.52**
		10000	36.21	42.55	51.01
		15000	**39.34**	**43.90**	50.98
CIFAR10	10	500	55.57	60.50	79.09
		2500	60.34	65.72	81.49
		5000	66.22	70.55	83.79
		7500	**69.71**	**73.48**	**84.35**

to distributed FL [410] and FEDKSAM to intrusion detection systems [411]. In [412], an approach similar to FEDSAM is used for multi-task learning.

However, all these approaches primarily rely on *local* sharpness information, assuming that minimizing local sharpness directly translates to a globally flat minimum. This assumption may not always hold true, as discrepancies often exist between the geometries of local and global loss surfaces. Optimizing local sharpness alone does not guarantee a resulting global model residing in a flat region of the *global* loss landscape. Addressing these limitations, FEDSMOO [413] leverages ADMM to enforce consistency between global and local sharpness, at the cost of doubled communication. The next section proposes an alternative solution to achieve global flatness while maintaining communication efficiency.

Limitations

Despite their effectiveness on various tasks and outperforming state-of-the-art performance, FEDSAM, FEDASAM and SWA present some limitations.

As discussed in Section 4.1, FEDSAM introduces two key drawbacks. Firstly, as highlighted in [413], FEDSAM assumes that local optimization for flat minima translates to achieving globally flat regions. However, this may not always hold true, particularly in heterogeneous settings where local and global loss landscapes exhibit greater inconsistencies. Secondly, it requires double the computation compared to optimizers like SGD due to the alternating ascent-descent steps. This can be a significant hurdle for resource-constrained edge devices.

As shown in Table 4.12, the deployment of SWA instead is limited to later training stages, rendering it unusable for most of the training process. This hinders its applicability to broader scenarios.

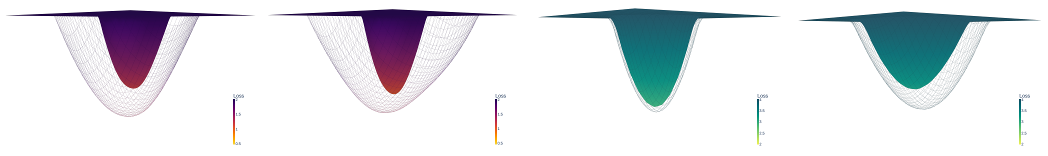
4.4 Beyond Local Sharpness: Communication-Efficient Global Sharpness-aware Minimization for Federated Learning

Data heterogeneity across edge devices (clients) can cause models to converge to sharp minima, negatively impacting generalization and robustness. Recent approaches use client-side sharpness-aware minimization (SAM) to encourage flatter minima, but the discrepancy between local and global loss landscapes often undermines their effectiveness, as optimizing for local sharpness does not ensure global flatness. This work introduces FEDGLOSS (**Fed**erated **Glo**bal **S**erver-side **S**harpness), a novel FL approach that directly optimizes for global sharpness on the server using SAM. To reduce communication overhead, FEDGLOSS cleverly approximates sharpness using the previous global gradient, eliminating the need for additional client communication.

4.4.1 Motivation

Recent approaches like FEDSAM and its variants [407, 408] belong to a recent trend leverages the geometry of the loss landscape to improve generalization [7, 406–408, 413]. These methods build upon the notion that convergence to sharp minima correlates with poor generalization [204, 52, 414] and have shown their effectiveness in various FL scenarios. However, they share a critical limitation: their dependence solely on local sharpness information. The underlying assumption is that minimizing sharpness during local training directly translates to achieving a globally flat minimum. In real-world scenarios with significant data heterogeneity (Figure 4.15), there can be substantial discrepancies between the local and global loss landscapes. As a consequence, **optimizing for local sharpness does not guarantee that the global model will reside in a flat region**.

Addressing these limitations, FEDSMOO [413] leverages the alternating direction method of multipliers (ADMM) [289] to include global sharpness information in SAM's local training. While this approach reduces the inconsistency between local and global geometries, it increases communication cost by **requiring double the bandwidth** in each round. This hinders its real-world applicability, as FL relies on minimizing communication overhead (*i.e.*, both message size and exchange



(a) CIFAR10 $\alpha = 0$ (b) CIFAR10 $\alpha = 0.05$ (c) CIFAR100 $\alpha = 0$ (d) CIFAR100 $\alpha = 0.5$

Fig. 4.15 Comparison of **FEDAVG** (*solid*) and **FEDSAM** (*net*) loss landscapes with varying degrees of data heterogeneity (α) on the CIFAR datasets. **FEDSAM's effectiveness in converging to *global* flat minima is highly influenced by the data heterogeneity**, where higher heterogeneity ($\alpha \to 0$) leads to sharper minima, and the complexity of the task, *e.g.*, higher sharpness for the more complex CIFAR100. This highlights the importance of optimizing global sharpness. Model: CNN.

frequency) to avoid network congestion and account for potential connection failures, that are common in practical deployments.

Given the limitations of existing methods, achieving convergence to global flat minima while maintaining communication efficiency in heterogeneous FL remains a critical challenge. To address this, this research introduces FEDGLOSS (**Fed**erated **Glo**bal **S**erver-side **S**harpness), that directly **optimizes global sharpness by using SAM on the server side**, avoiding additional exchanges over the network. Such adaptation is not straightforward, as SAM would require dual exchanges with each client set per round to solve its optimization problem. Instead, FEDGLOSS approximates the sharpness measure using available **previous pseudo-gradients**. As a result, it facilitates faster training and keeps communication efficiency.

To summarize, this work contributes to the current research field as follows:

- **Empirical proof of local-global discrepancies**: it provides the first empirical evidence showing the limitations of approaches that focus solely on local sharpness. The proposed analysis highlights the **inconsistency between local and global loss geometries** even when using sharpness-aware approaches like FEDSAM, demonstrating that local flatness does not necessarily ensure a flat global minimum. While reaching flat global solutions in simpler problems, this work shows that their effectiveness diminishes as data complexity and heterogeneity increase (Figure 4.15).

  Extensive evaluations show that FEDGLOSS consistently outperforms state-of-the-art methods across various benchmarks and achieves flatter minima, validating its effectiveness without incurring extra communication.

- To bridge this gap and motivated by **communication efficiency**, the proposed novel FEDGLOSS directly optimizes for **global sharpness** on the server using SAM, reducing the communication overhead and the clients' computational costs compared to previous works. FEDGLOSS consistently achieves flatter minima and outperforms state-of-the-art methods across various vision benchmarks.

- This work underlines the importance of aligning global and local solutions and illustrates how SAM, especially on the server side, enables **effective ADMM use in FL**. While typically ADMM-based methods suffer from parameter explosion [8], the introduced findings indicate that by targeting flat minima, SAM encourages smaller gradient steps and minimal weight updates, leading to a significantly more stable algorithm.

4.4.2 The Inconsistency between Local and Global Sharpness

This section empirically investigates the hypothesis that discrepancies between local and global loss landscapes impact FEDSAM's performance, using a CNN model on CIFAR10 and CIFAR100 datasets.

Figure 4.15 compares the loss surfaces of CNNs trained with FEDAVG and FEDSAM on CIFAR10 and CIFAR100 under varying data heterogeneity. On the easier CIFAR10, FEDSAM exhibits noticeably flatter minima w.r.t. FEDAVG, suggesting its effectiveness in navigating simpler landscapes. However, their performance difference diminishes with increasing dataset complexity (CIFAR100) and heterogeneity ($\alpha \rightarrow 0$). This suggests that **larger discrepancies between local and global geometries arise as tasks become more complex and data distributions more diverse**. The observed difficulty in reaching flat regions under high heterogeneity (smoother landscapes with $\alpha = 0.05$ and 0.5) further motivates the introduction of FEDGLOSS.

To highlight the existing difference between local and global behavior, Figure 4.16 investigates the behavior of client models at the end of local training when tested on their own data $\mathscr{D}_k$ (*bottom* landscape), prior to server-side aggregation, w.r.t. the overall dataset $\mathscr{D}$ (*top* landscape). Each plot (Figures 4.16a to 4.16e) depicts the behavior of one of the five randomly selected clients during the last training round with FEDSAM, distinguished by the locally seen class. The inconsistency between local and global behavior can be easily spotted: locally, each model lands in a flat

Table 4.13 **Maximum Hessian eigenvalues of local models**, computed on global ($\lambda_{1,g}$) and local datasets ($\lambda_{1,l}$) with CIFAR10 $\alpha = 0$ and CIFAR100 $\alpha = 0$. Each client is identified via its local `class`. The lowest $\lambda_{1,g}$ in **bold**. FEDDYN does not converge on CIFAR100 with $\alpha = 0$ [7, 8], hence the lack of results (✗).

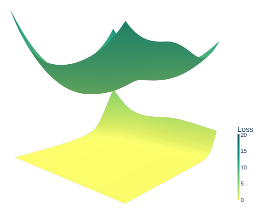
	Local Class	FEDAVG		FEDSAM		FEDDYN		FEDDYN + SAM		FEDSMOO		FEDGLOSS (ours)	
		$\lambda_{1,l}$	$\lambda_{1,g}$	$\lambda_{1,l}$	$\lambda_{1,g}$	$\lambda_{1,l}$	$\lambda_{1,g}$	$\lambda_{1,l}$	$\lambda_{1,g}$	$\lambda_{1,l}$	$\lambda_{1,g}$	$\lambda_{1,l}$	$\lambda_{1,g}$
CIFAR10	`airplane`	9.1	239.1	100.6	36.4	752.5	347.8	199.6	12.0	122.1	26.5	190.1	**4.3**
	`cat`	424.2	273.6	28.8	16.5	59.9	242.3	122.0	11.1	82.4	26.9	106.9	**3.9**
	`bird`	18.4	237.0	106.4	35.7	894.0	371.2	200.2	12.0	134.2	25.7	200.1	**4.1**
	`airplane`	483.5	269.5	103.2	30.6	761.6	348.9	206.9	12.3	122.8	25.2	207.8	**4.0**
	`frog`	263.2	259.6	68.1	32.9	528.9	286.0	155.6	11.7	79.3	33.5	84.8	**4.1**
CIFAR100	`sea`	251.0	224.5	0.1	238.5					33.2	31.4	28.3	**19.6**
	`snail`	91.2	267.0	0.2	149.1					331.2	102.2	260.8	**40.7**
	`bear`	108.4	215.2	6.7	129.3	✗	✗	✗	✗	428.6	121.0	220.3	**49.6**
	`skyscraper`	613.3	300.1	1.3	194.6					143.5	40.2	269.2	**22.2**
	`possum`	37.9	259.6	15.3	142.6					455.5	90.9	392.4	**39.0**

region; differently, the same model is found close to saddle points (Figures 4.16a and 4.16e), or sharp minima (Figures 4.16b and 4.16d) in the global loss landscape. For completeness, Figure 4.16f shows FEDSAM's global model.

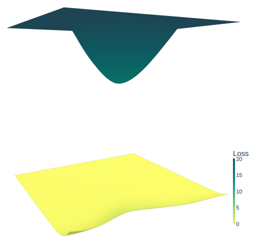
These findings are further corroborated by the Hessian eigenvalues presented in Table 4.13. FEDSAM's local maximum Hessian eigenvalue, denoted by $\lambda_{1,l}$ and computed on each client's individual dataset, is significantly lower than the global eigenvalue $\lambda_{1,g}$, calculated on the overall dataset, on the more complex CIFAR100. This suggests that FEDSAM effectively achieves *local* convergence to flatter regions of the loss landscape on individual devices. However, the higher global eigenvalue indicates limitations in reaching a *globally* flat minimum. The challenge of achieving flat regions under high heterogeneity and the gap between local and global flatness support the introduction of FEDGLOSS.

4.4.3 Rethinking SAM in Federated Learning

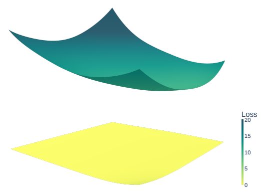
In Equation (2.29), SAM is directly used to solve the local optimization problem, without explicitly optimizing for global flatness. As shown in the previous section, this can lead to possible discrepancies between local and global geometries of the loss surface, which are exacerbated in non-*i.i.d.* settings. This work addresses these limitations of FEDSAM by proposing FEDGLOSS (Federated Global Server-side Sharpness), which optimizes both global sharpness and global consistency while maintaining communication efficiency.



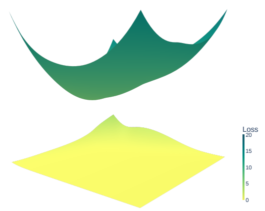
(a) Local model trained on class `sea` with FEDSAM



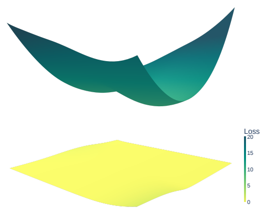
(b) Local model trained on class `snail` with FEDSAM



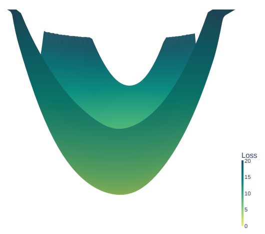
(c) Local model trained on class `bear` with FEDSAM



(d) Local model trained on class `skyscraper` with FEDSAM



(e) Local model trained on class `possum` with FEDSAM



(f) FEDSAM global model

Fig. 4.16 **Global *vs.* local perspective on FEDSAM**. CIFAR100 $\alpha = 0$ with SAM as local optimizer @ 20k rounds on CNN. *(a) - (e):* Local models trained on one `class`, tested on the local (*bottom landscape*) or global dataset (*top landscape*). *(f):* Resulting global model @ $t = 20k$, computed using the reported clients' models. Models trained with FEDSAM present significant differences between local and global behaviors.

Aiming to optimize SAM's objective (Equation (2.29)) on the global function, FEDGLOSS modifies Equation (4.1) as follows

$$\min_{\boldsymbol{w}} \left\{ \mathscr{F}(\boldsymbol{w}) = \frac{1}{K} \sum_{k \in \mathscr{C}} \mathscr{F}_k(\boldsymbol{w}) \right\}, \mathscr{F}_k(\boldsymbol{w}) \triangleq \max_{\|\boldsymbol{\epsilon}\| \leq \rho} f_k(\boldsymbol{w} + \boldsymbol{\epsilon}), \tag{4.5}$$

where $\boldsymbol{\epsilon}$ is the global perturbation. Calculating the true $\boldsymbol{\epsilon}$ value requires the global gradient $\nabla_{\boldsymbol{w}} f$ (Equation (2.31)) computed on the entire dataset $\mathscr{D} \triangleq \cup_{k \in \mathscr{C}} \mathscr{D}_k$, which is not directly available in FL due to local data privacy and communication constraints. While FEDSMOO [413] tackles this issue by using ADMM on the sharpness with the constraint $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_k$, it necessitates transmitting $\boldsymbol{\epsilon}$ alongside the model parameters $\boldsymbol{w}$ both to clients and back to the server in each round, hindering its practicality in real-world scenarios with limited communication budgets. This observation motivates the question: *how to minimize the global sharpness while maintaining communication efficiency in FL*?

Challenges of Server-side SAM

To address this question, this work proposes applying SAM on the server side. This approach directly optimizes global sharpness, eliminating the need to align local sharpness on the clients. At each round t, the global model has to be updated as $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta_g \nabla_{\boldsymbol{w}} \mathscr{F}(\boldsymbol{w})|_{\boldsymbol{w}^t + \hat{\boldsymbol{\varepsilon}}^t(\boldsymbol{w})}$, where $\hat{\boldsymbol{\varepsilon}}^t$ is the global perturbation at round t. However, a key challenge arises: the computation of both $\hat{\boldsymbol{\varepsilon}}^t$ and the sharpness-aware gradient necessitates two transmissions with the clients, making its direct application in server-side FL non-trivial. A straightforward solution is to emulate SAM's double computation step through two communication exchanges within each round t.

- **Step 1**: the server selects a subset $\mathscr{C}^t$ of clients, sends them the global model $\boldsymbol{w}^t$, which they update using their local data. The resulting pseudo-gradient serves as an estimate of the global gradient, *i.e.*, $\Delta^t_{\boldsymbol{w}} \approx \nabla_{\boldsymbol{w}} \mathscr{F}(\boldsymbol{w}^t)$, and is used to compute $\hat{\boldsymbol{\varepsilon}}^t(\boldsymbol{w}) = \rho(\Delta^t_{\boldsymbol{w}}/\|\Delta^t_{\boldsymbol{w}}\|)$ and the perturbed model $\tilde{\boldsymbol{w}}^t = \boldsymbol{w}^t + \hat{\boldsymbol{\varepsilon}}^t(\boldsymbol{w})$.
- **Step 2**: the server transmits $\tilde{\boldsymbol{w}}^t$ to the *same* $\mathscr{C}^t$. Each client k computes its update $\tilde{\boldsymbol{w}}^t_k$ on the perturbed model, and the consequent global pseudo-gradient $\tilde{\Delta}^t_{\boldsymbol{w}} \triangleq \sum_{k \in \mathscr{C}^t} N_k/N(\tilde{\boldsymbol{w}}^t - \tilde{\boldsymbol{w}}^t_k)$ is an estimate of the sharpness-aware gradient, $\nabla_{\boldsymbol{w}} \mathscr{F}(\boldsymbol{w})|_{\boldsymbol{w}^t + \hat{\boldsymbol{\varepsilon}}^t(\boldsymbol{w})}$.

This two-step approach, referred to as NAIVEFEDGLOSS, is conceptually simple but suffers from communication *in*efficiency, doubling the communication and computational cost compared to the original server-side with FEDAVG. In addition, this method requires the same subset of clients $\mathscr{C}^t$ to remain active for two consecutive exchanges. This may be unrealistic in real-world settings often characterized by network failures. These limitations highlight the need for an efficient alternative that accounts for practical real-world FL factors.

4.4.4 Federated Global Server-side Sharpness

To overcome the challenges posed by NAIVEFEDGLOSS, following [219], FEDGLOSS estimates $\hat{\boldsymbol{\varepsilon}}^t$ using the **perturbed global pseudo-gradient from the previous**

Table 4.14 Overview of FL methods using SAM. Differently from previous works, FEDGLOSS uses SAM as server optimizer and allows any local optimizer.

Method	SERVEROPT	CLIENTOPT	Global Flatness	Communication Cost	Local Computation Cost
FEDSAM [7, 406]	SGD	SAM	✗	1×	2×
FEDDYN [288] + SAM	SGD	SAM	✗	1×	2×
FEDSPEED [407]	SGD	Similar to SAM	✗	1×	2×
FEDGAMMA [408]	SGD	SAM	✓	2×	2×
FEDSMOO [413]	SGD	SAM	✓	2×	2×
FEDGLOSS	**SAM**	**Any optimizer**	✓	1×	1× or 2×

round $\tilde{\Delta}_{\boldsymbol{w}}^{t-1}$ at each round t. This approach leverages available information *without incurring extra communications* and avoiding unnecessary computations. Intuitively, the use of the previous pseudo-gradient to minimize the sharpness allows FEDGLOSS to access information on the *global* loss landscape geometry, thus **guiding the *global* optimization towards flatter minima**. Figure 4.17 depicts the proposed approach.

From Equations (2.31) and (3.5), FEDGLOSS updates the global model $\boldsymbol{w}^t$ as

$$①\,\text{Sharpness approximation with } \tilde{\Delta}_{\boldsymbol{w}}^{t-1}\text{: } \tilde{\epsilon}^t(\boldsymbol{w}) \triangleq \rho \frac{\tilde{\Delta}_{\boldsymbol{w}}^{t-1}}{\|\tilde{\Delta}_{\boldsymbol{w}}^{t-1}\|} \tag{4.6}$$

$$②\,\text{Global model perturbation: } \tilde{\boldsymbol{w}}^t \leftarrow \boldsymbol{w}^t + \tilde{\epsilon}^t(\boldsymbol{w}) \tag{4.7}$$

$$③\,\text{Send } \tilde{\boldsymbol{w}}^t \text{ to the clients, obtain } \tilde{\boldsymbol{w}}_k^t \text{ and compute: } \tilde{\Delta}_{\boldsymbol{w}}^t = \sum_{k \in \mathscr{C}^t} \frac{N_k}{N}(\tilde{\boldsymbol{w}}^t - \tilde{\boldsymbol{w}}_k^t) \tag{4.8}$$

$$④\,\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \text{FEDGLOSS}(\boldsymbol{w}^t, \tilde{\Delta}_{\boldsymbol{w}}^t, \eta_g, t) = \boldsymbol{w}^t - \eta_g \tilde{\Delta}_{\boldsymbol{w}}^t, \tag{4.9}$$

where with a slight abuse of notation SERVEROPT from Equation (3.5) was substituted with the server-side strategy proposed by FEDGLOSS. The notation follows the colors of Figure 4.17 for better understanding.

Notably, as summarized in Table 4.14, FEDGLOSS enables SAM on the server side while **allowing any CLIENTOPT for local training**, with computational costs varying based on the chosen optimizer. This differs from previous methods constrained to the more computationally expensive SAM. In addition, differently from FEDSMOO, FEDGLOSS maintains FEDAVG's communication complexity while optimizing for global flatness.

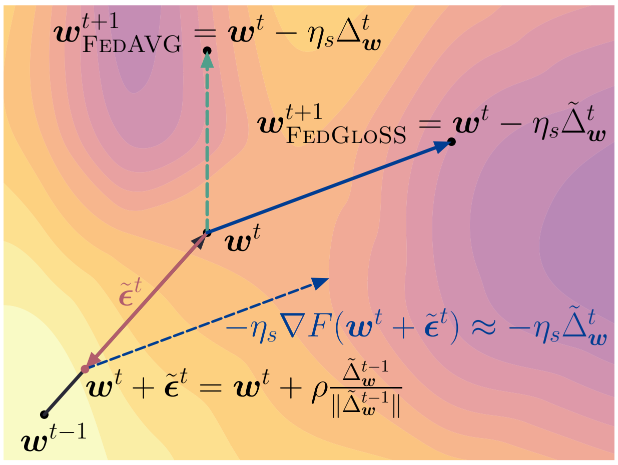

$\boldsymbol{w}^{t+1}_{\text{FEDAVG}} = \boldsymbol{w}^t - \eta_s \Delta^t_{\boldsymbol{w}}$
$\boldsymbol{w}^{t+1}_{\text{FEDGLOSS}} = \boldsymbol{w}^t - \eta_s \tilde{\Delta}^t_{\boldsymbol{w}}$
$\boldsymbol{w}^t$
$\tilde{\boldsymbol{\epsilon}}^t$
$-\eta_s \nabla F(\boldsymbol{w}^t + \tilde{\boldsymbol{\epsilon}}^t) \approx -\eta_s \tilde{\Delta}^t_{\boldsymbol{w}}$
$\boldsymbol{w}^t + \tilde{\boldsymbol{\epsilon}}^t = \boldsymbol{w}^t + \rho \frac{\tilde{\Delta}^{t-1}_{\boldsymbol{w}}}{\|\tilde{\Delta}^{t-1}_{\boldsymbol{w}}\|}$
$\boldsymbol{w}^{t-1}$

Fig. 4.17 Illustration of FEDGLOSS. The model $\mathbf{w}^t$ is perturbed using $\tilde{\Delta}^{t-1}_{\mathbf{w}}$. The sharpness-aware direction (*dashed*) is used to compute $\mathbf{w}^{t+1}$ (*solid*), which lands in a flat region. Compared to FEDAVG.

Promoting global consistency with ADMM

The difference in using the approximated $\tilde{\epsilon}^t$ (FEDGLOSS) and the true $\hat{\epsilon}^t$ (NAIVEFEDGLOSS) is

$$\delta^t_\epsilon \triangleq \|\tilde{\epsilon}^t(\mathbf{w}) - \hat{\epsilon}^t(\mathbf{w})\| = \rho \left\| \frac{\tilde{\Delta}^{t-1}_{\mathbf{w}}}{\|\tilde{\Delta}^{t-1}_{\mathbf{w}}\|} - \frac{\Delta^t_{\mathbf{w}}}{\|\Delta^t_{\mathbf{w}}\|} \right\|, \tag{4.10}$$

where $\tilde{\Delta}^{t-1}_{\mathbf{w}}$ is computed using the updates of the clients in $\mathscr{C}^{t-1}$ and $\tilde{\Delta}^t_{\mathbf{w}}$ with $\mathscr{C}^t$. Equation (4.10) suggests that δ^t_ϵ is minimized when $\tilde{\Delta}^{t-1}_{\mathbf{w}}$ and $\tilde{\Delta}^t_{\mathbf{w}}$ are aligned. However, in real-world heterogeneous FL, *i*) to due clients' unavailability, only a subset of them participates in training at each round, with $\mathscr{C}^t$ likely differing from $\mathscr{C}^{t-1}$, and *ii*) clients hold different data distributions, *i.e.*, local optimization paths likely converge towards different local minima, leading to unstable global updates [280]. As a consequence, $\delta^t_\epsilon \not\to 0$ necessarily. Specifically, due to triangle inequality and normalized gradients norms, $\delta^t_\epsilon \leq 2\rho$, with $\rho < 0.2$ in this case, *i.e.*, larger neighborhoods lead to increased difference.

To align local and global objectives - ensuring client and server gradient alignment and minimizing Equation (4.10) - FEDGLOSS leverages the Alternating Direction Method of Multipliers (ADMM) [289] on $\mathbf{w}^t$ [288, 413, 407]. While alternative approaches could be used, they either lack full immunity to data heterogeneity

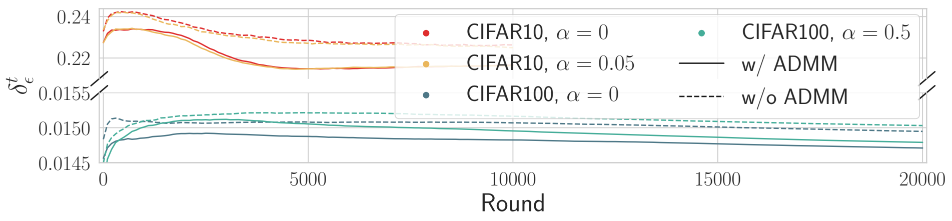


Fig. 4.18 Trend of the difference δ_ϵ^t (Equation (4.10)), which decreases as ADMM is used and over training rounds. CIFAR datasets, CNN.

or have shown poor performance on realistic scenarios (*e.g.*, variance reduction [280, 408]). In contrast, ADMM has been proved to converge under arbitrary heterogeneity [288] and can thus be leveraged as a base algorithm for FEDGLOSS, as shown in Algorithm 5.

ADMM makes use of the augmented Lagrangian function $\mathscr{L}(\boldsymbol{w},\boldsymbol{W},\boldsymbol{\sigma}) = \sum_{k\in\mathscr{C}} L(\boldsymbol{w},\boldsymbol{w}_k,\sigma_k)$ where $\boldsymbol{W} = \{\boldsymbol{w}_1,\cdots,\boldsymbol{w}_C\}$ and $\boldsymbol{\sigma}$ is the Lagrangian multiplier. The problem solved by $\mathscr{L}$ is

$$\frac{1}{K}\sum_{k\in\mathscr{C}}(f_k+\sigma_k^\top(\boldsymbol{w}^t-\boldsymbol{w}_k^t)+\frac{1}{2\beta}\|\boldsymbol{w}^t-\boldsymbol{w}_k^t\|^2)\ s.t.\ \boldsymbol{w}=\boldsymbol{w}_k \tag{4.11}$$

with $\beta > 0$ being an hyperparameter. Equation (4.11) is split into C sub-problems of the form $\boldsymbol{w}_{k,E} = \arg\min_{\boldsymbol{w}_k}\{f_k - \sigma_k^\top(\boldsymbol{w}^t-\boldsymbol{w}_k)+\frac{1}{2\beta}\|\boldsymbol{w}^t-\boldsymbol{w}_k^t\|^2\}$. The local dual variable is updated as $\sigma_k \leftarrow \sigma_k - \frac{1}{\beta}(\boldsymbol{w}_{k,E}^t - \boldsymbol{w}_{k,0}^t)$. The global one σ is updated by adding the averaged $\boldsymbol{w}_k - \boldsymbol{w}^t\ \forall k\in\mathscr{C}$.

Figure 4.18 confirms the effect of ADMM on gradient alignment: the difference between the true and approximated perturbation, δ_ϵ^t (Equation (4.10)), decreases over training rounds and with the use of Lagrangian multipliers.

4.4.5 Experimental Results

The purpose of this section is to outline the datasets, models, and baseline methods used in our experiments. For comprehensive details on the implementation and hyperparameter settings, please refer to Chapter A.

Algorithm 5 FEDGLOSS with SAM or SGD as local optimizers

```
1: Input: Global model $\boldsymbol{w}$, clients $\mathscr{C}$, rounds $T$, local epoch $E$, clients' learning rate $\eta_l$, clients' SAM neighborhood size $\rho_l$, FEDGLOSS neighborhood size $\rho$, Lagrangian hyperparameter $\beta$.
2: Initialize: $\boldsymbol{w}^0$, $\sigma^0 = \sigma_k = 0$, $\Delta_{\boldsymbol{w}}^0 = 0$.
3: for each round $t \in [1,T]$ do
4:     $\tilde{\varepsilon}^t(\boldsymbol{w}) = \rho \frac{\Delta_{\boldsymbol{w}}^{t-1}}{\|\Delta_{\boldsymbol{w}}^{t-1}\|_2}$    ▷ Global model perturbation with past pseudo-gradient
5:     $\tilde{\boldsymbol{w}}^t = \boldsymbol{w}^t + \tilde{\varepsilon}^t(\boldsymbol{w})$    ▷ Server approximated FEDGLOSS ascent step
6:     Randomly select a subset of clients $\mathscr{C}^t \subset \mathscr{C}$
7:     for each client $k \in \mathscr{C}^t$ in parallel do
8:         $\boldsymbol{w}_{k,0} = \tilde{\boldsymbol{w}}^t$    ▷ Initialize local model with *perturbed* global model $\tilde{\boldsymbol{w}}^t$
9:         Set iteration counter $i = 1$
10:        for each epoch $e \in [1,E]$ do
11:            for each batch $\mathscr{B} \in \mathscr{D}_k$ do
12:                $g_{k,i} = \nabla f_{\mathscr{B}}(\boldsymbol{w}_{k,i-1})$    ▷ SGD gradient
13:                $\hat{\varepsilon}_{k,i} = \rho_l \frac{g_{k,i}}{\|g_{k,i}\|_2}$    ▷ SAM local perturbation
14:                $g_{k,i} = \nabla f_{\mathscr{B}}(\boldsymbol{w}_{k,i-1} + \hat{\varepsilon}_{k,i})$    ▷ Local sharpness-aware gradient
15:                $\boldsymbol{w}_{k,i} \leftarrow \boldsymbol{w}_{k,i-1} - \eta_l [g_{k,i} - \sigma_k + (\boldsymbol{w}_{k,i-1} - \boldsymbol{w}_{k,0})/\beta]$    ▷ Local step w/ ADMM
16:                $i \leftarrow i + 1$
17:            end for
18:        end for
19:        $\sigma_k \leftarrow \sigma_k - (\boldsymbol{w}_{k,E} - \tilde{\boldsymbol{w}}^t)/\beta$    ▷ Update the local dual variable
20:        Send back to the server the locally updated model $\boldsymbol{w}_k^t = \boldsymbol{w}_{k,E}$
21:    end for
22:    $\sigma^{t+1} = \sigma^t - \frac{1}{\beta|\mathscr{C}|}\sum_{k\in\mathscr{S}}(\boldsymbol{w}_k^t - \boldsymbol{w}^t)$    ▷ Update the global dual variable
23:    $\tilde{\Delta}_{\boldsymbol{w}}^t = \sum \frac{N_k}{N}(\tilde{\boldsymbol{w}}^t - \boldsymbol{w}_k^t)$    ▷ Compute the global pseudo-gradient
24:    $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \tilde{\Delta}_{\boldsymbol{w}}^t - \beta\sigma^{t+1}$    ▷ FEDGLOSS descent step with pseudo-grad $\tilde{\Delta}_{\boldsymbol{w}}^t$
25: end for
```

Setting

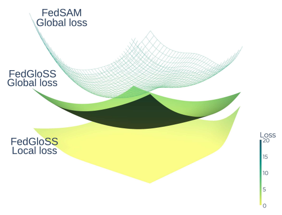
Federated datasets. The proposed experiments leverage established FL benchmarks [394, 278, 290], encompassing a variety of tasks. The federated versions of CIFAR10 (10 classes) and CIFAR100 (100 classes) [113] for small-scale classification, based on the Dirichlet distribution. $\alpha \in \{0, 0.05\}$ for CIFAR10 and $\{0, 0.5\}$ for CIFAR100 [290]. LANDMARKS-USER-160K [278] instead is chosen for large-scale classification (2,028 classes). For additional information on the datasets, please refer to Section 3.4.1.

Models. To demonstrate the effectiveness of FEDGLOSS, experiments are run with different model architectures. As done in [278, 7], the experiments are run using a Convolutional Neural Network (CNN) similar to LeNet5 [112] on both CIFAR10 (10k rounds) and CIFAR100 (20k rounds). Additionally, ResNet18 [114] is deployed on the more complex CIFAR100, running for 10k rounds. For LANDMARKS-USER-160K, MobileNetv2 [415, 278], accounts for the reduced available resources at the edge, with $T = 1.3k$.

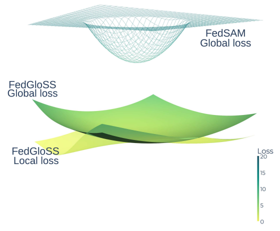
Baselines. To study real-world settings in typical cross-device scenarios, only a small fraction of clients is deemed available at each round. To test varying degrees of client participation, the participation rate is set to 5% with the CNN on both CIFARS and 10% with ResNet18, while 50 clients at each round are selected in LANDMARKS-USER-160K ($\approx 4\%$). FEDGLOSS is compatible with any local optimizer (Section 4.4.4). The chosen algorithms are SGD and SAM to comply with previous works and compare FEDGLOSS with state-of-the-art (SOTA) methods for statistical heterogeneity in FL, distinguishing the results by optimizer type to highlight performance differences. The SGD-based approaches are FEDAVG [34], FEDPROX [45], which adds a regularization term to the local training to encourage proximity between local and global parameters, FEDDYN [288] that uses ADMM for global consistency, while SCAFFOLD [280] leverages stochastic variance reduction to reduce the client drift. The SAM-based methods instead are FEDSAM [7, 406], FEDDYN with SAM as a local optimizer, FEDSPEED [407], which combines SAM on the client-side and ADMM, FEDGAMMA [408] that combines SCAFFOLD with client-side SAM, and FEDSMOO [413], using ADMM for both global sharpness and consistency.

Achieving Local and Global Sharpness Consistency with FedGloSS

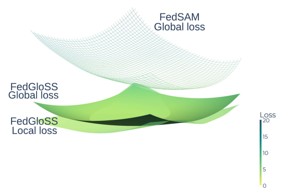
To assess the effectiveness of FEDGLOSS in promoting consistency between local and global loss landscapes, Figure 4.19 replicates the analysis previously conducted on FEDSAM (Figure 4.16) for direct comparison. The behavior of local models is shown from both local and global perspectives (referred to as "*Local loss*" and "*Global loss*", respectively). Additionally, FEDSAM's global perspective is included for reference (*net* surface). Compared to FEDSAM, the gap between local and global loss landscapes in FEDGLOSS is significantly smaller, and both global and local



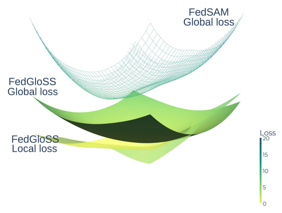
(a) Local model trained on class `sea` with FEDGLOSS



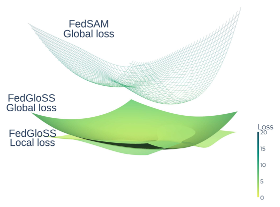
(b) Local model trained on class `snail` with FEDGLOSS



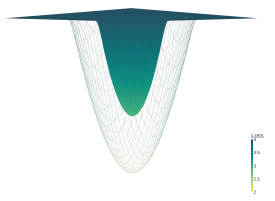
(c) Local model trained on class `bear` with FEDGLOSS



(d) Local model trained on class `skyscraper` with FEDGLOSS



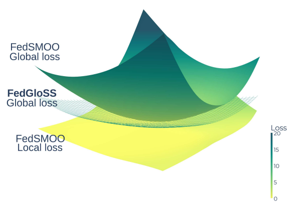
(e) Local model trained on class `possum` with FEDGLOSS



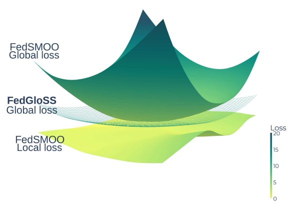
(f) FEDGLOSS (*net*) *vs.* FEDSAM (*solid*) global model

Fig. 4.19 **Global *vs.* local perspective on FEDGLOSS**. CIFAR100 $\alpha = 0$ with SAM as local optimizer @ 20*k* rounds on CNN. *(a) - (e):* Local models trained on one `class`, tested on the local ("*Local loss*") or global dataset ("*Global loss*"). Corresponding global perspective of local model trained with FEDSAM (*net*) added as reference. *(f):* FEDGLOSS (*net*) *vs.* FEDSAM (*solid*) resulting global model @ $t = 20k$, computed using the reported clients' models. **FEDGLOSS achieves aligned low-loss flat regions, effectively reducing the discrepancy between local and global sharpness.**

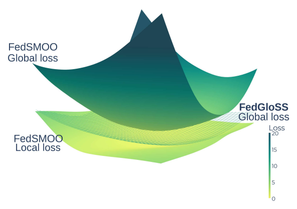
loss surfaces are found in *flat and low-loss regions*. This suggests that the proposed method effectively promotes convergence towards **aligned low-loss flat regions**, minimizing the discrepancy between local and global sharpness. This results in a global model residing in flatter minima in the global landscape (Figure 4.19f). Table 4.13 reinforces the effectiveness of FEDGLOSS. By combining ADMM for global consistency and server-side SAM for global flatness, FEDGLOSS prioritizes achieving a flatter *global* minimum rather than a local one during training, as evidenced by the lowest global maximum eigenvalue $\lambda_{1,g}$, but larger $\lambda_{1,l}$, across all clients and methods. This strategy successfully steers the optimization process towards a flatter global landscape, outperforming FEDSAM and FEDSMOO.



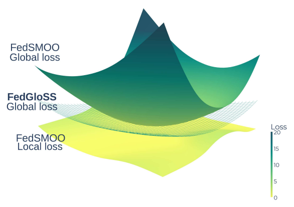
(a) Local model trained on class `sea` with FEDSMOO



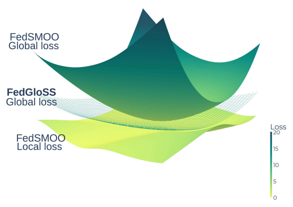
(b) Local model trained on class `snail` with FEDSMOO



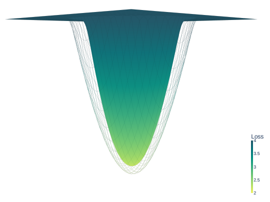
(c) Local model trained on class `bear` with FEDSMOO



(d) Local model trained on class `skyscraper` with FEDSMOO



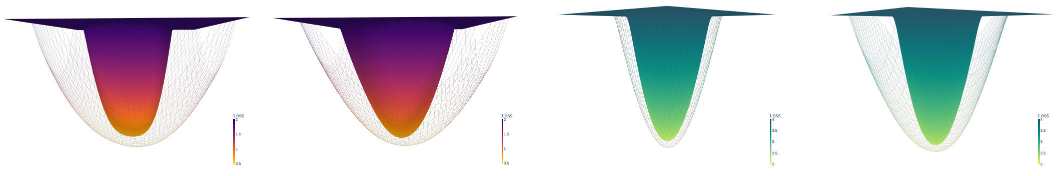
(e) Local model trained on class `possum` with FEDSMOO



(f) FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*) global model

Fig. 4.20 **Global *vs.* local perspective on FEDSMOO.** CIFAR100 $\alpha = 0$ with SAM as local optimizer @ 20k rounds on CNN. *(a) - (e):* Local models trained on one `class`, tested on the local ("*Local loss*") or global dataset ("*Global loss*"). Corresponding global perspective of local model trained with FEDGLOSS (*net*) added as reference. *(f):* FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*) resulting global model @ $t = 20k$, computed using the reported clients' models. **Local models trained with FEDGLOSS are found in lower and flatter regions in the global loss landscape w.r.t. FEDSMOO.**

Comparison with FEDSMOO. For completeness, the same analysis is repeated on FEDGLOSS *vs.* the best-performing SOTA method FEDSMOO in Figure 4.20, including the position in the global landscape of local models trained with FEDGLOSS for reference. While FEDSMOO achieves improved consistency between local and global sharpness compared to FEDSAM, it falls short of FEDGLOSS in reaching a flatter global minimum. As the figure shows, FEDGLOSS local models converge to flatter regions of the global landscape, demonstrably surpassing the performance of FEDSMOO. These conclusions are confirmed by the values of the maximum Hessian eigenvalues of the local models, computed on local and global datasets (Table 4.13): FEDSMOO achieves flatter *local* minima but *sharper* global regions w.r.t. FEDGLOSS.



(a) CIFAR10 $\alpha = 0$ (b) CIFAR10 $\alpha = 0.05$ (c) CIFAR100 $\alpha = 0$ (d) CIFAR100 $\alpha = 0.5$

Fig. 4.21 Visualization of the loss landscapes of the CNN trained with **FEDGLOSS** (*net*) and the best-performing SOTA **FEDSMOO** (*solid*). Comparison with varying degrees of heterogeneity on CIFAR10 (*left*) and CIFAR100 (*right*). FEDGLOSS consistently achieves flatter minima and lower loss values.

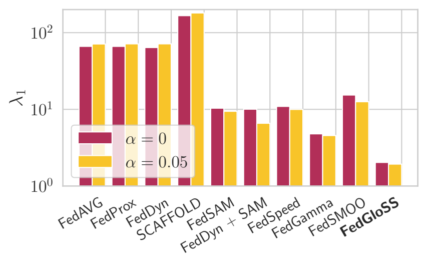
Table 4.15 **FEDGLOSS against the state of the art** on CIFAR datasets with varying degrees of heterogeneity. Results distinguished by local optimizer, SGD (top) and SAM (bottom). Comparison in terms of minimum communication cost, higher accuracy (%) and lower maximum Hessian eigenvalue. Best results in **bold**. Model: CNN.

	Method	Comm. Cost	CIFAR10				CIFAR100			
			$\alpha = 0$		$\alpha = 0.05$		$\alpha = 0$		$\alpha = 0.5$	
			Accuracy	λ_1	*Accuracy*	λ_1	*Accuracy*	λ_1	*Accuracy*	λ_1
Client SGD	FEDAVG	1×	59.9±0.4	66.23±0.50	65.7±1.0	71.14±4.07	28.6±0.7	66.30±3.08	38.5±0.5	68.77±0.96
	FEDPROX	1×	59.8±0.5	66.19±0.52	65.6±1.0	71.41±4.40	28.8±0.7	66.34±3.75	38.7±0.4	**68.63**±1.37
	FEDDYN	1×	65.5±0.3	63.94±4.41	70.1±1.2	71.44±8.73	✗	-	✗	-
	SCAFFOLD	2×	25.1±3.7	166.54±6.93	54.0±2.6	180.51±30.08	✗	-	30.0±1.1	120.01±0.76
	FEDGLOSS (ours)	1×	**69.5**±0.4	**58.26**±3.49	**75.5**±0.3	**56.28**±4.19	**42.5**±0.6	96.01±9.00	**47.9**±0.5	107.35±7.5
Client SAM	FEDSAM	1×	70.2±0.9	10.35±0.07	71.5±1.08	9.43±0.28	28.7±0.5	58.38±2.93	39.6±0.5	57.54±1.21
	FEDDYN	1×	79.3±3.1	10.04±5.38	81.5±0.6	6.58±0.20	✗	-	✗	-
	FEDSPEED	1×	70.9±0.4	10.92±0.17	72.3±1.1	9.97±0.12	28.9±0.5	58.23±3.18	39.7±0.5	58.00±1.86
	FEDGAMMA	2×	58.9±1.8	4.79±0.20	61.9±1.8	4.55±0.20	✗	-	29.4±1.4	99.86±6.74
	FEDSMOO	2×	81.3±0.5	15.37±1.67	82.8±0.6	12.57±0.56	47.8±0.5	28.43±1.97	51.7±0.46	29.23±0.17
	FEDGLOSS (ours)	1×	**83.9**±0.4	**2.03**±0.05	**84.4**±0.5	**1.93**±0.03	**50.6**±0.6	**17.18**±0.97	**53.4**±0.5	**16.22**±0.35

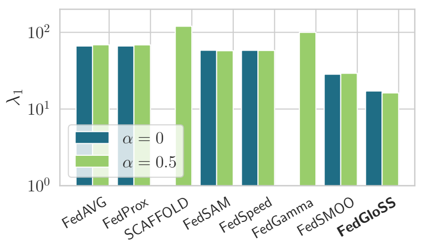
FedGloSS against the State of the Art

This section compares FEDGLOSS with the state-of-the-art methods described in Section 4.4.5 on CIFAR10 and CIFAR100 with varying degrees of heterogeneity. The methods are compared in terms of communication efficiency, accuracy, and global flatness (measured by the lowest Hessian eigenvalue, λ_1). As noted in Section 4.4.4, FEDGLOSS is compatible with any local optimizer. To clearly illustrate performance differences based on the choice of local optimizer, results with FEDGLOSS are reported using both SGD and SAM, distinguishing the comparison by optimizer type.

CNN. The main results obtained with the CNN model are presented in Table 4.15. Several observations highlight the advantages of FEDGLOSS. It is straightforward to notice how FEDGLOSS achieves the **best results** among both SGD and SAM-based approaches while maintaining **communication efficiency**. FEDGLOSS with



(a) CIFAR10



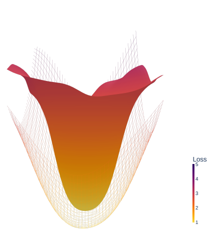
(b) CIFAR100

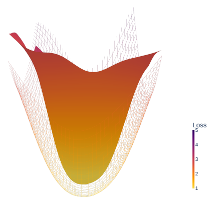
Fig. 4.22 Maximum Hessian eigenvalues in CIFAR10 (*left*) and CIFAR100 (*right*), with varying data heterogeneity (α), CNN. Value shown only if the algorithm converged.

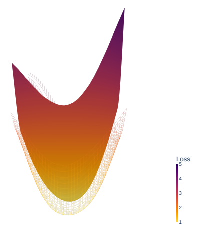
local SAM consistently outperforms the best-performing SOTA FEDSMOO by $\approx$ 2.5 percentage points in accuracy across all dataset configurations *with half the communication cost*. FEDGLOSS also reaches the **flattest global minima** (*e.g.*, $\lambda_1^{\text{FEDGLOSS}} = 2.03$ *vs.* $\lambda_1^{\text{FEDSMOO}} = 15.37$ on CIFAR10 with $\alpha = 0$), as shown in Figure 4.21, achieving the **best overall performance**. FEDGLOSS with local SGD overcomes by ≈ 4 percentage points *all* SGD-based approaches. As studied in [8], FEDDYN suffers from parameter explosion in highly heterogeneous settings, failing to converge on CIFAR100. Differently, FEDGLOSS successfully employs ADMM to align global and local solutions, reaching the best results under extreme heterogeneity. Finally, studies showed SCAFFOLD exhibits limited performance in complex heterogeneous environments [281, 7], resulting in its inability to converge on CIFAR alongside FEDGAMMA.

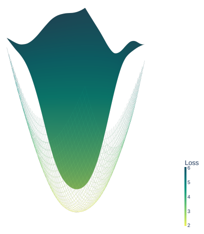
Lastly, Figure 4.22 compares the value of the maximum Hessian eigenvalues λ_1 of the considered algorithms. Results are distinguished by dataset (CIFAR10 on the left, CIFAR100 on the right), data heterogeneity ($\alpha \in \{0, 0.05\}$ in CIFAR10 and $\alpha \in \{0, 0.5\}$ in CIFAR100) and local optimizer (SGD or SAM). First, as expected, SAM-based methods achieve flatter minima w.r.t. the counterpart. Notably, the main competitor FEDSMOO presents higher sharpness than FEDSAM in the simpler CIFAR10, regardless of the data distribution. In addition, FEDGLOSS with local SAM achieves the lowest sharpness (*i.e.*, lowest λ_1) on *all* configurations, outperforming the state of the art and, specifically, *all* sharpness-aware methods.

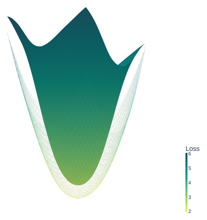
ResNet18. Table 4.16 and Figure 4.23 further confirm FEDGLOSS' effectiveness, consistently outperforming SOTA methods with the more complex ResNet18 architecture, with ≈ 8 points higher accuracy w.r.t. FEDAVG with both SGD and SAM,


(a) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDAVG (*solid*)


(b) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDSAM (*solid*)


(c) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*)


(d) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDAVG (*solid*)


(e) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDSAM (*solid*)

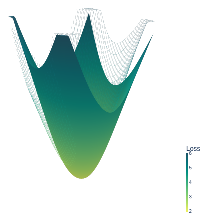

(f) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*)

Fig. 4.23 Loss landscapes with **ResNet18** on CIFAR10 $\alpha = 0.05$ (*top*) and CIFAR100 $\alpha = 0.5$ (*bottom*). FEDGLOSS achieves the flattest and lowest-loss regions in the global landscape.

and +5 w.r.t. FEDSMOO, with the flattest solutions. The accuracy trends can be found in Figures 4.24 to 4.26.

Homogeneous settings. For completeness, Table 4.17 evaluates FEDGLOSS against the main methods FEDAVG, FEDSAM, FEDDYN and FEDSMOO in *homogeneous* FL settings. Here, client gradients are naturally more aligned due to reduced client drift [280]. Thus, FEDGLOSS is tested with and without ADMM for global consistency. As expected, it achieves similar accuracy with or without ADMM, particularly when using SAM as the local optimizer. However, ADMMs facilitate convergence to flatter minima (evidenced by lower λ_1 values) by aligning local and global convergence points. Notably, FEDGLOSS achieves the flattest minima (lowest λ_1) across both datasets, and the best accuracy on the more complex CIFAR100. While FEDSMOO achieves slightly higher accuracy on CIFAR10, FEDGLOSS finds a flatter minimum and achieves competitive accuracy with significantly lower communication costs (halved).

Table 4.16 **FedGloSS against the state of the art with ResNet18** on Cifar100 $\alpha = 0.5$ and Cifar10 $\alpha = 0.05$. Comparison in accuracy (%).

	Method	Comm. cost	C100 **Accuracy**	C10 **Accuracy**
Client Sgd	FedAvg	1×	37.4±0.2	72.6±0.1
	FedProx	1×	37.6±0.1	72.2±0.2
	FedDyn	1×	38.8±0.6	70.2±0.6
	Scaffold	2×	38.6±0.1	70.8±0.6
	FedGloSS (ours)	1×	**46.7**±0.6	**79.1**±0.5
Client Sam	FedSam	1×	38.5±0.1	72.8±0.1
	FedDyn	1×	39.6±0.8	72.6±0.2
	FedSpeed	1×	38.7±0.6	72.6±0.1
	FedGamma	2×	38.8±0.3	72.2±0.1
	FedSmoo	2×	44.8±0.5	75.3±0.6
	FedGloSS (ours)	1×	**47.2**±0.2	**80.0**±0.3

Table 4.17 **FedGloSS against SOTA FL methods on homogeneous settings** with Cifar datasets, compared in terms of communication costs, accuracy (%) and maximum Hessian eigenvalue λ_1. Model: CNN. Best result in **bold** and second best underlined.

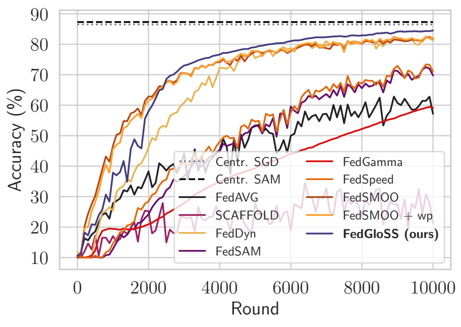
Method	Comm. Cost	ADMM	**Cifar10** $\alpha = 100$		**Cifar100** $\alpha = 1000$	
			Accuracy	λ_1	Accuracy	λ_1
FedAvg	1×	✗	84.0	68.4	50.1	49.4
FedSam	1×	✗	84.7	36.2	53.4	32.6
FedDyn (Sgd)	1×	✓	83.8	47.8	51.9	91.7
FedDyn (Sam)	1×	✓	84.5	27.9	52.5	46.0
FedSmoo	2×	✓	**85.1**	<u>6.4</u>	53.9	24.6
FedGloSS (Sgd)	1×	✗	84.0	67.7	50.5	50.8
	1×	✓	83.1	7.1	51.7	47.9
FedGloSS (Sam)	1×	✗	<u>84.8</u>	36.2	**55.8**	<u>13.9</u>
	1×	✓	<u>84.8</u>	**2.8**	<u>55.7</u>	**11.8**

FedGloSS on Real-World Large-Scale Vision Datasets

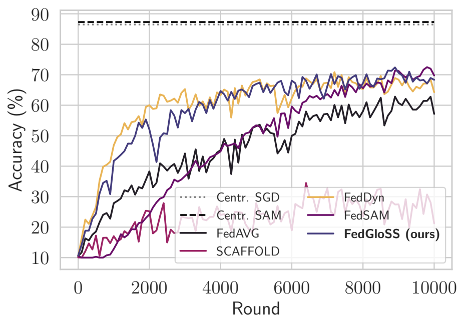
To further validate FedGloSS's effectiveness, its performance is evaluated on a large-scale image classification task using the challenging Landmarks-User-160k dataset, which provides a closer representation of real-world scenarios. Table 4.18 compares FedGloSS with local Sam against the best-performing baselines. Similar to the Cifar100 experiments (Table 4.15), both Scaffold and FedGamma fail to converge. FedGloSS is among the few methods, alongside FedSam and FedSmoo, that outperform FedAvg. Importantly, FedGloSS once again achieves the best overall performance (3.4% improvement over FedAvg) while maintaining reduced communication overhead.

ADMM and SAM Interaction in FedGloSS

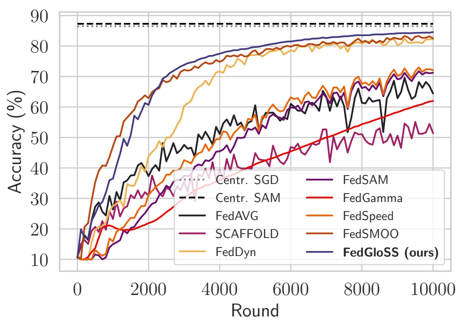
ADMM-based methods are often prone to parameter explosion in highly heterogeneous FL settings with many clients [8]. This occurs as multiple gradients accumulate



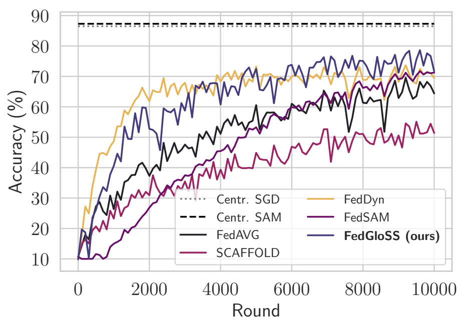
(a) $\alpha = 0$ SAM



(b) $\alpha = 0$ SGD



(c) $\alpha = 0.05$ SAM



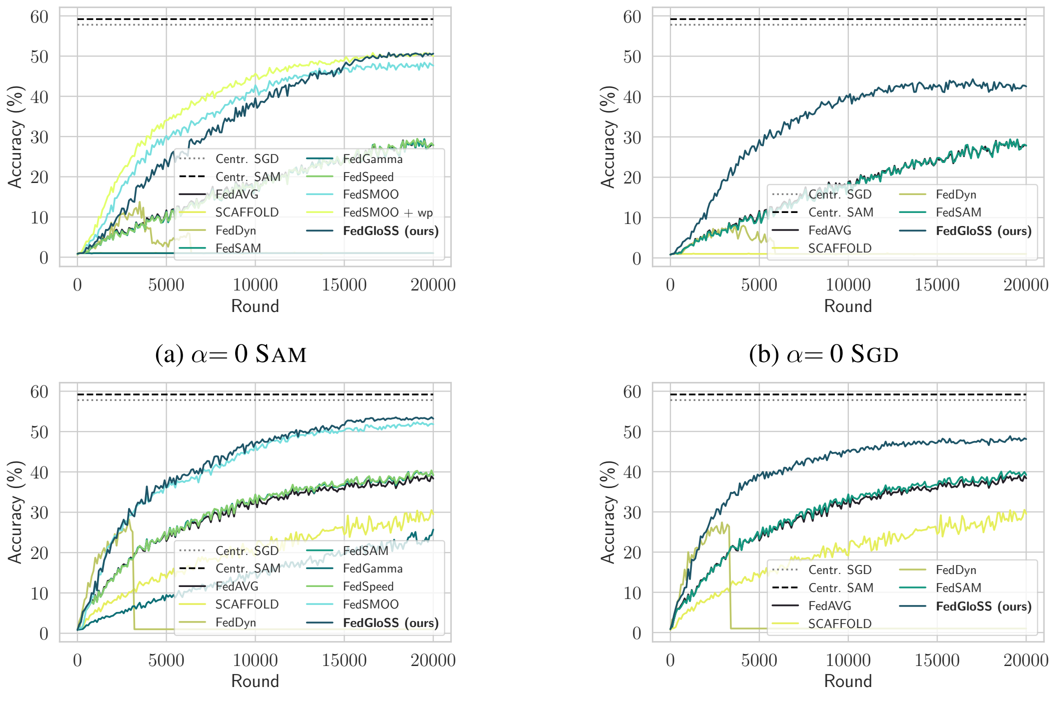
(d) $\alpha = 0.05$ SGD

Fig. 4.24 **Comparison of FEDGLOSS with state-of-the-art approaches. Accuracy trends with CNN on CIFAR10** with varying degrees of heterogeneity ($\alpha \in \{0, 0.05\}$). Methods distinguished by local optimizer, SAM (*a* and *c*) or SGD (*b* and *d*). Results of centralized runs (*dashed lines*) added as reference. **FEDGLOSS consistently achieves the best performance.**

Table 4.18 FEDGLOSS with local SAM with MobileNetv2, LANDMARKS-USER-160K

Method	**Comm. cost**	**Accuracy** (%)
FEDAVG	1×	56.3$\pm$0.2
FEDPROX	1×	55.0$\pm$0.2
FEDDYN	1×	55.2$\pm$0.6
SCAFFOLD	2×	✗
FEDSAM	1×	56.7$\pm$0.1
FEDDYN (w/ SAM)	1×	56.0$\pm$1.3
FEDGAMMA	2×	✗
FEDSMOO	2×	59.5$\pm$0.1
FEDGLOSS (ours)	1×	**59.7**$\pm$1.2

in the global dual variable σ (Section 4.4.4), causing the parameter norms to grow uncontrollably. However, empirical results indicate that this issue is mitigated with SAM (*e.g.*, see FEDDYN vs. FEDGLOSS in Table 4.15). We hypothesize this is due to SAM 's nature: by targeting flat minima, it promotes smaller gradient steps and minimal weight updates, resulting in a more stable algorithm. Figure 4.27 confirms our hypothesis by showing SAM's stability effectively lowers parameter norms and



(a) $\alpha = 0$ SAM (b) $\alpha = 0$ SGD

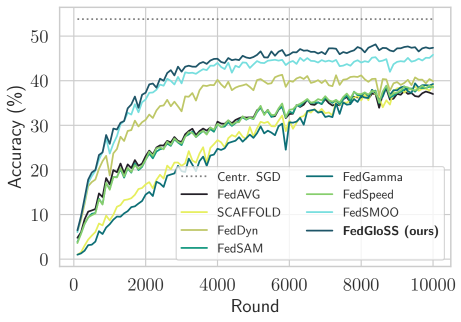
(c) $\alpha = 0.5$ SAM (d) $\alpha = 0.5$ SGD

Fig. 4.25 **Comparison of FEDGLOSS with state-of-the-art approaches. Accuracy trends with CNN on CIFAR100** with varying degrees of heterogeneity ($\alpha \in \{0, 0.5\}$). Methods distinguished by local optimizer, SAM (*a* and *c*) and SGD (*b* and *d*). Centralized upper bound (*dashed lines*) added as reference. **FEDGLOSS consistently achieves the best performance.**

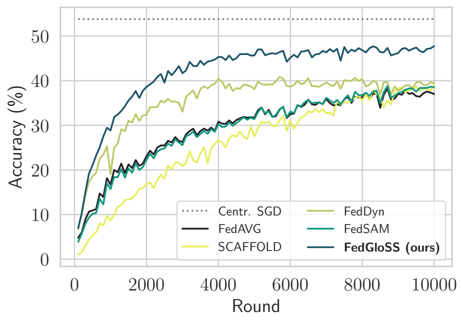
the consequent risk of explosion, particularly when SAM is applied directly to the global model, as in FEDGLOSS.

Communication Efficiency with FedGloSS

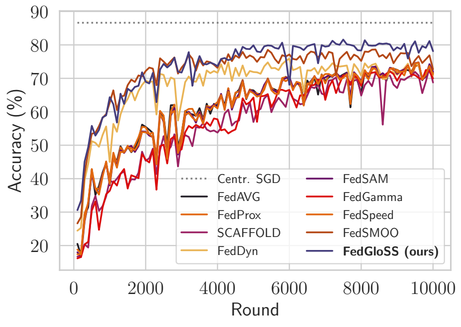
Communication cost is the main bottleneck in FL [277], making its optimization a relevant challenge. As previously highlighted, FEDGLOSS considers communication efficiency its primacy concern. Defined B the number of bits exchanged by FEDAVG in T training rounds, Table 4.19 studies FEDGLOSS' communication cost against the SOTA baselines in terms of rounds necessary to reach FEDAVG's performance and quantity of exchanged bits. The ADMM-based methods are usually faster, with FEDGLOSS being the fastest with ResNet18 and MobileNetv2. While FEDSMOO is faster when using the CNN model, the transmitted bits double due to its increased



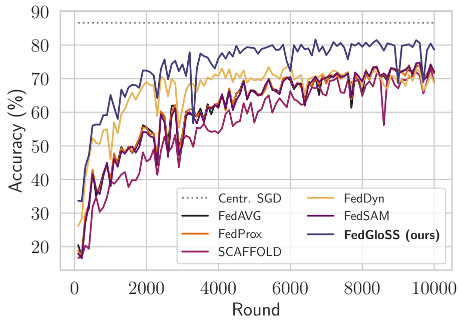
(a) CIFAR100 $\alpha = 0.5$
SAM



(b) CIFAR100 $\alpha = 0.5$
SGD



(c) CIFAR10 $\alpha = 0.05$
SAM



(d) CIFAR10 $\alpha = 0.05$
SGD

Fig. 4.26 **Comparison of FEDGLOSS with state-of-the-art approaches. Accuracy trends with ResNet18** on CIFAR100 (*top*) and CIFAR10 (*bottom*). Methods distinguished by local optimizer, SAM (*a* and *c*) and SGD (*b* and *d*). **FEDGLOSS consistently achieves the best performance, both in terms of final accuracy and convergence speed.**

communication cost, making **FEDGLOSS the most efficient method in all cases**. The reference performances are 59.9% for CIFAR10 and 28.6% for CIFAR100.

Ablation Studies

Communication-efficient Sharpness. Following the demonstration of global sharpness minimization's importance, this paragraph studies the efficacy of using the pseudo-gradient from the previous round $\tilde{\Delta}_{w}^{t-1}$ (Equation (4.10)) as an estimate of the sharpness measure. FEDGLOSS uses past gradients as a reliable indication on the global loss landscape and, by aligning global and local optimization paths through ADMM, enables consistent trajectories across rounds.

Table 4.20 compares FEDGLOSS with its baseline, NAIVEFEDGLOSS (described in Section 4.4.4), which computes the true perturbation $\hat{\varepsilon}^{t}$ at the expense of

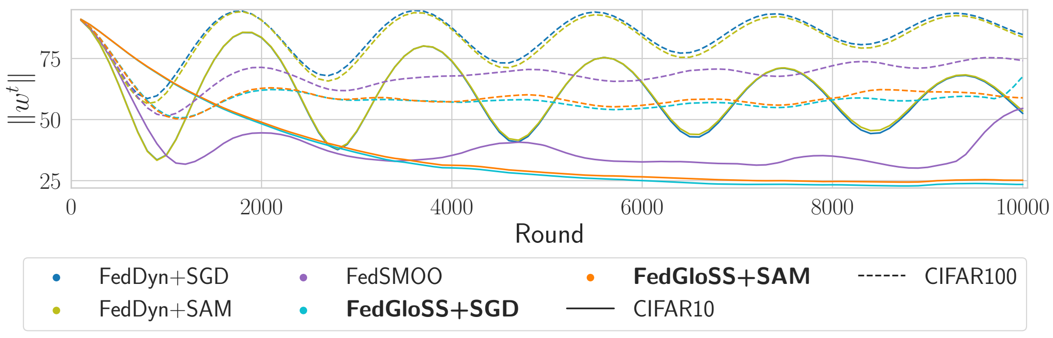


Fig. 4.27 **Trend of model parameters norm**, $\| \boldsymbol{w}^t \|_2$, on SAM-based methods with ResNet18 on CIFAR datasets. **SAM reduces the norm and the risk of parameters explosion, successfully enabling ADMM in heterogeneous FL**.

Table 4.19 Communication costs comparison in terms of number of rounds and transmitted bits w.r.t. FEDAVG (highlighted). “-” for not reached accuracy, “✗” for non-convergence.

	Method	**CNN**								**ResNet18**				**MobileNetv2**	
		CIFAR10				**CIFAR100**				**CIFAR10**		**CIFAR100**		**LANDMARKS-USER-160K**	
		$\alpha = 0$		$\alpha = 0.05$		$\alpha = 0$		$\alpha = 0.5$		$\alpha = 0.05$		$\alpha = 0.5$			
		Rounds	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*
Client SGD	FEDAVG	10*k* (1×)	1*B*	10*k* (1×)	1*B*	20*k* (1×)	1*B*	20*k* (1×)	1*B*	10*k* (1×)	1*B*	10*k* (1×)	1*B*	1.3*k* (1×)	1*B*
	FEDPROX	7.6*k* (1.3×)	0.8*B*	7.9*k* (1.3×)	0.8*B*	18.7*k* (1.1×)	0.9*B*	18.6*k* (1.1×)	0.9*B*	8.8*k* (1.1×)	0.9*B*	8.3*k*(1.2×)	0.8*B*	-	-
	FEDDYN	2*k* (5×)	**0.2B**	1.9*k* (5×)	**0.2B**	✗		✗		-	-	3.5*k* (2.9×)	0.4*B*	-	-
	SCAFFOLD	-	-	-	-	✗		-	-	-	-	8.9*k* (1.1×)	1.8*B*	✗	
	FEDGLOSS (ours)	3.4*k* (2.9×)	0.3*B*	3.8*k* (2.6×)	0.4*B*	5*k* (4×)	**0.3B**	4.7*k* (4.3×)	**0.2B**	2.4*k* (4.2×)	**0.2B**	1.9*k* (5.3×)	**0.2B**	-	-
Client SAM	FEDSAM	6.3*k* (1.6×)	0.6*B*	7.8*k* (1.3×)	0.8*B*	18.3*k* (1.1×)	0.9*B*	16.3*k* (1.2×)	0.8*B*	9.2*k* (1.1×)	0.9*B*	7.8*k* (1.3×)	0.8*B*	1.3*k* (1×)	1*B*
	FEDDYN	3*k* (3.3×)	0.3*B*	4.2*k* (2.4×)	0.4*B*	✗		✗		4.1*k* (2.4×)	0.4*B*	3.5*k* (2.9×)	0.4*B*	-	-
	FEDSPEED	6.3*k* (1.6×)	1.3*B*	6.9*k* (1.4×)	1.4*B*	18.3*k* (1.1×)	1.8*B*	15.7*k* (1.3×)	1.6B	8.3*k* (1.2×)	0.7*B*	8.3*k* (1.2×)	1.7*B*	1.3*k* (1×)	2*B*
	FEDGAMMA	-	-	-	-	✗		-	-	9.3*k* (1.1×)	1.9*B*	8.1*k* (1.2×)	1.6*B*	✗	
	FEDSMOO	1.9*k* (5.3×)	0.4*B*	2.2*k* (4.5×)	0.4*B*	4.5*k* (4.4×)	0.5*B*	6.5*k* (3.1×)	0.7*B*	2.4*k* (4.2×)	0.5*B*	2.3*k* (4.3×)	0.5*B*	200 (6.5×)	0.3*B*
	FEDGLOSS (ours)	2.2*k* (4.5×)	**0.2B**	2.2*k* (4.5×)	**0.2B**	6.3*k* (3.2×)	**0.3B**	5.2*k* (3.8×)	**0.3B**	2.4*k* (4.2×)	**0.2B**	1.9*k* (5.3×)	**0.2B**	200 (6.5×)	**0.2B**

doubled communication costs with a subset of selected clients. The results show that FEDGLOSS achieves accuracy comparable to NAIVEFEDGLOSS while maintaining communication efficiency. The performance difference is minimal or negligible. This aligns with the observed sharpness of the achieved minima, as measured by the maximum Hessian eigenvalue, λ_1. To ensure a fair comparison in communication cost, Table 4.20 also compares FEDGLOSS’ final accuracy with NAIVEFEDGLOSS’ performance at 50% training progress, showing that FEDGLOSS achieves higher accuracy with the same number of exchanges, benefiting from the additional global optimization steps deriving from its communication-efficient strategy. This shows the proposed approximation does not slow convergence, as also confirmed by the accuracy trends in Figure 4.28. In particular, this plot reports the accuracy trends of the two methods, showing that NAIVEFEDGLOSS is slightly faster ($\approx 1.1\times$) than FEDGLOSS in CIFAR100, while FEDGLOSS surpasses the speed of the baseline af-

Table 4.20 FEDGLOSS *vs*. its naïve implementation NAIVEFEDGLOSS in terms of communication cost, accuracy (50% and 100% of training), with change in performance in brackets, and maximum Hessian eigenvalue λ_1. SAM as CLIENTOPT.

Method	**Comm. Cost**	**CIFAR10** $\alpha = 0$			**CIFAR100** $\alpha = 0$		
		Acc@50% (↑)	Acc@100% (↑)	λ_1 (↓)	Acc@50% (↑)	Acc@100% (↑)	λ_1 (↓)
NAIVEFEDGLOSS	2×	$77.6_{\pm 0.3}$	$83.9_{\pm 0.2}$	$2.78_{\pm 0.13}$	$42.6_{\pm 0.8}$	$50.8_{\pm 0.1}$	$16.93_{\pm 0.27}$
FEDGLOSS	1×	$78.9_{\pm 0.5}$ (+1.3)	$83.9_{\pm 0.4}$	$2.03_{\pm 0.05}$ (−0.75)	$39.5_{\pm 0.9}$ (−3.1)	$50.6_{\pm 0.6}$ (−0.2)	$17.18_{\pm 0.97}$ (+0.25)
		CIFAR10 $\alpha = 0.05$			**CIFAR100** $\alpha = 0.5$		
		Acc@50% (↑)	Acc@100% (↑)	λ_1 (↓)	Acc@50% (↑)	Acc@100% (↑)	λ_1 (↓)
NAIVEFEDGLOSS	2×	$78.7_{\pm 0.1}$	$84.4_{\pm 0.2}$	$2.75_{\pm 0.09}$	$49.4_{\pm 0.5}$	$53.7_{\pm 0.3}$	$15.84_{\pm 0.52}$
FEDGLOSS	1×	$79.7_{\pm 0.4}$ (+1.2)	$84.4_{\pm 0.5}$	$1.93_{\pm 0.03}$ (−0.82)	$47.2_{\pm 1.1}$ (−2.2)	$53.4_{\pm 0.5}$ (−0.3)	$16.22_{\pm 0.35}$ (+0.38)

ter ≈ 25% of training in CIFAR10. However, both methods reach the same accuracy at the of training.

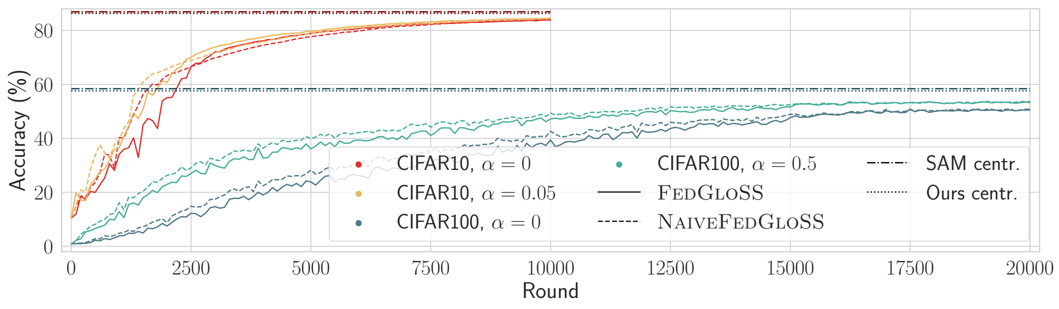


Fig. 4.28 **Accuracy trends of FEDGLOSS *vs*. NAIVEFEDGLOSS.** The comparison includes the centralized upper bounds of SAM and the adaptation of FEDGLOSS' strategy to the centralized scenario. CNN on CIFAR10 and CIFAR100 with varying heterogeneity degree (α). NAIVEFEDGLOSS is ≈ 1.1× faster than its efficient alternative FEDGLOSS in CIFAR100, while FEDGLOSS shows increased convergence speed after ≈ 25% of training rounds in CIFAR10. However, both methods reach the same accuracy at the of training. These results motivate the choice of FEDGLOSS over NAIVEFEDGLOSS.

Notably, FEDGLOSS' strategy in centralized settings lowers the performance w.r.t. SAM, thus reducing our centralized upper bound w.r.t. NAIVEFEDGLOSS. At equal performance, FEDGLOSS narrows the gap to the upper bound: −2.4% on CIFAR10 with $\alpha = 0$ and −1.9% with $\alpha = 0.05$ *vs*. respectively −3.2% and −2.7% of NAIVEFEDGLOSS w.r.t. SAM. In CIFAR100 instead, −7% on $\alpha = 0$ and −4.2% on $\alpha = 0.5$ of FEDGLOSS *vs*. −8.1% and −5.2% of NAIVEFEDGLOSS.

Lastly, Figure 4.29 shows that the use of the sharpness approximation does not steer the optimization path: the models trained with FEDGLOSS and NAIVEFEDGLOSS end up in the same basin (no loss barrier), with similar flatness. With the goal of reducing the impact of the communication bottleneck while achieving superior results, those results confirm our choice of FEDGLOSS over NAIVEFEDGLOSS.

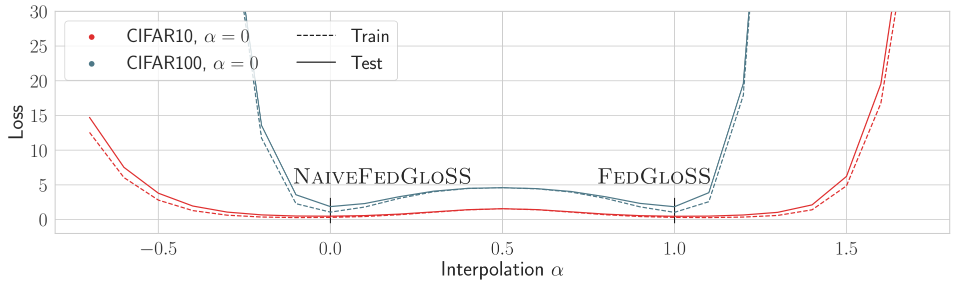
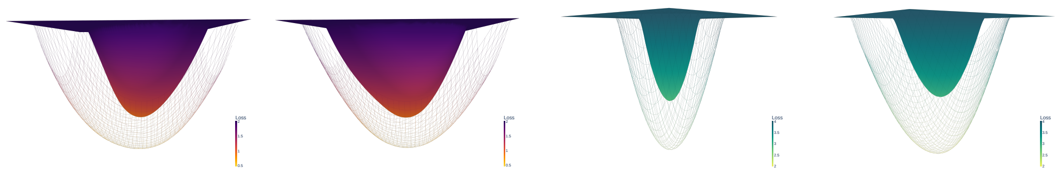


Fig. 4.29 Loss barriers resulting from interpolating NAIVEFEDGLOSS and FEDGLOSS' models, which are found in the same basin. CIFAR100 and CIFAR10 with CNN.

The Role of Global Consistency and Flatness. Table 4.21 isolates the impact of global consistency and global sharpness minimization in FEDGLOSS under extreme heterogeneity on the CIFAR datasets. As previously discussed, using the ADMM for alignment of local and global convergence points is equivalent to FEDDYN (Section 4.4.4), and FEDAVG with client-side SAM is FEDSAM. The table shows both components significantly impact performance, with FEDGLOSS achieving the best overall results. As expected, SGD-based approaches reach sharper minima than the SAM-based counterpart, while the proposed method achieves the lowest Hessian eigenvalue λ_1 on both datasets, demonstrating the efficacy of using SAM on the server side. Furthermore, unlike FEDDYN, FEDGLOSS is not prone to parameter explosion (marked with ✗), achieving the best results even where FEDDYN fails to converge [8]. The effectiveness of global sharpness minimization in FEDGLOSS is further highlighted by comparing its loss landscapes with FEDSAM in Figure 4.30.

Table 4.21 Efficacy of global sharpness minimization in FEDGLOSS. Analysis of the effect of ADMM for global consistency and server-side SAM for minimizing the global sharpness. Comparison in terms of accuracy (%) and maximum Hessian eigenvalue, distinguished by local optimizer. Model: CNN.

CLIENT OPT	**Method**	**Global Consistency**	**Global Sharpness**	**CIFAR10 α= 0**		**CIFAR100 α= 0**	
				Accuracy	λ_1	**Accuracy**	λ_1
SAM	FEDSAM	✗	✗	70.2$_{\pm 0.9}$	10.35$_{\pm 0.07}$	28.7$_{\pm 0.5}$	58.38$_{\pm 2.93}$
	FEDDYN	✓	✗	79.3$_{\pm 3.1}$	10.04$_{\pm 5.38}$	✗	-
	FEDGLOSS	✓	✓	83.9$_{\pm 0.4}$	2.03$_{\pm 0.05}$	50.6$_{\pm 0.6}$	17.18$_{\pm 0.97}$
SGD	FEDAVG	✗	✗	59.9$_{\pm 0.4}$	66.23$_{\pm 0.50}$	28.6$_{\pm 0.7}$	66.30$_{\pm 3.08}$
	FEDDYN	✓	✗	65.5$_{\pm 0.3}$	63.94$_{\pm 4.41}$	✗	-
	FEDGLOSS	✓	✓	69.5$_{\pm 0.4}$	58.26$_{\pm 3.49}$	42.5$_{\pm 0.6}$	96.01$_{\pm 9.00}$



(a) CIFAR10 $\alpha = 0$ (b) CIFAR10 $\alpha = 0.05$ (c) CIFAR100 $\alpha = 0$ (d) CIFAR100 $\alpha = 0.5$

Fig. 4.30 Visualization of the loss landscapes of the CNN trained with **FEDGLOSS** (*net*) and **FEDSAM** (*solid*). Comparison with varying degrees of heterogeneity on CIFAR10 (*left*) and CIFAR100 (*right*). These plots validate the necessity for global sharpness.

4.4.6 Limitations

A key limitation of FEDGLOSS for real-world deployments is its reliance on *stateful* clients for utilizing ADMMs. This necessitates each client to store information about its most recently trained model. Beyond memory constraints, this requirement might not be feasible in all cross-device FL settings due to the potential for a large number of clients and the low probability of any single client being selected multiple times.

4.4.7 Discussion

This work tackled the challenge of limited generalization in heterogeneous Federated Learning (FL), prioritizing communication efficiency for real-world use. Building on research linking poor generalization to sharp minima in the loss landscape, the proposed findings showed data heterogeneity worsens discrepancies between local and global loss surfaces, a problem not resolved by methods focusing only on local sharpness. To address this, a novel method, namely FEDGLOSS, finds flat minima in the *global* loss landscape with server-side Sharpness-Aware Minimization and achieves communication efficiency by approximating SAM's sharpness through past global pseudo-gradients, distinguishing it from prior approaches. This work revealed SAM prevents ADMM-related parameter explosion by guiding optimization along flat directions, enabling stable updates in heterogeneous FL.

Extensive evaluations showed FEDGLOSS outperforms SOTA methods in accuracy, flatness and communication efficiency, making it a strong candidate for real-world FL applications.

4.5 Window-based Model Averaging Improves Generalization in Heterogeneous Federated Learning

4.5.1 Motivation

Stochastic Weight Averaging (SWA) [237] has been shown to improve model robustness and reduce noise during training in heterogeneous FL by applying it to the server-side aggregation. However, as discussed in Section 4.3.6, this approach has a key limitation: it can only be effectively utilized in the later training stages (typically after 75% of the training rounds are complete), limiting its overall impact. The limitations of early SWA initialization are highlighted in Figure 4.31, where the performance of FEDAVG+SWA is compared across different starting rounds. The results demonstrate that SWA applied too early in the training process can lead to saturation and result in worse performance compared to FEDAVG alone. This emphasizes the importance of careful tuning for the starting round of SWA.

To overcome this limitation, this work introduces WIMA (**Wi**ndow-based **M**odel **A**veraging) to aggregate global models from different rounds using a window-based approach, effectively capturing knowledge from multiple users and reducing the bias from the last ones. By adopting a windowed view on the rounds, WIMA can be applied from the initial stages of training. Importantly, this method does not introduce any additional communication or client-side computation overhead.

4.5.2 Window-based Model Averaging (WiMA)

To overcome the instability and bias proper of training in heterogeneous cross-device federated scenarios, this work introduces ***Wi**ndow-based **M**odel **A**veraging* (WIMA). Defined a window size of W rounds, at the end of round $t \geq W$, WIMA averages the

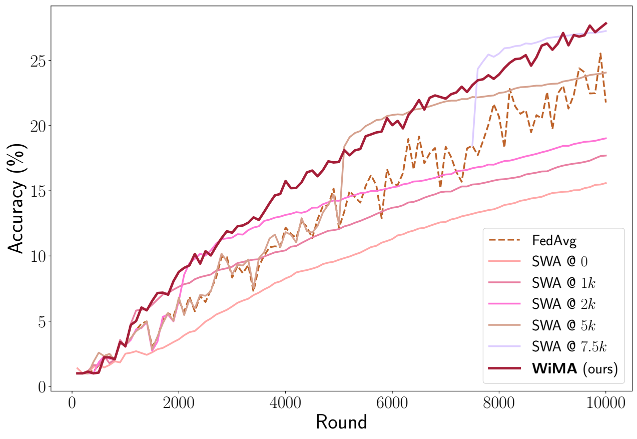


Fig. 4.31 Accuracy trends of WIMA and SWA starting at different rounds on CIFAR100 with $\alpha = 0$, using FEDAVG as reference. SWA suffers from early initialization.

last W global models built using FEDAVG as:

$$\boldsymbol{w}^{t+1}_{\text{WIMA}} = \boldsymbol{w}^{t'+W}_{\text{WIMA}} := \frac{1}{W} \sum_{\tau=t'}^{t'+W-1} \boldsymbol{w}^{\tau+1}_{\text{FEDAVG}}, \tag{4.12}$$

where $t' = t - W + 1$ is the first round comprised in the window frame. The rationale behind this approach is to enhance robustness of the global model towards distribution shifts across rounds and diminish bias towards the last-seen clients by averaging models that are still experiencing significant changes. By considering the last W rounds, sufficient history is retained to stabilize the model without hindering the training process, in contrast to what observed with SWA.

Unveiling the Window Contents

To understand the information stored inside the rolling window, Equation (4.12) can be reformulated using FEDOPT (Equation (3.4)) as follows:

$$\begin{aligned}
\boldsymbol{w}^{t'+W}_{\text{WIMA}} &= \frac{1}{W} \sum_{\tau=t'}^{t'+W-1} \boldsymbol{w}^{\tau+1}_{\text{FEDAVG}} && \text{(Equation (4.12))} \\
&= \frac{1}{W} \sum_{\tau=t'}^{t'+W-1} \sum_{k \in \mathscr{C}^\tau} \frac{N_k}{N} \boldsymbol{w}^\tau_k && \text{(FedAvg in Eq. 3.4)} \\
&= \frac{1}{W} \sum_{\tau=t'}^{t'+W-1} \left(\boldsymbol{w}^\tau - \eta_g \sum_{k \in \mathscr{C}^\tau} \frac{N_k}{N} (\boldsymbol{w}^\tau - \boldsymbol{w}^\tau_k) \right). && (4.13)
\end{aligned}$$

For simplicity, it is first assumed all clients have access to the same number of images, *i.e.* $\frac{N_k}{N} = \frac{1}{K^t} \forall k \in \mathscr{C}^t$. Since the same number of clients is selected at each round, $\frac{1}{|\mathscr{C}^t|} = \frac{1}{|\mathscr{C}|^{t-1}} = K^t \, \forall t \in [T]$. By unraveling the summation over the last W rounds and recursively writing each update $\boldsymbol{w}^\tau$ in Equation (4.13) using Equation (3.4), WIMA model's update becomes

$$\begin{aligned}
\boldsymbol{w}^{t'+W}_{\text{WIMA}} &= \frac{1}{W}\sum_{\tau=t'}^{t'+W-1}\Big(\boldsymbol{w}^\tau - \frac{1}{K^\tau}\sum_{i\in\mathscr{C}^\tau}(\boldsymbol{w}^\tau - \boldsymbol{w}^\tau_i)\Big)\\
&= \frac{1}{W}\sum_{\tau=t'}^{t'+W-1}\Big(\boldsymbol{w}^\tau - \frac{1}{K^\tau}\sum_{i\in\mathscr{C}^\tau}\underbrace{(\boldsymbol{w}^{\tau-1} - \frac{1}{K^{\tau-1}}\sum_{j\in\mathscr{C}^{\tau-1}}(\boldsymbol{w}^{\tau-1}-\boldsymbol{w}^{\tau-1}_j)}_{\boldsymbol{w}^\tau} - \boldsymbol{w}^\tau_i)\Big)\\
&\overset{K^{\tau-1}=K^\tau}{=} \frac{1}{W}\sum_{\tau=t'}^{t'+W-1}\Big(\boldsymbol{w}^\tau - \frac{1}{K^\tau}\sum_{i\in\mathscr{C}^\tau}(\boldsymbol{w}^{\tau-1}+\\
&-\frac{1}{K^\tau}\sum_{j\in\mathscr{C}^{\tau-1}}\underbrace{(\boldsymbol{w}^{\tau-2}-\frac{1}{K^\tau}\sum_{l\in\mathscr{C}^{\tau-2}}(\boldsymbol{w}^{\tau-2}-\boldsymbol{w}^{\tau-2}_l)}_{\boldsymbol{w}^{\tau-1}} - \boldsymbol{w}^{\tau-1}_j) - \boldsymbol{w}^\tau_i)\Big) = \dots\\
&= \frac{1}{W}\sum_{\tau=t'}^{t'+W-1}\Big(\boldsymbol{w}^\tau - \frac{1}{K^\tau}\sum_{i\in\mathscr{C}^\tau}(\boldsymbol{w}^{\tau-1} - \dots - \frac{1}{K^\tau}\sum_{m\in\mathscr{C}^1}(\boldsymbol{w}^0 - \frac{1}{|K^\tau|}\sum_{l\in\mathscr{C}^0}(\boldsymbol{w}^0+\\
&-\boldsymbol{w}^0_l) - \boldsymbol{w}^1_m) - \dots - \boldsymbol{w}^{\tau-1}_j) - \boldsymbol{w}^\tau_i)\Big)
\end{aligned}$$

As in standard SGD, each model implicitly contains information on the previous updates. By unraveling the summation over τ,

$$\begin{aligned}
\boldsymbol{w}^{t'+W}_{\text{WIMA}} &= \boldsymbol{w}^0 - \frac{1}{|K^0|}\Big(\underbrace{\sum_{i\in\mathscr{C}^0}(\boldsymbol{w}^0 - \boldsymbol{w}^0_i) + \dots + \sum_{i\in\mathscr{C}^{t'}}(\boldsymbol{w}^{t'} - \boldsymbol{w}^{t'}_i)}_{\color{red}{\tau\le t'}} +\\
&\underbrace{+ \underbrace{\frac{W-1}{W}}_{t'+W-(t'+1)=W-1}\sum_{i\in\mathscr{C}^{t'+1}}(\boldsymbol{w}^{t'+1} - \boldsymbol{w}^{t'+1}_i) + \dots + \frac{1}{W}\sum_{i\in\mathscr{C}^{t'+W-1}}(\boldsymbol{w}^{t'+W-1} - \boldsymbol{w}^{t'+W-1}_i)\Big)}_{\color{red}{t'<\tau<t'+W}}\\
&= \boldsymbol{w}^{t'} - \frac{1}{|K^0|}\Big(\frac{W-1}{W}\sum_{i\in\mathscr{C}^{t'+1}}(\boldsymbol{w}^{t'+1} - \boldsymbol{w}^{t'+1}_i) + \dots\\
&+ \frac{1}{W}\sum_{i\in\mathscr{C}^{t'+W-1}}(\boldsymbol{w}^{t'+W-1} - \boldsymbol{w}^{t'+W-1}_i)\Big)
\end{aligned}$$

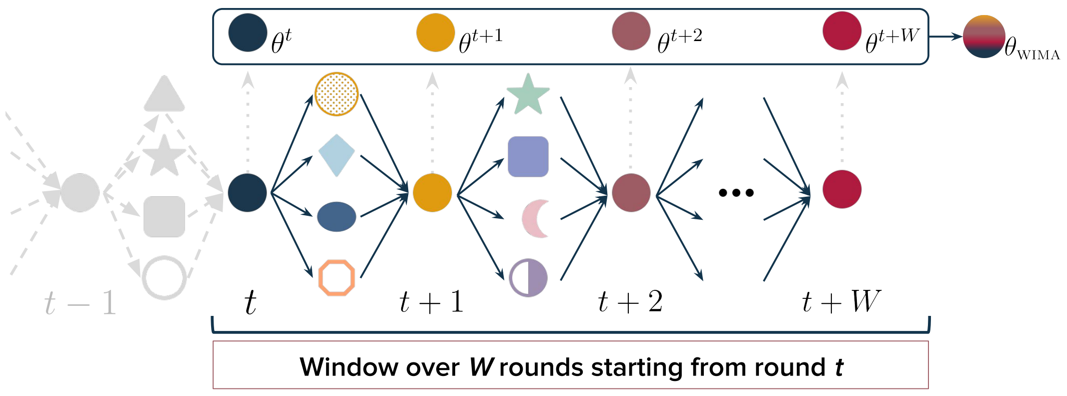


Fig. 4.32 Overview of WIMA. A window of size W slides across the sequence of global models generated by server-side aggregation, from round t to round $t+W$. The WIMA parameters are calculated as the average of the models within this window.

$$= \boldsymbol{w}^{t'} - \frac{1}{|K^0|} \sum_{\tau=t'+1}^{t'+W-1} \frac{t'+W-\tau}{W} \sum_{i\in\mathscr{C}^\tau} (\boldsymbol{w}^\tau - \boldsymbol{w}_i^\tau).$$

By dropping the constraint $\frac{N_k}{N} = \frac{1}{K^t}\,\forall k \in \mathscr{C}^t$ and inserting the server learning rate η_g, the results is summarized as

$$\boldsymbol{w}^{t'} - \eta_g \sum_{\tau=t'}^{t'+W-1} \frac{t'+W-\tau}{W} \sum_{i\in\mathscr{C}^\tau} \frac{N_i}{N} (\boldsymbol{w}^\tau - \boldsymbol{w}_i^\tau), \tag{4.14}$$

or more in general,

$$\boldsymbol{w}^{t'} - \sum_{\tau=t'}^{t'+W-1} \frac{t'+W-\tau}{W} \textsc{ServerOpt}(\boldsymbol{w}^\tau, \Delta_{\boldsymbol{w}}^\tau, \eta_g, \tau). \tag{4.15}$$

The term ${}^{t'+W-\tau}/_{W}$ tends to 1 when $\tau = t'$, *i.e.*, at the beginning of the queue, and to ${}^1/_W$ when $\tau = t'+W-1$, *i.e.*, in the last round. Thus, Equation (4.15) can be interpreted as $W-1$ SGD steps starting from the initial model $\boldsymbol{w}^{t'}$ with a ***learning rate decay*** that depends on the position in the queue, *i.e.*, ${}^{t'+W-\tau}/_{W}$. Indeed, WIMA assigns higher significance to previous updates, as they are perceived as more stable, while also integrating new knowledge at a rate proportional to the window size W. This sets it apart from methods like momentum, which prioritizes more recent updates.

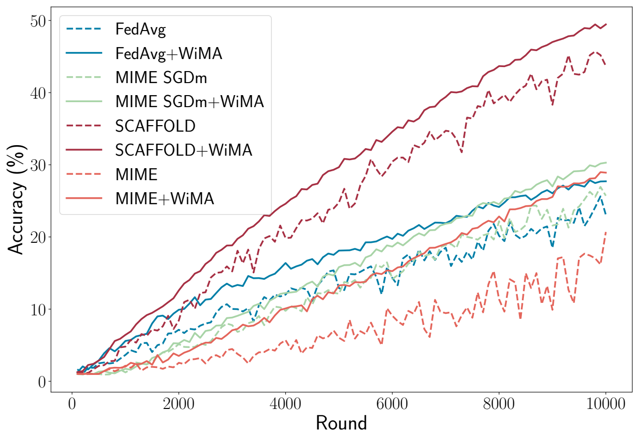


Fig. 4.33 Accuracy trends of different SOTA algorithms on CIFAR100 $\alpha = 0$ across rounds, with and without WIMA (dashed lines). The application of WIMA results in smoother and more stable trends, leading to enhanced robustness and improved performance.

4.5.3 Results in Real-World Vision Scenarios

This section introduces the experimental results of WIMA on the federated CIFAR10, CIFAR100 (both Dirichlet and PAM), FEMNIST and LANDMARKS-USER-160K. For the CIFAR datasets, the chosen architecture is ResNet20 with BN layers substitued with GN ones. Following [34], a 2-layer Convolutional Neural Network is instead trained on FEMNIST, and MobileNetv2 pre-trained on ImageNet on LANDMARKS-USER-160K, as in [278, 7]. Additional information on the datasets and the implementation details can be found in Section 3.4.1 and Chapter A, respectively.

Reducing noise and increasing stability with WiMA

Thanks to the window-based average of global models, WIMA mitigates the negative impact of statistical heterogeneity inherent in cross-device federated settings. As shown in Figure 4.33, WIMA effectively smooths the learning trends, resulting in enhanced robustness and reduced instability. Notably, these benefits are observed across all performance levels, with improvements evident in low-performing approaches (*e.g.*, MIME) as well as the best-performing ones (*e.g.*, SCAFFOLD).

Table 4.22 WIMA combined with state-of-the-art FL algorithms. For each configuration, the first column reports the accuracy (%) reached by each standalone method; in the second column, the performance achieved when adding WIMA. Between brackets the improvements introduced by WIMA, underlined the best ones in each dataset. For simplicity, the table only reports gains in improvements ≥ 1.5. Best overall accuracy in bold.

Algorithm	**CIFAR10**				**CIFAR100**						**FEMNIST**	
	$\alpha = 0$		$\alpha = 0.05$		$\alpha = 0$		$\alpha = 0.5$		PAM			
		w/ WIMA		w/ WIMA		w/ WIMA		w/ WIMA		w/ WIMA		w/ WIMA
FEDAVG	64.37	69.95 ($\uparrow$ 5.3)	68.50	72.69 ($\uparrow$ 4.2)	23.00	27.91 ($\uparrow$ 4.9)	31.21	34.45 ($\uparrow$ 3.2)	47.41	48.53	83.59	85.06 ($\uparrow$ 1.5)
FEDAVGM	73.32	75.72 ($\uparrow$ 2.4)	73.10	75.30 ($\uparrow$ 2.2)	24.27	28.77 ($\uparrow$ 4.5)	31.78	33.97 ($\uparrow$ 2.2)	55.96	61.63 ($\uparrow$ 5.7)	85.00	85.26
MIME SGD	74.92	80.65 ($\uparrow$ 5.7)	78.82	82.81 ($\uparrow$ 4.0)	17.55	29.05 ($\uparrow$ 11.5)	27.30	40.37 ($\uparrow$ 13.1)	54.33	57.44 ($\uparrow$ 3.1)	85.37	86.40
MIME SGDM	74.58	76.20 ($\uparrow$ 1.6)	78.39	80.38 ($\uparrow$ 2.0)	25.78	30.11 ($\uparrow$ 4.3)	38.42	43.08 ($\uparrow$ 4.7)	54.62	57.28 ($\uparrow$ 2.7)	86.67	87.40
MIMELITE	64.42	67.78 ($\uparrow$ 3.4)	68.27	71.21 ($\uparrow$ 2.9)	20.00	24.69 ($\uparrow$ 4.7)	35.56	39.15 ($\uparrow$ 3.6)	53.97	60.34 ($\uparrow$ 6.4)	**86.82**	**87.51**
FEDCM	78.83	81.73 ($\uparrow$ 2.9)	73.94	80.28 ($\uparrow$ 6.3)	19.62	25.29 ($\uparrow$ 5.7)	36.12	40.10 ($\uparrow$ 4.0)	53.16	54.12	83.88	84.90
FEDACG	55.27	60.09 ($\uparrow$ 4.8)	63.20	66.35 ($\uparrow$ 3.2)	20.09	23.55 ($\uparrow$ 3.5)	29.74	32.46 ($\uparrow$ 2.7)	58.88	61.38 ($\uparrow$ 2.5)	85.73	86.14
FEDPROX	64.25	69.90 ($\uparrow$ 6.7)	67.82	71.90 ($\uparrow$ 4.1)	22.59	27.58 ($\uparrow$ 5.0)	30.70	33.68 ($\uparrow$ 3.0)	55.91	62.25 ($\uparrow$ 6.3)	84.50	85.21
SCAFFOLD	**81.45**	**83.96** ($\uparrow$ 2.5)	**83.24**	**85.17** ($\uparrow$ 1.9)	**45.65**	**49.77** ($\uparrow$ 4.1)	**50.93**	**53.75** ($\uparrow$ 2.8)	56.09	57.64 ($\uparrow$ 1.6)	85.87	86.61
FEDDYN	N/A	N/A	N/A	N/A	5.88	8.48 ($\uparrow$ 2.6)	20.88	24.54 ($\uparrow$ 3.7)	**57.42**	**63.00** ($\uparrow$ 5.6)	N/A	N/A
ADABEST	66.05	73.95 ($\uparrow$ 7.9)	71.54	77.42 ($\uparrow$ 5.9)	24.92	31.41 ($\uparrow$ 6.5)	37.45	43.81 ($\uparrow$ 6.4)	54.98	57.57 ($\uparrow$ 2.6)	84.95	86.02

The Effectiveness of WIMA combined with SOTA

Table 4.22 presents the results achieved by using WIMA on top of state-of-the-art (SOTA) federated algorithms designed to handle statistical heterogeneity.

Looking at standalone algorithms (*i.e.*, w/o WIMA), SCAFFOLD achieves the best performances overall. FEDDYN is not able to converge in the most heterogeneous settings, as already shown in [8, 7]. Notably, WIMA enables each method to achieve better final accuracy, showcasing substantial improvements compared to the algorithm without WIMA. The most significant gains are observed on the more challenging CIFAR datasets. In particular, WIMA proves especially beneficial for the worst-performing methods, increasing the final accuracy by over 11 points for MIME SGD on both α values in CIFAR100. On the other hand, using the aggregation proposed by WIMA is effective even on the overall best-performing SCAFFOLD, or on the less challenging FEMNIST. Thus, all methods and settings are positively affected by the increased robustness and stability introduced by WIMA.

The same efficacy of WIMA can be appreciated in Table 4.23, using the large-scale LANDMARKS-USER-160K. Without redundancy and loss of generality, the performance of WIMA is presented when integrated with both the standard FedAvg and the best-performing SCAFFOLD. Even in this more complex vision scenario, WIMA achieves large gains in accuracy.

Table 4.23 Large-scale experiments. Results in test accuracy (%) on LANDMARKS-USER-160K. Best result in bold.

Algorithm	w/o WIMA	w/ WIMA
FEDAVG	58.17	63.05
SCAFFOLD	62.32	**68.30**

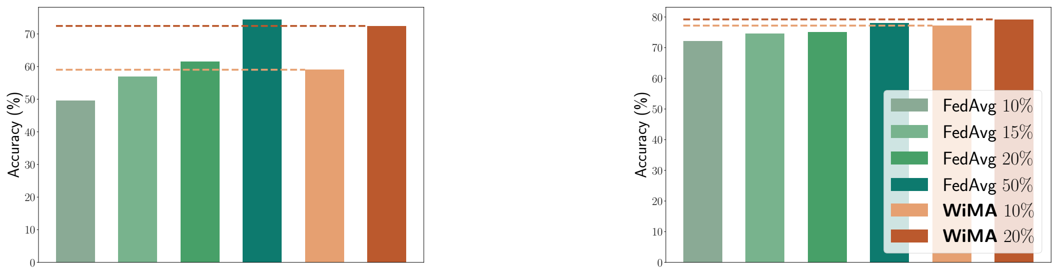
WiMA *vs.* SWA

Figure 4.31 compares the application of WIMA or SWA (with various starting rounds) on top of the baseline, FEDAVG. As already discussed in Section 4.5.1, SWA suffers from early initialization. Differently, thanks to the windowed view of the global models across rounds, WIMA can be used from the beginning of training, and presents a constantly stable and better trend than SWA.

WiMA Enables Reduced Client Participation

In Figure 4.34, the enhanced generalization capability achieved with WIMA allows narrowing the gap with runs involving higher client participation rates. Specifically, it compares the results of FEDAVG with and without the proposed method, using varying numbers of clients selected at each round on CIFAR10. Experiments are run using batch size 20 to account for more local iterations, highlighting the client drift. WIMA enables the model with a 10% participation rate to attain a final accuracy that is *at least* comparable to the run involving 1.5 times the number of devices with α= 0 and twice that number with α= 0.05. When 20% of clients are involved instead, WIMA reaches performances comparable (α= 0) or better (α= 0.05) than FEDAVG involving half the devices (50% rate).

This result holds significant importance in cross-device settings, where devices are often unavailable due to factors such as limited battery life, network connectivity issues, and communication overload (Section 3.3). The ability to achieve improved results with fewer clients involved aligns favorably with real-life requirements, making it a valuable contribution.



(a) $\alpha = 0$ (b) $\alpha = 0.05$

Fig. 4.34 WIMA performances compared with varying client participation rates at each round on CIFAR10 using FEDAVG. **a)** WIMA achieves higher accuracy with 10% participation compared to FEDAVG with 1.5 times the number of devices per round. WIMA with 20 clients performs similarly to FEDAVG with half the clients. **b)** WIMA with 10% rate performs almost on par with FEDAVG w/o WIMA selecting 50% of the devices.

Table 4.24 WIMA accuracy (%) with varying W on the CIFAR datasets with $\alpha = 0$. Best results in bold.

Window size W	**WIMA Accuracy (%)**	
	CIFAR10	CIFAR100
5	67.25	22.71
10	69.12	25.70
50	69.79	22.75
100	69.94	27.91
200	68.74	27.72

Ablation Studies

Window size. The dimension W of the window used in WIMA plays a crucial role in achieving a trade-off between retaining useful historical information and avoiding excessively old data. Table 4.24 compares the accuracy reached with varying values of W on the heterogeneous CIFARS. Smaller W values lead to lower performance as the WIMA model fails to capture sufficient information from the underlying distribution, while excessively large W values slow down training by relying on outdated updates. The optimal results are obtained with $W = 100$ in both cases.

Better features quality with WiMA. Aiming to understand which part of the architecture WIMA affects the most, Table 4.25 reports its performances when acting only on the feature extractor, or the classifier, *i.e.*, the output layer of the model. To

Table 4.25 Accuracy (%) reached when applying WIMA only on the classifier, the feature extractor, or all the model parameters (*All*) as reference.

Dataset	α	WIMA *Classifier*	WIMA *Feature extractor*	WIMA *All*
CIFAR10	0	47.76	59.03	59.53
	0.05	71.01	76.72	78.87
CIFAR100	0	25.13	27.10	27.91
	0.5	36.12	36.29	37.88

allow for more client-side fine-tuning, the batch size is set to 20 rather than 100. The results demonstrate that WIMA is mainly acting on the feature extractor. Thanks to the more robust and less biased output features, the classifier is consequently able to give more accurate predictions.

4.5.4 Discussion

This work proposed Windowed Model Averaging (WIMA) to mitigate the negative effects of statistical heterogeneity in Federated Learning. WIMA aims to reduce noise and instability in the learning trends of models trained in non-*i.i.d.* FL settings.

To address these issues, WIMA averages the last W global models generated by any server-side optimizer at each training round. This windowed approach maintains sufficient historical information to stabilize the model without impeding the training process. WIMA is readily combined with most existing state-of-the-art FL algorithms, demonstrably improving their performance and leading to smoother, more stable training trends.

The analysis suggests that WIMA primarily influences the network's backbone, generating improved output features that consequently enhance the classifier's prediction accuracy. Finally, WIMA helps narrow the performance gap compared to settings with higher client participation rates, a desirable outcome for realistic FL scenarios.

4.6 Summary

This chapter introduced a new perspective on the use of the geometry of the loss surface to explain and mitigate the lack of generalization in heterogeneous federated learning, with a specific focus on vision tasks.

To summarize, the main contributions are as follows:

- Section 4.3 highlights the convergence to sharp minima of models trained in heterogeneous FL, linking it to poor generalization proper of heterogeneous federated settings.
- To address this issue, FEDSAM introduces Sharpness-aware Minimization (SAM) on the client-side training to locally learn better models. The intuition behind this approach is that the improved local models' generalization ability positively reflects on the global model. The work analyzes both SAM and its adaptive version ASAM as valuable solutions, showing how the latter better adapts to the FL scenario.
- In addition, Section 4.3 proposes to use SWA (Stochastic Weight Averaging) on the server-side to aggregate global models across rounds and improve their robustness to distribution shifts. The combination of server-side SWA and client-side SAM/ASAM leads to the overall best performance.
- In heterogeneous scenarios, local and global loss landscapes are usually inconsistent due to the difference in the optimization objectives, as shown in Section 4.4. This limits the effectiveness of approaches like FEDSAM that only account for local flatness. FEDGLOSS (Federated Global Server-side Sharpness) addresses this issue by moving SAM the to server-side optimization, thus optimizing for *global* flat minima.
- The use of SAM, especially on the server side, enables effective ADMM use in FL. While typically ADMM-based methods suffer from parameter explosion [8], the introduced findings indicate that by targeting flat minima, SAM encourages smaller gradient steps and minimal weight updates, leading to a significantly more stable algorithm.
- Lastly, while effective in stabilizing the learning trend and improving the model robustness, SWA can only be applied during the latest training stages,

limiting its potential impact. This issue is overcome by WIMA (Window-based Model Averaging), that can be used from the beginning of training thanks to a window-based aggregation of global models (Section 4.5). Notably, the use of WIMA is empirically equivalent to involving more clients for training at each round, and does not require any additional communication exchange or client-side computation. These advantages make WIMA an efficient and scalable solution for real-world deployments.

Chapter 5

Cluster-based Approaches for Improved Generalization and Convergence Speed in Heterogeneous Federated Learning

The way up to the top of the mountain is always longer than you think. Do not fool yourself, the moment will arrive when what seemed so near is still very far.

PAULO COELHO

This chapter explores leveraging client data distribution similarities and dissimilarities to enhance the convergence speed of models trained in heterogeneous federated learning settings. Section 5.2 introduces FEDSEQ, a sequential training orchestration method that groups clients with diverse data distributions. In Section 5.3 instead the user image style information is used to identify client similarities and assign them to clusters, with each cluster receiving a dedicated model. Finally, Section 5.4 describes a method employing a Graph Convolutional Neural Network to model client interactions and similarities, facilitating weighted knowledge exchange.

5.1 Introduction

In heterogeneous federated learning, most of the methods introduced in Section 3.3 struggle to reach the performance of the centralized upper bound. Thus, this chapter approaches heterogeneity from a different angle, focusing on the **training orchestration** rather than modifying the local training objectives, to learn more robust models before the server-side averaging step, resulting in reduced noise and achieving centralized performance.

In particular, the presented works leverage similarities in the clients' data distributions to build clusters. FEDSEQ (Section 5.2) groups together clients having dissimilar distributions to speed up the training convergence, while LADD (Section 5.3) uses the local style to assign cluster-specific models, and FEDCG (Section 5.4) clusters the clients' images into various domains and models the interactions among domains through a graph. Both LADD and FEDCG work on unsupervised information.

The results presented in this chapter led to the publication of the following papers:

- Caldarola, D., Mancini, M., Galasso, F., Ciccone, M., Rodolà, E., & Caputo, B. (2021).
  Cluster-driven Graph Federated Learning over Multiple Domains.
  In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop on Learning from Limited and Imperfect Data* (pp. 2749-2758) (**CVPRW 2021**).

- Zaccone*, R., Rizzardi*, A., Caldarola, D., Ciccone, M., & Caputo, B. (2022).
  Speeding Up Heterogeneous Federated Learning with Sequentially Trained Superclients.
  In *2022 26th International Conference on Pattern Recognition* (pp. 3376-3382). IEEE. (**ICPR 2022**)

- Shenaj*, D., Fanì*, E., Toldo, M., Caldarola, D., Tavera, A., Michieli, U., Ciccone, M., Zanuttigh, P., & Caputo, B. (2023).
  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* (pp. 444-454). (**WACV 2023**)

- Silvi*, A., Rizzardi*, A., Caldarola*, D., Caputo, B., & Ciccone, M. (2024). Accelerating Federated Learning via Sequential Training of Grouped Heterogeneous Clients.
  IEEE Access.

5.2 Accelerating Federated Learning via Sequential Training of Grouped Heterogeneous Clients

Federated Learning facilitates training machine learning models in privacy-preserving settings by enabling collaboration among edge devices without local data sharing. However, this approach presents challenges due to the heterogeneity of local datasets' statistical distributions and clients' computational capabilities. Notably, the presence of highly non-*i.i.d.* data significantly hinders both the trained neural network's performance and its convergence rate, requiring more communication rounds to achieve centralized performance.

This work proposes FEDSEQ (Federated Learning via Sequential Superclients Training), a novel framework that leverages the sequential training of heterogeneous client subgroups (termed *superclients*) to learn more robust models before server-side averaging and speed up convergence, without breaking any privacy constraints. The code is available at https://github.com/RickZack/FedSeq.

5.2.1 Introduction

FEDSEQ (*Federated Learning via Sequential Superclients Training*) is a novel approach addressing statistical heterogeneity in FL, based on *sequential* training among clients, carried out in parallel across distinct client groups to harness the distributed setting's parallelism. By allowing the model to access a larger portion of data before the averaging step, the negative effects of data heterogeneity are mitigated, speeding up the training and moving closer to the desired minimum. By grouping clients having diverse local distributions together in a *superclient*, we simulate the existence of a larger, homogeneous dataset while maintaining data

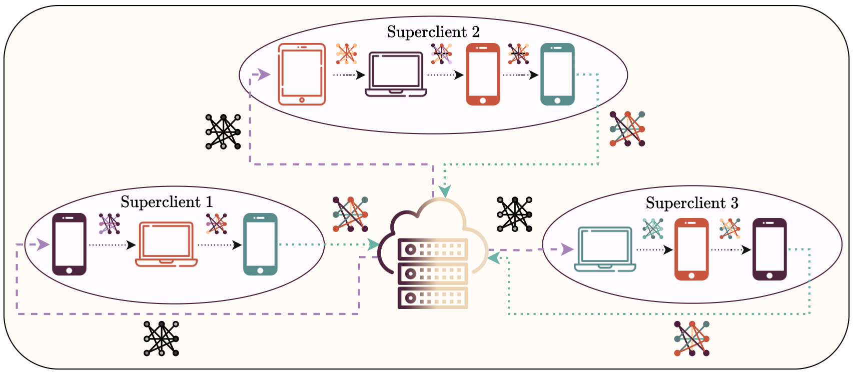


Fig. 5.1 To mitigate statistical heterogeneity in FL, FEDSEQ forms *superclients* by grouping clients with distinct local data distributions (different colors), creating simulated larger and homogeneous datasets. Sequential training takes place within the selected superclients at each round. The current global model is received by the first client in the chain and sent back by the last one.

privacy, as illustrated in Figure 5.1. Clients within the same superclient form a chain and train the received model in a sequential manner. The final updates are sent from the last client to the server and merged there. Intuitively, **this scheme emulates the training dynamics observed on devices with more extensive and evenly distributed datasets**, resulting in a favorable setting for FL.

Communication is known to be the main bottleneck in federated training, *e.g.*, due to the clients' unavailability and unreliability Section 3.3. While sequential training provides robustness against data heterogeneity, it can potentially result in slower training progress. This occurs when slower clients end up in the same superclient, leading to increased waiting times on the server side. To overcome this limitation, **FEDASYNCSEQ** introduces **asynchronous client-server communication by implementing sequential training among superclients**. Rather than merging updates at the end of each training round, FEDASYNCSEQ allows the model updated by one superclient to be sent directly to another one, enabling faster groups of clients to complete multiple training iterations before merging their updates. At regular intervals of every R rounds, the updates received by the server, potentially stemming from varying numbers of training iterations, are aggregated. This approach not only reduces the number of aggregation and synchronization steps with the server but also allows the model to be trained on a larger number of clients before the averaging step. Consequently, this brings the model closer to a centralized scenario, as ideally, it encounters all superclients before being merged.

To further increase the parallelism as training moves on, **FedSeq2Par** dynamically updates the number of superclients and their corresponding client assignments as the rounds progress. It prioritizes sequentiality, *i.e.* larger groups, during the initial stages, and gradually transitions to parallelism, emphasizing smaller groups in the later stages. This allows easier adaptation to varying numbers of devices and their respective distributions.

Lastly, the robustness of the algorithm against threats posed by potentially malicious participants is a crucial aspect in the FL paradigm [416, 364, 365]. Malevolent clients may attempt to disrupt the training process by manipulating their input data [417, 418], or even try to infer private information of other clients by exploiting the received global model [252]. To assess whether the novel *client-to-client* sequential training approach introduces any privacy vulnerabilities, this work conducts tests against these attacks and observes that **FedSeq often exhibits higher privacy resistance compared to the widely-used FedAvg** [34], considered the de-facto standard algorithm for Federated Learning.

To summarize, the main contributions are the following:

- The introduction of FedSeq, a new FL algorithm that learns from groups of sequentially-trained clients (*superclients*).
- Several lightweight procedures to compare the clients' probability distributions, analyzing their impact on the creation of superclients.
- Three grouping strategies are evaluated and compared with the naïve random assignment, showing the impact of group quality on the algorithm convergence.
- To speed up training, FedAsyncSeq decreases the need for synchronization between superclients and server, and FedSeq2Par increases parallelism.
- The extensive empirical analyses and tests demonstrate that the developed approaches outperform the state of the art in terms of convergence performance and speed in both *i.i.d.* and non-*i.i.d.* scenarios.
- FedSeq is shown to resist to common attacks against clients' privacy, demonstrating its robustness.

5.2.2 Federated Learning via Sequential Superclients Training

Building upon the problem formulation introduced in Section 3.2, this section details the components of the proposed method, distinguishing between FEDSEQ, FEDASYNCSEQ and FEDSEQ2PAR.

Building Superclients

This section details how to create a superclient S from users with diverse local distributions while respecting the privacy constraints, *i.e.*, without accessing clients' data directly. Ihe overall goal is to group clients with diverse data distributions into *superclients* $\{S_i\}_{i=1}^{N_S}$, aiming to minimize the divergence in distribution among superclients. Sequential training is then performed by clients within the same group. Intuitively, this approach allows local models to accumulate knowledge from the overall data distribution, even when client datasets exhibit significant heterogeneity.

Assigning clients to equally-sized groups while minimizing distribution distance is a challenging problem similar to the bin packing problem [419], and is NP-hard in nature. Thus, this work proposes using multiple greedy strategies to estimate the local distributions in a privacy-preserving way and solve the clients' clustering problem, being flexible towards dynamic and constantly evolving FL environments.

To this end, different grouping criteria G_S are introduced, based on *i*) a *client distribution estimator* $\psi_{(.)}$, providing privacy-preserving statistics on the local data distribution, *ii*) a *metric* τ, for evaluating the distance between the estimated data distributions, and *iii*) a *grouping method* $\phi_{(.)}$, to assemble dissimilar clients, *i.e.* $G_S := \{\psi_{(.)};\tau;\phi_{(.)}\}$. The approach is depicted in Figure 5.2.

Privacy-preserving Estimation of Clients' Data Distributions. The model $f_{\boldsymbol{w}}$ can be defined as a combination of a deep feature extractor $h_{\boldsymbol{w}_{\text{feat}}} : \mathscr{X} \rightarrow \mathscr{Z}$ and a classifier $g_{\boldsymbol{w}_{\text{clf}}} : \mathscr{Z} \rightarrow \mathscr{Y}$, where $\boldsymbol{w} = \{\boldsymbol{w}_{\text{feat}}, \boldsymbol{w}_{\text{clf}}\}$ is the entire set of model parameters and $\mathscr{Z}$ the output feature space. The classification output is given by $g \circ h : \mathscr{X} \rightarrow \mathscr{Y}$, where the subscripts are dropped to ease the notation.

FEDSEQ exploits a *pre-training phase* to estimate the users' data distribution. Each client $k \in \mathscr{C}$ trains a common random model $\boldsymbol{w}_0$ on its dataset $\mathscr{D}_k$ for E epochs,

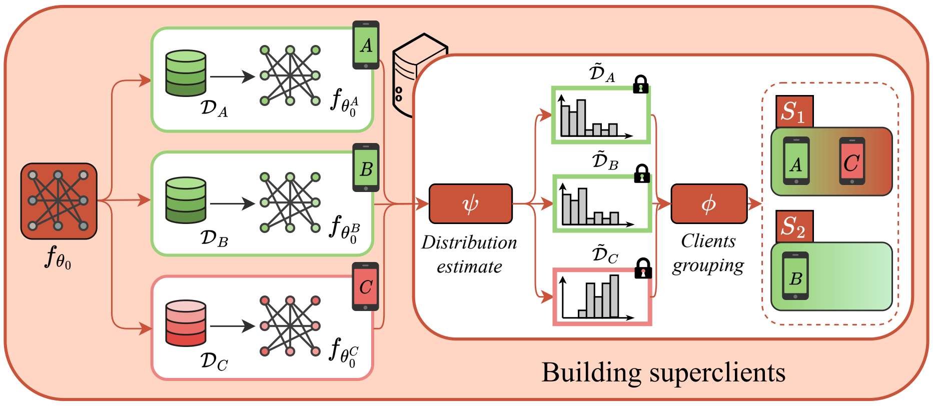


Fig. 5.2 FEDSEQ pre-training phase to build superclients. **a)** The initial random global model f_{w_0} is sent to all the clients, which train it using their local data $\mathscr{D}_k \, \forall k \in \mathscr{C}$. **b)** The local data distributions are estimated (ψ) using the clients' updates while preserving their privacy. **c)** Based on the grouping strategy ϕ, clients are assigned to N_S superclients.

resulting in $f_{\boldsymbol{w}_k^0}$, which serves as a starting point for the following distribution estimation approaches. The proposed client distribution estimators are

- ψ_{clf}: as the **model classifier** is biased towards the training data [201], its parameters $\boldsymbol{w}_{k,\text{clf}}^0$ serve as a proxy for the client's local data distribution.
- ψ_{conf} relies on the **predictions of the local models on a server-side public dataset** $\mathscr{D}_{pub}$, *i.e.*, $\{f_{\boldsymbol{w}_k^0}(z) =: f_k^0,\ z \in \mathscr{D}_{pub},\ k \in \mathscr{C}\}$. $\mathscr{D}_{pub}$ contains J samples for each class $c \in [N_C]$. The predictions are averaged by class as $p_{k,c} = \frac{1}{J}\sum_{x \in \mathscr{D}_c} f_k^0(x)$, where $\mathscr{D}_c \subset \mathscr{D}_{pub}$ is contains only samples of class c. The k-th client's *confidence vector* is defined as:

$$p_k := \text{softmax}(\{p_{k,1}, \dots, p_{k,N_C}\}) \in [0,1]^{N_C} \tag{5.1}$$

  Since the k-th model's predictions are favorable towards the majority of the classes seen in $\mathscr{D}_k$ [420], p_k is an acceptable privacy-preserving representation of $\mathscr{D}_k$. However, ψ_{conf} relies on the availability of a public dataset that accurately reflects the overall data distribution — a requirement that can be challenging to meet.
- ψ_{t2v}: to overcome the limitations of ψ_{conf}, ψ_{t2v} extracts **vectorial representations of given tasks** based on *Task2Vec*[421]. The representation is an

approximation of the Fisher Information Matrix (FIM), defined as

$$F := \mathbb{E}_{(x,y)\sim f_{\boldsymbol{w}}(x,y)}[\nabla_{\boldsymbol{w}} \log f_{\boldsymbol{w}}(y|x) \nabla_{\boldsymbol{w}} \log f_{\boldsymbol{w}}(y|x)^T]. \tag{5.2}$$

The FIM serves as a metric for the information content of a parameter regarding the joint distribution $f_{\boldsymbol{w}}(x,y)$. If it has limited influence on the classification performance for a specific task, its corresponding entry in the FIM will be low. Thus, the FIM represents the task itself, here corresponding to each client's local dataset. Starting from a pre-trained set of weights $\tilde{\boldsymbol{w}}^0$, the classifier is fine-tuned on $\mathscr{D}_k$ and the FIM is computed on the feature extractor parameters. The resulting representations are demonstrated to capture taxonomic and semantic similarities between tasks.

In the following sections, $\tilde{\mathscr{D}}_k$ will indicate the estimate provided by $\psi_{(.)}$ for the k-th device's data distribution.

Grouping Clients. $\mathscr{D}_S = \bigcup_{k\in\mathscr{C}_S} \mathscr{D}_k$ is defined as the union of the data from the clients $\mathscr{C}_S \subset \mathscr{C}$ belonging to a superclient S. The aim is to find the maximum amount of superclients N_S satisfying the following constraints: *i*) minimum number of samples $|\mathscr{D}_S|_{min}$, and *ii*) maximum number of clients $K_{S,max}$ per superclient. Given $\psi_{(.)}$ and τ, FEDSEQ approximates the solution of the problem using:

- ϕ_{rand}, a naïve yet practical method that **randomly** assigns clients to superclients until the stopping criterion is met.
- ϕ_{kmeans}: **K-means** [422] is first applied to obtain N_S homogeneous clusters. Each superclient is formed by iteratively extracting one client at a time from each cluster, until $|\mathscr{D}_S| \geq |\mathscr{D}_S|_{min}$ and $K_S \leq K_{S,max} \forall S$.
- ϕ_{greedy} initially assigns one random client $k_i \in \mathscr{C}$ to the superclient S. The next $k_j \in \mathscr{C} \setminus \{k_i\}$ is chosen so as **the distance between** k_i **and** k_j **is maximized**, *i.e.* $\max_{j\in[K]} \tau\ (\tilde{\mathscr{D}}_{k_i}, \tilde{\mathscr{D}}_{k_j})$. The process is repeated by iteratively maximizing $\tau\ (\tilde{\mathscr{D}}_j, \frac{1}{|S|}\sum_{i\in|S|} \tilde{\mathscr{D}}_i)$, with $|S|$ being the cardinality of S, until the defined constraints are met.
- While highly effective [423], the best-performing ϕ_{greedy} is hindered by its iterative nature, leading to slower execution. In response, the ***Inter-Cluster***

Grouping (ICG) algorithm from [322] is adapted to the proposed approach (ϕ_{ICG}). Differently from ICG that requires superclients of equal size, FEDSEQ relaxes this constraint by redistributing the unassigned clients.

The output of this procedure is a set $\mathscr{S}$ of N_S superclients, where each superclient S_i includes $K_{S_i} := |\mathscr{C}_{S_i}| \leq K_{S,max}$ clients and $|\mathscr{D}_{S_i}| \geq |\mathscr{D}_S|_{min}$ data points, with $\sum_{S_i \in \mathscr{S}} K_{S_i} = K$.

Sequential Training

This work proposes three alternatives to leverage *sequential training* within the created superclients, namely FEDSEQ, FEDASYNCSEQ and FEDSEQ2PAR. The approaches are summarized in Algorithm 6.

FedSeq. As shown in Figure 5.3, the clients $\{k_{i,1}, \ldots, k_{i,|S_i|}\} \in \mathscr{C}$ belonging to the superclient S_i $\forall i \in [N_S]$ form a chain performing sequential training. At each round t, the server selects a subset of $S_t \in \mathscr{S}$ superclients. Within each superclient S_i, the first device $k_{i,1}$ receives the global model $f_{\boldsymbol{w}_t}$ from the server and locally trains it for E_k epochs on $\mathscr{D}_{k_{i,1}}$. The updated parameters $\boldsymbol{w}_{k_{i,1}}^{t+1}$ are sent to the next client of the sequence $k_{i,2}$. Such training procedure continues until the last client $k_{|S_i|}$ updates the received model and sends it back to the server. The passage over all the clients of the chain can be repeated E_S times allowing a ring communication strategy. However, $E_S \neq 1$ leads to an increase in communication as multiple messages have to be exchanged between clients, which is why we discourage this approach and set $E_S = 1$ in our experiments. On the server-side, the superclients updates are averaged following Equation (3.5). Intuitively, the training of the model over multiple clients' data before the averaging step simulates the existence of a larger, homogeneous dataset.

FedAsyncSeq. In realistic scenarios, synchronous federated training can become impractical, especially when considering factors such as the latency of slower devices. The delays can be further exacerbated by FEDSEQ since there is no control over the capabilities of clients' systems, and multiple slow clients may be grouped together within the same superclient, leading to a significant increase in server-side waiting time. To mitigate the challenges posed by latency and ensure efficient training,

Algorithm 6 FedSeq, FedAsyncSeq and FedSeq2Par

```
Require: $f_{w^0}$, $G_S$, $K_{S,max}$, $|\mathscr{D}_S|_{min}$. Epochs $E$, $E_k$, $E_S$. $T$ rounds. Clients $\mathscr{C}$. Fraction
    $C$ of superclients selected at each round.
 1: Growth function and parameters $f_{gr}$, $\alpha_{gr}$ and $\beta_{gr}$.
 2: $\mathscr{S} \leftarrow$ CREATE_SUPERCLIENTS$(f_{w^0}, G_S, e, K_{S,max}, |\mathscr{D}_S|_{min}, K, f_{gr}(\alpha_{gr}, \beta_{gr}; t))$
 3: $N_S \leftarrow |\mathscr{S}|$
 4: $w \leftarrow [w^0, \dots, w^0]_{1\times CN_S}$, $p \leftarrow [0, \dots, 0]_{1\times CN_S}$
 5: for $t = 1$ to $T$ do
 6:     if $f_{gr}(t, \alpha_{gr}, \beta_{gr}) > |N_S|$ then
 7:         $\mathscr{S} \leftarrow$ CREATE_SUPERCLIENTS$(f_{w^0}, G_S, e, K, f_{gr}(\alpha_{gr}, \beta_{gr}; t))$
 8:         $N_S \leftarrow |\mathscr{S}|$
 9:     end if
10:     $\mathscr{S}^t \leftarrow$ Subsample fraction $C$ of $N_S$ superclients
11:     for $S_i \in \mathscr{S}^t$ in parallel do
12:         Shuffle clients in $S_i$
13:         $w^t_{S_i,0} \leftarrow w^t$                                  ▷ FedSeq and FedSeq2Par
14:         $w^t_{S_i,0} \leftarrow w[i]$
15:         for $e_S = 1$ to $E_S$ do
16:             $w^{t+1}_{S_i} \leftarrow$ SEQUENTIALTRAINING$(w^t_{S_i,0}, E_k)$
17:         end for
18:         $w[i] \leftarrow w^{t+1}_{S_i}$, $p_i \leftarrow p_i + |\mathscr{D}_{S_i}|$
19:     end for
20:     $w^{t+1} \leftarrow$ FEDAVG $(\{w^{t+1}_{S_i}, \forall S_i \in \mathscr{S}^t\})$       ▷ FedSeq and FedSeq2Par
21:     if $t \mod N_S = 0$ then
22:         $w^{t+1} \leftarrow \sum_i \frac{p_i}{p} w[i]$, $p = \sum_i p_i$
23:         $w \leftarrow [w_{t+1}, \dots, w_{t+1}]_{1\times CN_S}$, $p \leftarrow [0, \dots, 0]_{1\times CN_S}$
24:     end if
25: end for
```

FEDASYNCSEQ leverages sequentiality at the superclient level while allowing asynchronous updates to be merged (Figure 5.4). Instead of aggregating the models after *every* round, the server does so every R rounds. During this time frame, the model received by each superclient S_i is instantly sent to another random superclient S_j (where $i \neq j$). This approach allows the fastest chains to continue with additional training iterations instead of waiting for slower ones. After every R rounds, the most recent updates from each chain of superclients, which may originate from distinct rounds, are combined. This approach shares similarities with asynchronous settings, where models are aggregated as soon as they become available. It is worth noting that R can potentially equal T, meaning that the updated models are only

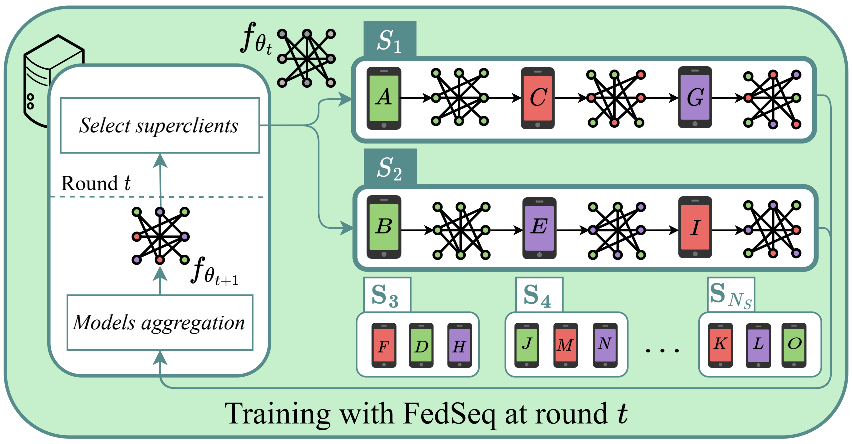


Fig. 5.3 Sequential training with FEDSEQ. At each round t, a subset of superclients (here S_1 and S_2) is selected and receives $\boldsymbol{w}_t$, which is trained sequentially by the clients. Final updates are sent back to the server, where they are aggregated with FEDAVG.

averaged at the end of training. This strategy reduces the number of aggregation and synchronization steps with the server while enabling the model to potentially observe the entire dataset before averaging. This brings the paradigm closer to the centralized setting while still leveraging FL's parallelism.

FedSeq2Par. Sequential training of the model through a long chain of clients with highly diverse data distributions may lead to catastrophic forgetting [401]. This refers to the model forgetting the knowledge acquired from the initial users while becoming overly specialized to the most recently seen datasets [322]. Additionally, maintaining a static sequence of clients within each superclient may result in the model learning information based on the order of the clients, introducing biases or unintended patterns. Building upon [322], this work introduces the *Sequential-to-Parallel* (STP) approach in FEDSEQ2PAR to overcome these issues. Sequential training is exploited during the initial stages, facilitating information exchange among heterogeneous clients. **Parallelism is gradually introduced by incrementally increasing the number of superclients** and reducing their sizes, promoting faster convergence. The dynamic creation of superclients allows accounting for new clients, while eliminating potential biases related to the static nature of superclients. Formally, in each training round t, STP dynamically constructs an increasing number of superclients $\{S_i\}_{i=1}^{f_{gr}(\alpha_{gr},\beta_{gr};t)}$, where $f_{gr}(\cdot,\cdot;\cdot)$ is a non-decreasing growth function dependent on the training round t, and the hyperparameters $\alpha_{gr}, \beta_{gr} \in \mathbb{R}^+$ control

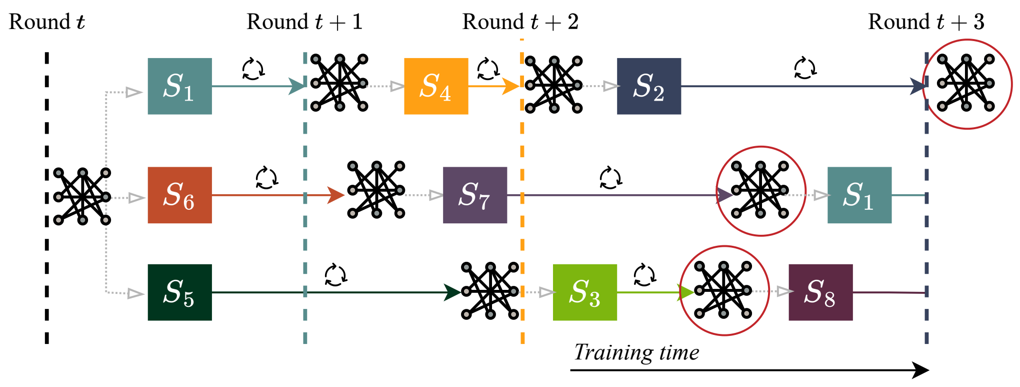


Fig. 5.4 **Training with FEDSEQ2PAR** with $R = 3$. At round t, the global model is sent to the selected superclients $\{S_1, S_5, S_6\}$, having varying latency (*full arrows*). The first superclient to complete training (S_1) marks the start of the new round $t + 1$. As soon as the server receives the updated model, it sends it to another superclient (*e.g.*, from S_1 to S_4). Within the time it takes for S_5 to finish training, the faster S_1 and S_4 can also complete theirs. After R rounds, the latest updates from each chain of superclients (*circled in red*) are combined, and the process begins anew.

the growth rate and the initial number of superclients respectively. The choice of f_{gr} is critical for the behavior of STP. As in [322], three possible growth functions are taken into account: *linear*, allowing for a smooth growth; *logarithmic*, promoting an initial faster dynamic; *exponential*, with slower changes at first,

$$f_{linear}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}[\alpha_{gr}\,(t-1)+1], \tag{5.3}$$

$$f_{log}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}[\alpha_{gr}\,\ln t + 1], \tag{5.4}$$

$$f_{exp}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}(1 + \alpha_{gr})^{t-1}. \tag{5.5}$$

5.2.3 Experimental Results

This section introduces the experimental results on the proposed methods, highlighting their improved convergence speed and generalization performance.

Datasets. FEDSEQ, FEDASYNCSEQ, and FEDSEQ2PARare evaluated on the federated CIFAR and FEMNIST image classification datasets. All datasets exhibit heterogeneous distributions concerning label skew. Differently from the CIFAR datasets, FEMNIST also presents notable feature shifts attributed to diverse calligraphy styles depicting the same letter or number, and the local dataset cardinality significantly varies across clients. In order to set up a heterogeneous scenario for CIFAR10 and CIFAR100, the local class distribution is sampled from a Dirichlet distribution with $\alpha \in \{0, 0.2, 0.5\}$ [290]. Both CIFAR datasets are divided into 500 clients with 100 images each. The *i.i.d.* and non-*i.i.d.* data distributions of FEMNIST introduced in [394] follow the writers' ownership, *i.e.*, each client is a distinct writer. The non-*i.i.d.* split accounts for both label skew and feature shift.

The methods are additionally tested on NLP (Natural Language Processing) tasks, using SHAKESPEARE [394] and STACKOVERFLOW [424] datasets for next character and next word predictions respectively. The *i.i.d.* and non-*i.i.d.* distributions of SHAKESPEARE reflect some Shakespearean characters, with 100 clients owning around $3,743$ samples each, while the implementation of STACKOVERFLOW follows [275], representing a realistic cross-device settings thanks to a larger number of clients, $40k$ in total. As done in [275], due to its prohibitively large number of clients and examples, testing on STACKOVERFLOW full dataset is performed only at the end and a subsample is used during training.

More in-depth details can be found in Chapter A.

Algorithms. To validate the effectiveness of the proposed approaches, this study conducts a comparison with state-of-the-art (SOTA) algorithms for heterogeneous FL. In addition to the standard FEDAVG, FEDSEQ and its variants are tested against FEDPROX, SCAFFOLD, FEDDYN, and FEDCYCLIC [425], which cyclically exchanges models across clients without relying on a central server. In particular, FEDCYCLIC can be seen as the extreme case of FEDSEQ with $N_S = 1$ and no server-side aggregation. FEDCYCLIC was chosen over FEDSTAR (from the same paper [425]) due to the latter's impractical communication overhead in realistic scenarios. FEDSTAR indeed requires each client to send its updates to *all* the other participants, leading to an exponential increase in communication costs. To ensure a fair comparison, FEDCYCLIC is not trained on all clients at each round but randomly selects a fraction C of the available ones. The same applies to all the other approaches and, most importantly, the number of model updates within rounds is the same for all the compared methods.

Table 5.1 Comparison with state-of-the-art FL algorithms. Color coding: first, second and third best results.

Algorithm	CIFAR10			CIFAR100			FEMNIST		SHAKESPEARE		STACKOVERFLOW
	$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	NIID	IID	NIID	IID	-
Centralized		85.64$_{\pm0.07}$			54.97$_{\pm0.19}$		87.52$_{\pm0.27}$		52.00$_{\pm0.02}$		28.50$_{\pm0.20}$
FEDAVG	71.27$_{\pm0.29}$	76.32$_{\pm0.36}$	77.39$_{\pm0.43}$	42.68$_{\pm0.22}$	48.79$_{\pm0.55}$	49.51$_{\pm0.61}$	81.55$_{\pm0.11}$	83.06$_{\pm0.12}$	48.68$_{\pm0.12}$	48.50$_{\pm0.07}$	24.68$_{\pm0.15}$
FEDPROX	71.52$_{\pm0.08}$	76.21$_{\pm0.50}$	77.38$_{\pm0.57}$	42.83$_{\pm0.18}$	48.84$_{\pm0.65}$	49.44$_{\pm0.49}$	81.55$_{\pm0.05}$	83.07$_{\pm0.05}$	48.73$_{\pm0.12}$	48.51$_{\pm0.15}$	24.59$_{\pm0.19}$
SCAFFOLD	78.82$_{\pm0.15}$	78.02$_{\pm1.13}$	78.51$_{\pm0.24}$	42.17$_{\pm0.10}$	51.06$_{\pm0.03}$	51.03$_{\pm0.12}$	82.56$_{\pm0.07}$	82.70$_{\pm0.01}$	50.91$_{\pm0.19}$	50.91$_{\pm0.12}$	26.10$_{\pm0.15}$
FEDDYN	**83.31**$_{\pm0.15}$	82.31$_{\pm0.41}$	82.97$_{\pm0.40}$	**50.35**$_{\pm0.27}$	**53.50**$_{\pm0.76}$	**54.32**$_{\pm0.63}$	81.95$_{\pm0.42}$	82.00$_{\pm0.13}$	**51.77**$_{\pm0.03}$	**51.94**$_{\pm0.09}$	25.51$_{\pm0.02}$
FEDCYCLIC	82.45$_{\pm0.18}$	**82.61**$_{\pm0.27}$	**83.49**$_{\pm0.08}$	47.46$_{\pm0.42}$	49.93$_{\pm0.16}$	50.47$_{\pm0.27}$	85.46$_{\pm0.05}$	**87.47**$_{\pm0.09}$	50.25$_{\pm0.03}$	50.68$_{\pm0.03}$	26.44$_{\pm1.68}$
FEDSEQ (ours)	81.89$_{\pm0.28}$	82.19$_{\pm0.26}$	82.77$_{\pm0.12}$	45.87$_{\pm0.45}$	49.26$_{\pm0.40}$	49.63$_{\pm0.42}$	**87.11**$_{\pm0.03}$	**87.48**$_{\pm0.03}$	51.70$_{\pm0.13}$	51.82$_{\pm0.00}$	**28.91**$_{\pm0.21}$
FEDASYNCSEQ (ours)	**83.03**$_{\pm0.31}$	**83.17**$_{\pm0.27}$	**83.57**$_{\pm0.22}$	**50.23**$_{\pm0.11}$	**51.39**$_{\pm0.18}$	**51.27**$_{\pm0.24}$	**86.96**$_{\pm0.07}$	87.20$_{\pm0.01}$	**51.82**$_{\pm0.13}$	**51.88**$_{\pm0.04}$	**27.44**$_{\pm0.35}$
FEDSEQ2PAR (ours)	**83.68**$_{\pm0.14}$	**84.21**$_{\pm0.40}$	**84.26**$_{\pm0.06}$	**51.46**$_{\pm0.23}$	**51.44**$_{\pm0.02}$	**51.72**$_{\pm0.10}$	**87.58**$_{\pm0.15}$	**87.95**$_{\pm0.05}$	**52.75**$_{\pm0.20}$	**52.71**$_{\pm0.05}$	**29.79**$_{\pm0.33}$

FedSeq Details. FEDSEQ and FEDSEQ2PAR build superclients using ψ_{t2v} and the best corresponding $\phi_{(\cdot)}$. Local pre-training runs for 10 epochs chosen from $\{1,5,10,20,30,40\}$ (see Chapter A for details). $R = N_S$ in FEDASYNCSEQ.

FedSeq Outperforms the State of the Art

Table 5.1 shows that FEDSEQ and its extensions are either competitive or outperform other SOTA algorithms on all tasks and datasets, especially on severe heterogeneous data distributions. It is important to underline that both SCAFFOLD and FEDDYN require stateful clients, and SCAFFOLD doubles the size of the communicated message, differently from this work's approaches. Being the extreme case of FEDSEQ with $N_S = 1$, FEDCYCLIC reaches similar performances on most of the datasets, implying that having one single superclient containing all clients does not dramatically increase performances and instead hugely increments the amount of training time, as each client needs to wait for the previous one's update.

Focusing on the extensions of FEDSEQ, both FEDASYNCSEQ and FEDSEQ2PAR improve the baseline's performances on all tasks. By aggregating superclients updates every N_S rounds not only the synchronization between clients and server is less frequent, but the reached accuracy also improves. FEDSEQ2PARachieves the best results in most cases, exploiting the benefits of sequential training and parallelism, being the second best to FEDDYN only in the case of $\alpha = 0.2$ and $\alpha = 0.5$ in CIFAR100. However, differently from FEDSEQ2PAR, FEDDYN relies on stateful clients, posing a significant challenge for real-world deployments with billions of edge devices [8, 7]. In such large-scale settings, individual devices are unlikely to be called upon for multiple training rounds. This transience renders their local states obsolete quickly, compromising their effectiveness in subsequent training iterations. The results obtained by FEDDYN on the more realistic FEMNIST (3,500 clients)

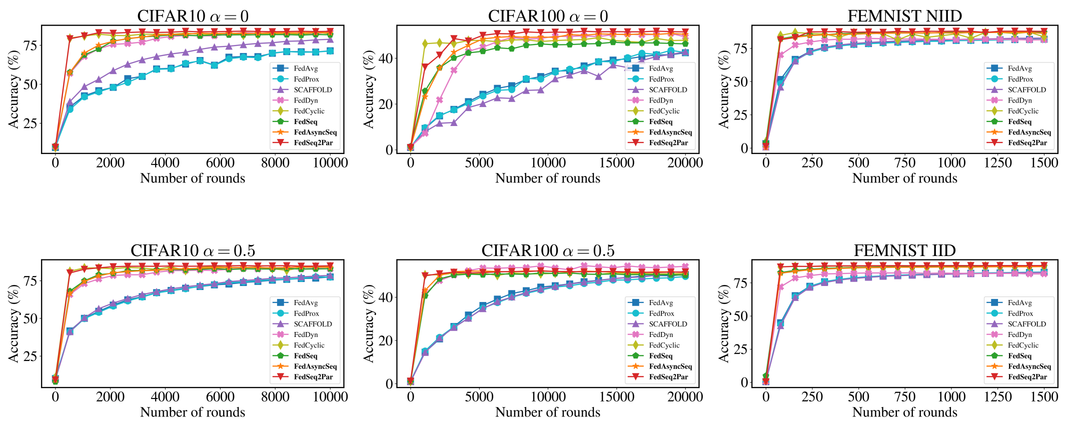
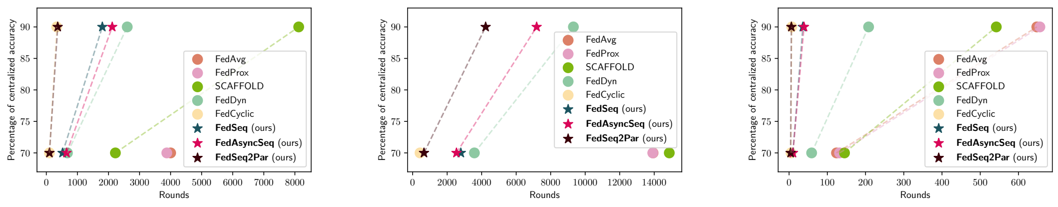


Fig. 5.5 Accuracy convergence plots of FEDSEQ, FEDASYNCSEQ, FEDSEQ2PAR (in bold) and SOTA algorithms on vision datasets. On average, FEDSEQ2PAR is the best-performing algorithm. All the proposed approaches can be distinguished for their improved speed.

underscore this point: it outperforms the baseline FEDAVG by only ≈ 0.4 points in accuracy. In contrast, FEDSEQ's variants, especially FEDSEQ2PAR, achieve a significant improvement of $+6$ points over FEDAVG. These results confirm the limitations of FEDDYN in real-world cross-device scenarios, where its reliance on stateful clients becomes a significant disadvantage.

Convergence Speed. Figure 5.5 evidently shows that FEDSEQ and its extensions not only achieve superior results but also exhibit accelerated performance. Figure 5.6 compares the rounds necessary to each algorithm to reach 70% and 90% of the centralized accuracy on CIFAR10/100 and FEMNIST. FEDSEQ consistently demonstrates significant improvements in convergence speed across all tasks, achieving a speed-up factor of over 18x on FEMNIST presenting high cross-device variability. Achieving superior overall accuracy, asynchronous training, and reduced latency compared to FEDSEQ does not compromise the convergence speed of FEDASYNCSEQ. FEDSEQ2PAR notably improves convergence speed on all tasks and datasets through its STP approach. FEDDYN enhances the convergence rates of FEDAVG but experiences parameter explosion in highly imbalanced settings [8], requiring gradient clipping techniques. Among the considered methods, FEDCYCLIC achieves comparable or better results than FEDSEQ2PAR. However, it is important to note that FEDCYCLIC, as an extreme case of FEDSEQ with $N_S = 1$, eliminates any form of parallelism inherent in distributed and FL settings. The results highlight the



(a) CIFAR10 $\alpha = 0$ (b) CIFAR100 $\alpha = 0$ (c) FEMNIST NIID

Fig. 5.6 Convergence rates in non-*i.i.d.* scenarios. Each plot shows the rounds necessary for each method to reach 70% and 90% of the centralized accuracy. Not all the algorithms reach the 90% target (missing line). FEDSEQ and its variants (*in bold, stars*) outperform the others in all settings.

significance of sequential training for rapid convergence in initial rounds, while the superior performance of FEDSEQ2PAR in later stages underlines the role of parallelism in achieving both improved final performance and convergence speed up.

Communication Costs

Communication is the main bottleneck of federated training [277], due to the overload of the networks and message size. Thus, when comparing the performance of FL algorithms, their impact on the communication cost is of the utmost importance. As already shown in Figures 5.5 and 5.6, all the introduced methods speed up the convergence, implying that fewer communication rounds are needed to reach a target performance. This work additionally compares the number of client-server exchanges required by the proposed method (FEDSEQ) with leading SOTA algorithms. Table 5.2 demonstrates that FEDSEQ achieves **less network communication** thanks to its client-to-client approach. Similar analyses can be easily extended to FEDSEQ2PAR and FEDASYNCSEQ.

Given the total number of clients K, the fraction selected at each round C, the total number of superclients N_S, and the rounds T, the first analyzed scenario assumes all superclients to be equally sized, *i.e.*, $K_{S_i} = K_{S_j} = {}^{K}\!/_{N_S} =: K_S \, \forall i \neq j$. In FEDAVG, the server sends the global model to the $C \cdot K$ selected clients, which then send back the updated version. As summarized in Table 5.2, this process accounts for $2C \cdot K$ exchanges over the network. The same goes for FEDPROX and FEDDYN. SCAFFOLD requires double the communication. In FEDSEQ with equal K_S instead, the server-to-client (S2C) and client-to-server (C2S) communication only happens between the first and last clients of the chain of each superclient respectively. Since

Table 5.2 Number of communication exchanges from server to client (C2S), client to server (S2C) and client to client (C2C) at each round t and across all rounds T.

Method	**S2C**	**C2S**	**C2C**	**Total** @ round	**Total** T rounds
FEDAVG	CK	CK	0	$2CK$	$2TCK$
FEDPROX	CK	CK	0	$2CK$	$2TCK$
FEDDYN	CK	CK	0	$2CK$	$2TCK$
SCAFFOLD	$2CK$	$2CK$	0	$4CK$	$4TCK$
FEDCYCLIC	1	1	$CK-1$	$CK+1$	$T(CK+1)$
FEDSEQ $K_{S_i} = K_{S_j}$	CN_S	CN_S	$\left(\frac{K}{N_S}-1\right)CN_S$	$C\left(N_S+K\right)$	$TC(N_S+K)$
FEDSEQ $K_{S_i} \neq K_{S_j}$	CN_S	CN_S	$\sum_{S_i \in \mathscr{S}_t} K_{S_i} - CN_S$	CN_S+ $+\sum_{S_i \in \mathscr{S}_t} K_{S_i}$	TCN_S+ $+\sum_{t\in[T]} \sum_{S_i \in \mathscr{S}_t} K_{S_i}$

the server selects $C \cdot N_S$ superclients, the process sums up to $2C \cdot N_S$ exchanges. Moreover, within each superclient, the clients exchange messages following the chain, for a total of $K_S - 1$ transmissions $\forall S$. Considering all $C \cdot N_S$ groups involved, this is equivalent to

$$(K_S - 1)C \cdot N_S = \left(\frac{K}{N_S} - 1\right) C \cdot N_S = C \cdot K - C \cdot N_S. \tag{5.6}$$

By summing everything up, the total is $2C \cdot N_S + C \cdot K - C \cdot N_S = C(N_S + K)$. Since $N_S < K$, **the overall communication cost of FEDSEQ is smaller than FEDAVG, FEDPROX, FEDDYN and SCAFFOLD**. If superclients are not equally sized, the client-to-client (C2C) cost is $\sum_{S_i \in \mathscr{S}_t}(K_{S_i} - 1) = \sum_{S_i \in \mathscr{S}_t} K_{S_i} - C \cdot N_S$, where $|\mathscr{S}_t| = C \cdot N_S$, and the total becomes $C \cdot N_S + \sum_{S_i \in \mathscr{S}_t} K_{S_i}$, *i.e.*, depends on the size of the selected superclients. However, to ensure a fair comparison, the proposed experiments select $K_{S,max}$ such that $K/K_{S,max} \approx N_S$, *i.e.*, most of the superclients are of the same size, falling back to the first scenario.

Results on NLP Datasets.

FEDSEQ and its variants are additionally evaluated on text classification tasks, namely next character prediction with SHAKESPEARE [394] and next word prediction with STACKOVERFLOW [424].

The results confirm the behavior of FEDSEQ, FEDASYNCSEQ and FEDSEQ2PAR already noted on the vision datasets. These methods are confirmed to be the fastest and best performing across all datasets, both in *i.i.d.* and non-*i.i.d.* settings (Figure 5.7 and table 5.1).

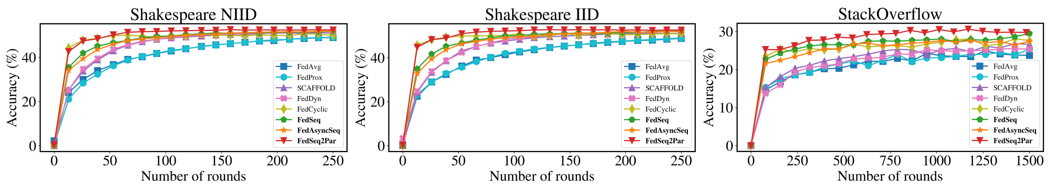
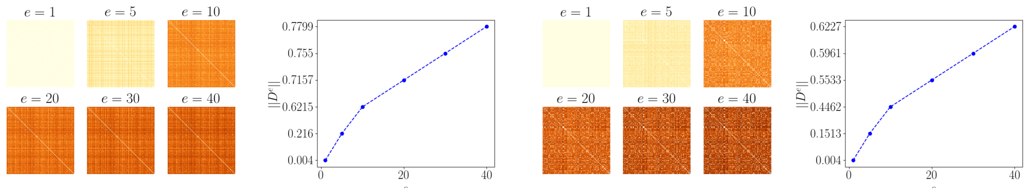


Fig. 5.7 Accuracy convergence plots of FEDSEQ, FEDASYNCSEQ, FEDSEQ2PAR (in bold) and SOTA algorithms on NLP datasets. On average, FEDSEQ2PAR is confirmed the best-performing algorithm.



(a) CIFAR100. Similarity matrix D^e. (b) CIFAR100. Trend of $||D^e||$. (c) CIFAR10. Similarity matrix D^e. (d) CIFAR10. Trend of $||D^e||$.

Fig. 5.8 Effect of pre-training $K = 500$ local models for $e \in \{1,5,10,20,30,40\}$ epochs on CIFAR100 and CIFAR10. From the trends on both datasets, it can be noted that after $e = 10$ the slope of the curve decreases.

Ablation Studies

This section discusses the impact of each method component introduced in Section 5.2.2.

Local Pre-Training. All grouping criteria introduced in Section 5.2.2 rely on an approximation, denoted by $\tilde{D}_k$ and derived using function ψ, of each client's data distribution. Regardless of the chosen approximation method, the initial step for constructing superclients involves a pre-training phase performed locally on each client's device. This phase lasts for e epochs. During pre-training, a randomly initialized model f_{w_0} is trained on the local data for all methods except ψ_{t2v}, where a pre-trained network's classifier layers (referred to as the probe network) are fine-tuned for the specific local task. The resulting model parameters are then used to estimate the data distributions of individual users without compromising privacy.

The number of pre-training epochs (e) needs to be carefully chosen. Ideally, it should be large enough for the model to learn from the local training set but small

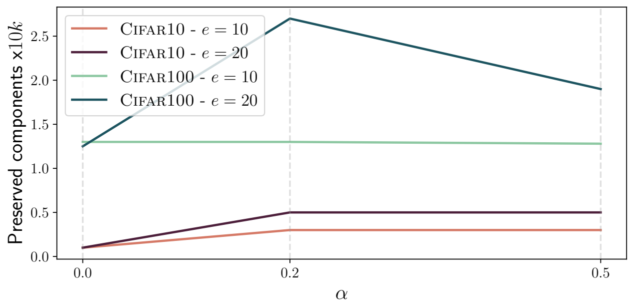
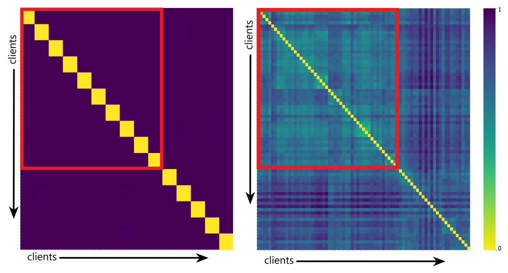


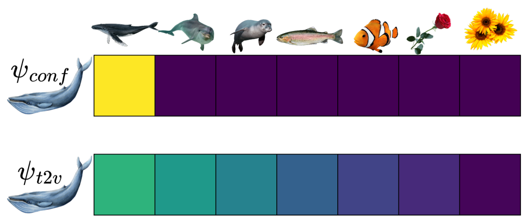
Fig. 5.9 CIFAR datasets. Ratio of the preserved components after applying PCA with 90% of explained variance when varying the number of local epochs e.

enough to avoid overwhelming the clients' devices. Models trained on similar data distributions are expected to exhibit greater similarity compared to those trained on disparate distributions [421]. The experiments evaluate different values for $e \in \{1, 5, 10, 20, 30, 40\}$. For each value, a similarity matrix is constructed. This matrix captures the cosine distance between models trained on each client's data for e epochs. Formally, $D^e := \{D^e_{ij} = \frac{\boldsymbol{w}^i_e \cdot \boldsymbol{w}^j_e}{||\boldsymbol{w}^i_e||\,||\boldsymbol{w}^j_e||}\}$, where $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ represent the parameters of client i and j models trained for e epochs, respectively (all client pairs are considered, *i.e.*,$(i, j) \in (K \times K)$). Figure 5.8 depicts these matrices as heatmaps for the CIFAR100 dataset. Figure 5.8b shows the trend of the matrix norm ($||D_e||$) for each value of e. It is evident that 5 epochs are sufficient to achieve significantly different models. The rate of change in similarity reduces after 10 epochs of pre-training. Therefore, considering the trade-off between the informativeness of the trained models and the computational burden on clients, a default value of $e = 10$ is chosen for client pre-training, regardless of the data distribution approximation method used. Notably, these results are consistent across both datasets, indicating that the chosen network architecture can effectively capture the characteristics of both datasets.

Clients' Data Distribution Estimation. The proposed method utilizes the parameters (ψ_{clf}) or pre-trained model predictions (ψ_{conf}, ψ_{t2v}) to compute a privacy-preserving estimate of the clients' dataset distribution. To mitigate the *curse of dimensionality* [426] on the classifier parameters in ψ_{clf}, PCA [427] is applied, keeping 90% of the explained variance. Figure 5.9 shows that the percentage of preserved components decreases with the complexity of the dataset, *e.g.* fewer components are needed for CIFAR10, and increases directly proportional to E. As for ψ_{conf}, not to



(a) Distance matrices on ψ_{conf} (*left*) and ψ_{t2v} (*right*) tasks embeddings.



(b) ψ_{conf} and ψ_{t2v}'s tasks embeddings similarity.

Fig. 5.10 CIFAR100, α= 0. **(a)** Focus on 75 clients. Each group of 25 clients has access to either images of aquatic mammals, fishes or flowers. **(b)** Client with images of *whales*. Comparison of embedding distances with clients containing images of progressively different entities. ψ_{t2v} accurately recognizes the similarities between animals, in contrast to ψ_{conf}.

severely impact the original dataset, $\mathscr{D}_{pub}$ is built using 10 images per class from the test set for computing the *confidence vectors* (Equation (5.1)). Once $\mathscr{D}_{pub}$ has served its purpose, it is not used again.

Since a public dataset capturing the overall global distribution may not be available in realistic settings, this paper introduces ψ_{t2v}, based on Task2Vec [421], which presents two main advantages: *i*) no external dataset is required, and *ii*) clients only fine-tune the classifier, reducing the latency. Following [421], pre-trained ResNet18 is used as backbone for image classification tasks. To better understand the difference in their behavior, the embeddings of ψ_{t2v} (*right*) and ψ_{conf} (*left*) are compared in Figure 5.10a. Specifically, it illustrates the distance between their embeddings computed over the first 75 clients of CIFAR100 with α= 0. The first 25 clients exclusively have images of aquatic mammals (beavers, dolphins, otters, seals, and whales), the next 25 clients have images of fishes (aquarium fishes, flatfishes, rays, sharks, and trouts), and the last 25 clients have images of various flowers. Vectors from ψ_{conf} lack class similarity representation, while the ψ_{t2v} distance matrix reveals that clients with fish and aquatic mammal images (*red square*) cluster together more closely than those with flower images. In Figure 5.10b, one client with only whale images is compared in terms of distance with other clients having progressively dissimilar images to whales. Once again, ψ_{t2v} accurately recognizes the similarities between animals, in contrast to ψ_{conf}. This behavior can be explained by looking at the embedding of the k-th client with samples belonging to class $c \in [N_c]$ given by

Table 5.3 FEDSEQ baselines: comparison of grouping criteria by varying ϕ, ψ and τ. Results in terms of accuracy (%).

Method	■	ϕ	τ	CIFAR10			CIFAR100			FEMNIST		SHAKESPEARE	
				$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.5$	NIID	IID	NIID	IID
FEDSEQ	-	Random	-	81.90	82.09	82.12	46.39	48.62	49.44	87.07	87.49	51.55	51.84
	clf	K-means	Euclidean	**82.30**	81.78	82.48	44.91	48.74	49.60	87.07	87.45	51.78	51.70
		Greedy	Cosine	79.95	82.06	83.32	45.22	48.92	49.62	87.12	87.42	51.72	51.88
	conf	K-means	Euclidean	82.04	81.99	82.37	43.55	49.43	49.79	87.10	**87.50**	51.79	51.87
		Greedy	KL	82.21	82.20	82.22	45.97	49.56	49.82	87.01	87.46	51.70	51.65
		Greedy	Gini Index	82.09	81.85	82.71	45.79	48.98	49.61	87.05	87.42	51.59	**51.98**
	t2v	Greedy	Norm-Cosine	82.28	82.48	**82.86**	**46.51**	**50.06**	**50.31**	87.11	87.48	51.65	51.82
		ICG	Euclidean	82.05	**82.51**	82.54	46.33	49.13	49.86	**87.18**	87.45	**51.84**	51.77
FEDASYNCSEQ	*t2v*	$*$	$*$	83.40	83.37	83.85	50.38	51.64	51.58	86.96	87.20	51.93	52.02
FEDSEQ2PAR	*t2v*	ICG	Euclidean	**83.86**	**84.66**	**84.35**	**51.23**	**51.41**	**51.78**	**87.58**	**87.96**	**52.84**	**52.54**

(*): (Greedy, Norm-Cosine) for CIFAR10/100; (K-means, Euclidean) for FEMNIST and SHAKESPEARE.

ψ_{conf} is

$$p_k[i] \approx \begin{cases} 1 & \textit{if } i = c, \\ 0 & \textit{otherwise}. \end{cases} \tag{5.7}$$

This aligns with the expectations, as $f_{\boldsymbol{w}_k^0}$ is trained to classify observations with label c. As a result, the embeddings of clients seeing different classes (regardless of the similarity of the depicted subjects) are equally distant. However, this contradicts the intuitive understanding, as one would expect that similarities in data distributions would manifest as closeness in the vector space. In contrast, the distance between ψ_{t2v} embeddings aligns with the intuition on semantic and taxonomic relations among entities. This behavior is evidently reflected in its performance in Table 5.3, where ψ_{t2v} consistently outperforms the other approaches.

Grouping Criterion. Table 5.3 compares the different combinations of grouping criteria G_S. As for ψ_{kmeans}, a reasonable value for the number of clusters is N_c, and the Euclidean distance is used to compare the resulting superclients. The confidence vectors extracted by ψ_{conf} have the form of a probability distribution (Section 5.2.2), additionally comparable via disomogeneity measures such as the KL divergence and the Gini Index. The normalized embeddings obtained with Task2Vec are compared with the cosine distance ("Norm-Cosine" in the Table) [421]. Notably, ϕ_{rand} returns groups obtaining competitive results with the other grouping methods. The reason lies in statistical considerations on the cross-device setting: with the number of clients being large in all datasets, a randomly created group is unlikely to contain clients belonging all to the same data distribution. ψ_{t2v} achieves the best performance across all settings, demonstrating effective capture of task similarities. ϕ_{icg} is selected

Table 5.4 Parallelism and test accuracy: FEDSEQ *vs*. FEDSEQ2PAR.

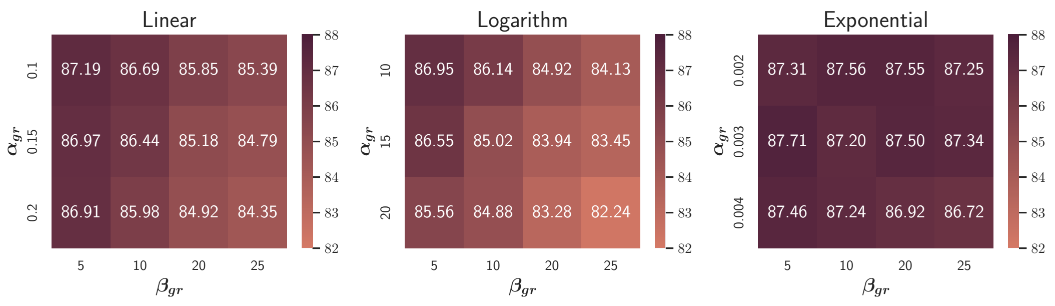
Dataset	N_S	$\overline{N}_S$	**Parallelism**	**Accuracy** (α= 0 / NIID)	
	FEDSEQ	FEDSEQ2PAR	↑	FEDSEQ	FEDSEQ2PAR
CIFAR10	50	**104**	2.09x	81.89	**83.68**
CIFAR100	50	**113**	2.26x	45.87	**51.46**
FEMNIST	175	**383**	2.19x	87.11	**87.58**
SHAKESPEARE	25	**52**	2.09x	51.70	**52.75**
STACKOVERFLOW	1900	**5966**	3.14x	28.91	**29.79**

as grouping method due to its satisfying results and efficiency, useful in the dynamic creation of superclients especially with several groups.

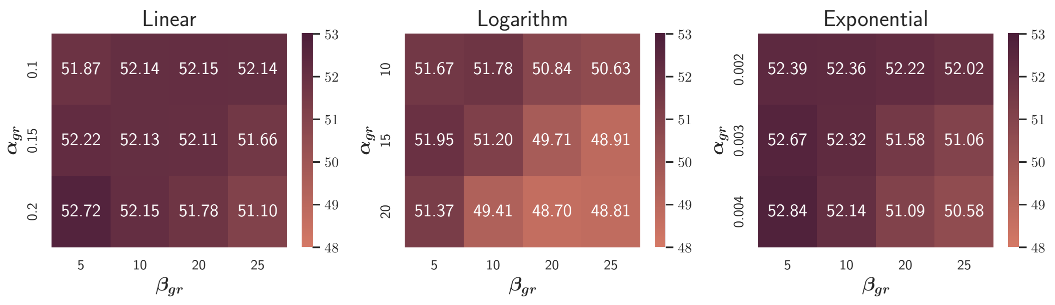
FedSeq2Par. As described in Section 5.2.2, the Sequential-to-Parallel approach used in FEDSEQ2PAR is based on the function $f_{gr}(\alpha_{gr}, \beta_{gr}, t)$ (Equations (5.3) to (5.5)), that defines the number of superclients at each round t. This section aims to understand which growth function better suits the analyzed settings (*linear*, *logarithmic*, or *exponential*) and the effect of the parameters α_{gr} (growth rate) and β_{gr} (initial number of superclients). α_{gr} is chosen so that a fully parallel scenario is reached in the last rounds, while favoring sequential training at the beginning. $\beta_{gr} \in \{5, 10, 20, 25\}$ for all datasets except for the larger STACKOVERFLOW, for which $\beta_{gr} \in \{50, 100, 200, 250\}$. Figure 5.11 analyzes the impact of these parameters on the non-*i.i.d.* splits of FEMNIST and SHAKESPEARE. Notably, starting with the smallest number of superclients β_{gr} consistently yields superior performance, as it exploits sequentiality more. f_{exp} has the best and most consistent results. Due to the uniformity of these results, the same configuration is maintained across all datasets. FEDSEQ2PAR is further compared with FEDSEQ in terms of number of superclients and final performance in Table 5.4. For FEDSEQ2PAR, it is reported the average number of superclients $\overline{N}_S$ across rounds. Notably, $\overline{N}_S$ is constantly larger than N_S, implying a more parallelized scenario *on average* with FEDSEQ2PAR w.r.t. FEDSEQ even if $\beta_{gr} \ll N_S$. This behavior positively reflects on the final performance, confirming the efficacy of the STP approach.

5.2.4 Privacy Robustness

Recent FL literature has highlighted the potential for attackers to reconstruct sensitive information through the clients' updates [252]. Thus, concerns on the potential privacy implications of the *client-to-client* sequential training approach introduced



(a) FEMNIST



(b) SHAKESPEARE

Fig. 5.11 Sensitivity of FEDSEQ2PAR to f_{gr} and the growth parameters α_{gr} and β_{gr}. Results in test accuracy (%) on the NIID split.

by FEDSEQ arise. Specifically, this extension poses the question: *does* FEDSEQ*'s client-to-client sequential training facilitate the retrieve of previous users' personal information by a malicious client?* To answer, FEDSEQ is evaluated against two famous attacks, namely the **label flipping** [418] (LFA) and the **GAN recovery** attacks (GRA) [350], and study potential private information leakages. The well-known gradient inversion attack [252] is not considered here, as its assumptions do not align with our approach (*e.g.*, access of the attacker to both initial and updated models, knowledge of private labels). Differently, in this case, clients only receive the updated parameters from the previous user and potentially malicious clients are not aware of other users' private labels.

Label Flipping Attack

LFA is an *active* privacy attack aiming at deteriorating the global model performances by switching labels at training time. Here, the focus is on models solving the classification task. To mislead the global model classification ability, the set of malicious clients $\mathscr{A} := \{a_i\}_{i=1}^{L \cdot K} \subseteq \mathscr{C}$ with $L \in [0,1]$ willingly swaps the labels of their local data following a set of criteria $\{\gamma_i\}_{i=1}^{L\cdot K}$. The criterion γ_i defines the labels to be swapped during the attack for each attacker a_i. For instance, $\gamma_i = \gamma_j$ implies that the attackers a_i and a_j will swap the same classes. This work tests two possible situations:

1. Different attackers swap distinct classes, *i.e.*, each attacker a_i chooses its γ_i independently (γ_{random}),
2. All the attackers swap the same classes, *i.e.*, $\gamma_i = \gamma_j \, \forall i, j \in \mathscr{A}$ (γ_{fixed}).

GAN Recovery Attack

GRA is a *passive* privacy attack that aims at reconstructing other clients' private information using GAN architectures [351]. It is important to highlight that the primary objective of GANs is to *generate* samples that closely resemble those found in the training set without direct access to the original ones. GANs rely on interactions with a *discriminative* deep neural network to learn and capture the underlying data distribution [350]. They are trained to mimic the images encountered by the discriminative network, starting from random initialization. However, a potential concern arises when the discriminator is trained on private data, as it can potentially be exploited to train a generator network capable of reconstructing the sensitive data. This poses significant privacy and security concerns. Formally, the GANs' optimization problem [350] is

$$\min_{\boldsymbol{w}_G} \max_{\boldsymbol{w}_D} \sum_{i=1}^{n} \log f(x_i, \boldsymbol{w}_D) + \sum_{j=1}^{n} \log(1 - f(g(z_j, \boldsymbol{w}_G), \boldsymbol{w}_D)), \tag{5.8}$$

where $f(x, \boldsymbol{w}_D) : \mathscr{X} \rightarrow \mathscr{Y}$ is a discriminative network parametrized by $\boldsymbol{w}_D$ that, given an image, outputs a class label. The generative network $g(z, \boldsymbol{w}_G) : \mathscr{X} \rightarrow \mathscr{X}$ receives random noise as input and outputs an image. x_i is the original image and $g(z_j)$ is a randomly generated one. In GRA, at round t, an attacker $a_i \in \mathscr{C}$ disguised as a client exploits the incoming trained model $\boldsymbol{w}_t$ as the discriminator of a *GAN*, *i.e.*, $\boldsymbol{w}_D \leftarrow \boldsymbol{w}_t$. The generator g is then trained for E_a epochs to reconstruct inputs similar to the ones previously accessed by $\boldsymbol{w}_t$, thus breaking the clients' privacy.

Table 5.5 Label Flipping Attack experiments after $1k$ rounds. Results in accuracy (%) and drop in accuracy (↓) w.r.t. to the reference. In bold smaller drops in each attack. Symbols: "○" (negligible or non-existing drops), "Fixed" (γ_{fixed}) and "Random" (γ_{random}).

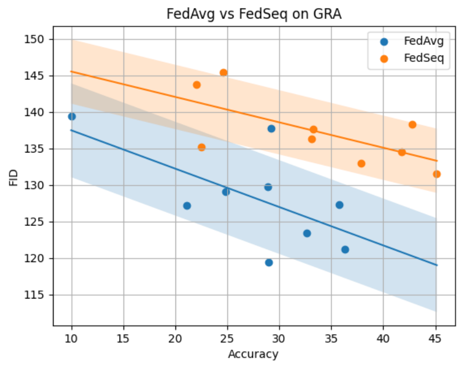
L_S	L	**Swapped Labels**	γ	CIFAR10				**Swapped Labels**	γ	CIFAR100			
				FEDSEQ		**FEDAVG**				**FEDSEQ**		**FEDAVG**	
				Accuracy ↑	Drop ↓	Accuracy ↑	Drop ↓			Accuracy ↑	Drop ↓	Accuracy ↑	Drop ↓
0.1	0.1			76.06	○	48.72	○			38.81	−0.89	12.57	**−0.28**
	0.5			75.92	**−0.20**	47.55	−1.14			38.87	−0.83	12.35	**−0.50**
0.3	0.1	0↔2	Fixed	76.25	**−0.15**	47.74	−0.95		Fixed	39.66	○	12.45	−0.40
	0.5			74.75	−1.35	48.30	**−0.39**			39.04	**−0.66**	11.55	−1.30
0.5	0.1			75.65	**−0.45**	47.55	−1.14			39.23	**−0.47**	12.35	−0.50
	0.5			75.50	**−0.60**	47.44	−1.25	INTRASC		38.40	**−1.30**	11.02	−1.83
0.1	0.1			76.05	○	47.92	−0.77			39.36	−0.34	13.23	○
	0.5			75.50	**−0.60**	47.93	−0.76			39.59	○	13.05	○
0.3	0.1	3↔5	Fixed	75.75	**−0.35**	47.80	−0.89		Random	39.67	○	13.23	○
	0.5			75.28	**−0.82**	47.74	−0.95			38.88	−0.82	12.47	**−0.38**
0.5	0.1			76.15	○	47.93	−0.76			39.68	○	13.05	○
	0.5			75.09	**−1.01**	46.94	−1.75			38.98	**−0.72**	11.27	−1.58
0.1	0.1			76.65	○	47.36	−1.33			40.11	○	12.37	−0.48
	0.5			76.01	○	48.12	−0.58			40.03	○	12.20	−0.65
0.3	0.1		Fixed	75.79	**−0.31**	48.29	−0.40		Fixed	39.29	−0.41	13.08	○
	0.5			75.19	**−0.91**	46.42	−2.27			38.73	−0.97	12.34	**−0.51**
0.5	0.1			75.35	−0.75	48.12	**−0.58**			38.95	−0.75	12.20	**−0.65**
	0.5	0↔2		75.09	**−1.01**	46.94	−1.75	EXTRASC		38.46	−1.24	12.30	**−0.55**
0.1	0.1	↔3↔5		76.52	○	48.65	○			39.17	−0.53	13.12	○
	0.5			75.71	**−0.39**	47.52	−1.17			39.12	−0.58	12.83	○
0.3	0.1		Random	75.98	**−0.12**	47.01	−1.68		Random	39.54	○	13.06	○
	0.5			74.75	**−1.35**	46.70	−1.99			37.90	−1.80	11.81	**−1.04**
0.5	0.1			75.93	**−0.17**	47.52	−1.17			39.03	−0.67	13.12	○
	0.5			73.46	−2.64	46.49	**−2.20**			37.76	−1.94	11.94	**−0.91**
		Reference accuracy: **FedSeq** 76.10% - **FEDAVG** 48.69%						Reference accuracy: **FedSeq** 39.70% - **FEDAVG** 12.85%					

FedSeq Privacy Resilience

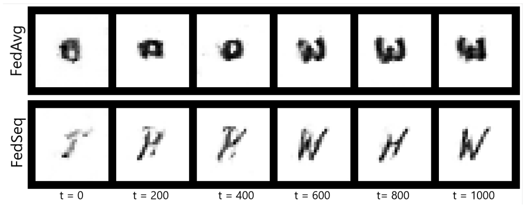
This section provides quantitative results of FEDSEQ's resilience against the *LFA* and *GRA* attacks. We show that FEDSEQ not only does not introduce additional privacy liabilities w.r.t. FEDAVG, but it learns more robust models.

FedSeq against LFA. Table 5.5 summarizes the results on the different setups proposed to evaluate the robustness of FEDSEQ to the LFA attack. The fraction of malicious *superclients* L_S is distinguished from the fraction of malicious clients *within* each malevolent superclient L. The corresponding fraction of attackers in FEDAVG becomes $L_S \cdot L$. L_S is tested in $\{0.1, 0.3, 0.5\}$ and L in $\{0.1, 0.5\}$. For example, $L_S = 0.1$ and $L = 0.5$ implies that 10% of the superclients are malevolent, and 50% of their clients are attackers.

Following [418], the four swapped classes in CIFAR10 are: *airplanes* (label 0) exchanged with *birds* (label 2), and *dogs* (label 5) with *cats* (label 3). The additional setting where all the aforementioned classes (0,2,3,5) are swapped at the same time is evaluated as well. When using CIFAR100 instead, the concept of "superclass" proper of the dataset is exploited (*e.g.*, *aquatic mammals*, *flowers*). The swap happens either between 20 classes that do not belong to the same superclass, *i.e.* one class for each



(a) FID scores after GRA attack on FEDSEQ and FEDAVG



(b) GRA attacker images reconstruction

Fig. 5.12 **(a)** GRA attack on global model with different accuracy. The resulting FID scores on FEDSEQ are consistently higher, implying a less effective attack. **(b)** Examples of images reconstructed by the GRA attacker at distinct rounds.

superclass (*e.g.*, *dolphins* and *roses*), or between pairs of 20 labels belonging to the same superclass (*e.g.*, *dolphins* with *whales*). The former is referred to as EXTRA$_{SC}$, and the latter as INTRA$_{SC}$.

To evaluate FEDSEQ against the easiest scenario for the attacker, all the experiments are run with α= 100 on both CIFAR10 and CIFAR100, meaning that all K clients see all the classes, and the LFA is always feasible. T is set equal to $1k$. Table 5.5 shows the results of the attack on each proposed configuration, analyzing both the accuracy of the model on the overall test set and the drop w.r.t. the reference experiment without attackers. On average, the *fixed* attacks are more effective than the *random* ones. The reason behind this behavior is intuitive: when using γ_{fixed}, the attackers never let the model learn the correct patterns for classifying the swapped labels, differently from the random acting. Swapping 4 labels in CIFAR10 rather than 2 brings on average more damage. For CIFAR100, the EXTRA$_{SC}$ attacks are significantly more effective: the model likely learns some common features for images belonging to the same superclass, leading to a reduced efficacy of the INTRA$_{SC}$ attack. Importantly, FEDSEQ outperforms FEDAVG on most scenarios both in terms of accuracy and drop w.r.t. to the reference: this means FEDSEQ is still able to achieve faster convergence if under attack, and is more robust than FEDAVG.

FedSeq against GRA. The *Fréchet inception distance* (FID) [428] assesses the quality of the images created by a generative model. Given a dataset $\mathscr{D}$ and its reconstruction $\hat{\mathscr{D}}$, the FID measures the distribution of their features, extracted using an InceptionV3 network [429], using the *Fréchet distance* [428]. A lower score indicates better-quality images. Within the context of an attack, the FID has to be as large as possible, signifying the attacker's inability to reconstruct private data effectively. Unlike the approach in [350], the presented approach refrains from incorporating a "fake" class in the classifier, deeming it unrealistic. Instead, the attacker is allowed to utilize an additional binary dense layer on top of the model to distinguish between "fake" and "real" data. Figure 5.12a results from attacks on models with varying levels of accuracy. It is clear that the attack conducted on FEDSEQ consistently yields higher FID scores in comparison to FEDAVG, underscoring its enhanced privacy characteristics. Figure 5.12b shows some examples of images reconstruction at different rounds.

5.2.5 Discussion

This work addressed the challenges of statistical heterogeneity in Federated Learning by proposing FEDSEQ (Federated Learning via Sequential Superclients Training). FEDSEQ leverages sequential training among heterogeneous client groups (superclients) to achieve more robust models before server-side averaging. The work explores various strategies for effective client grouping based on data distribution. To mitigate waiting times caused by slow superclients, FEDASYNCSEQ enables asynchronous communication between clients and the server. Finally, FEDSEQ2PAR dynamically adjusts the number of superclients per round to exploit both sequential and parallel processing effectively.

Extensive evaluations on multiple FL benchmarks demonstrate the efficacy of these approaches in terms of final performance, convergence speed, and privacy preservation. Future work could include a deeper analysis of FEDSEQ's convergence properties. Theoretical and empirical studies on large-scale vision datasets could offer valuable insights. Additionally, exploring the application of FEDSEQ beyond classification tasks represents a promising avenue for further research. Finally, mitigating potential catastrophic forgetting within superclients due to client heterogeneity is crucial. Developing techniques to address this issue and preserve knowledge across clients would significantly enhance FEDSEQ's overall effectiveness.

5.3 Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning

Federated Learning has gained significant attention as a potential approach for tackling domain shift in real-world Semantic Segmentation (SS) tasks while preserving data privacy. However, a common limitation in existing FL research is the unrealistic assumption of labeled data availability on client devices.

This work presents two key contributions:

- **Federated Source-Free Domain Adaptation** (FFREEDA): a novel task formulation for learning from unlabeled client data while leveraging a public, labeled dataset on the server side.
- **Learning Across Domains and Devices** (LADD): a novel method for tackling the FFREEDA task. utilizes the knowledge of a pre-trained model by employing self-supervision with specialized regularization techniques for local training. Additionally, it introduces a novel federated clustering aggregation scheme based on the client styles.

Furthermore, this paper establishes a new benchmark for studying SS tasks within an FL setting. Details of this benchmark are provided in Chapter 6.

The code is available at https://github.com/Erosinho13/LADD.

5.3.1 Motivation

FL offers a promising approach for tackling real-world vision tasks using data collected from various users in diverse scenarios, while preserving data privacy. For instance, FL can be applied in the context of semantic segmentation for self-driving

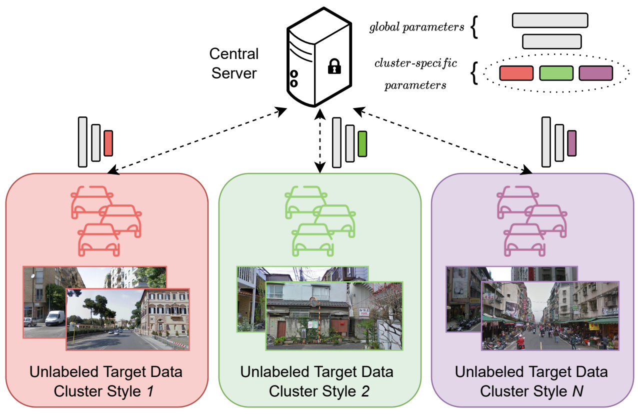

Central Server
global parameters
cluster-specific parameters
Unlabeled Target Data Cluster Style 1
Unlabeled Target Data Cluster Style 2
Unlabeled Target Data Cluster Style N

Fig. 5.13 Overview of FFREEDA. The clients' data is unlabeled and the source labeled dataset is kept on the server. Clients having similar styles are clustered together. Local training leverages both global and cluster-specific model parameters.

cars, enabling obstacle detection and avoidance tasks [271]. However, a common limitation in existing FL research for vision tasks is the assumption of labeled data availability on client devices. This assumption is impractical due to the high cost and significant manual effort associated with dense pixel-level annotations [297].

This work introduces a novel federated learning setting for semantic segmentation that is more applicable to real-world autonomous driving applications. This setting, denoted as Federated source-Free Domain Adaptation (FFREEDA), allows the server to pre-train a model on labeled source data. However, differently from to the Source-Free Domain Adaptation [430] setting, further access to the source data is prohibited after this pre-training step.

In the setting proposed in FFREEDA, clients only have access to their own unlabeled target datasets, which they cannot share with other clients or the server. The scenario focuses on real-world situations with multiple clients, each possessing a limited number of images. Following the pre-training phase, the training becomes fully unsupervised. However, the objective of FFREEDA goes beyond solving a traditional multi-target domain adaptation problem in SS. It aims to address specific challenges that arise in FL settings, such as statistical and system heterogeneity [45, 298], communication bottlenecks [431], and client privacy preservation [44, 329]. To the best of our knowledge, no prior work has tackled this combination of problems simultaneously.

To address the FFREEDA problem, a novel FL algorithm named Learning Across Domains and Devices (LADD) is proposed. LADD assumes the presence of multiple underlying data distributions across the clients. For instance, self-driving cars within the same city might collect visually similar images due to geographical proximity. Conversely, different weather conditions could lead to variations in the local datasets.

LADD addresses this heterogeneity by **clustering clients based on the styles of their images**, aiming to group clients with *similar* data distributions. This clustering helps to match the clients with their actual latent distributions. To improve communication efficiency and minimize parameter duplication, as shown in Figure 5.13, LADD splits the model's parameters into two categories:

- Shared parameters, globally aggregated across all clients.
- Cluster-specific parameters, only aggregated across clients within the same cluster.

Furthermore, LADD leverages the source dataset during pre-training by employing style transfer data augmentation [432]. This technique randomly incorporates target styles into the source images, mimicking the target distributions encountered by clients. Finally, LADD utilizes self-training with a custom pseudo-labeling strategy and incorporates regularization techniques to stabilize the training process.

5.3.2 Federated source-Free Domain Adaptation

This section details the proposed algorithm in its pre-training strategy, aggregation phase and adaptation techniques. The procedure is summarized in Figure 5.14 and Algorithm 8.

Server-side Pre-Training with Style Transfer

The first stage of LADD involves pre-training the model on the labeled source dataset $\mathscr{D}^S$. To mitigate domain shift between source and target data and improve the generalization of the pre-trained model, a style transfer technique based on Fourier Domain Transform (FDA) [432] is employed. This technique aims to bring the styles of source and target images closer.

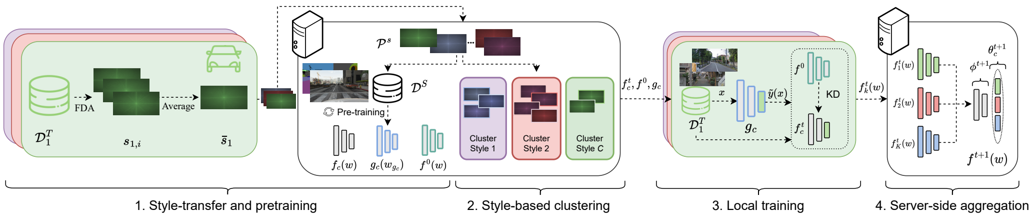


Fig. 5.14 Overview of LADD. 1) Each client k extracts the average style $\bar{s}_k$ of its local data $\mathscr{D}_k^T$ using FDA. At server-side, the collected styles $\mathscr{P}^s$ are applied to the source dataset $\mathscr{D}_s$ during the supervised pre-training. 2) Clients are clustered according to their style. 3) At client-side, the cluster-specific teacher g_c outputs the pseudo-labels, used for training f_c^t, leveraging KD from the pre-trained model. 4) At the server-side aggregation, global ($\boldsymbol{\Phi}^{t+1}$) and cluster-specific parameters ($\boldsymbol{\theta}_c^{t+1}$) are separately aggregated.

During pre-training, client styles are transferred to the server and applied to $\mathscr{D}^S$. Specifically, each client extracts a style representation, denoted as s_k, from its images. This style is obtained from a window of width l_s located at the center of the amplitude spectrum of the image, as described in [432]. Importantly, this window captures the lowest spatial frequency coefficients, which do not contain sensitive scene content, thus preserving client privacy.

The server-side maintains a pool of styles, denoted as $\mathscr{P}^s$, which is the union of average styles ($\bar{s}_k$) collected from all clients (*i.e.*, $\mathscr{P}^s = \bigcup_{k\in\mathscr{C}}\{\bar{s}_k\}$). The randomly initialized model $f(\boldsymbol{w})$ is then trained on the source dataset $\mathscr{D}^S$ with data augmentation using random styles from $\mathscr{P}^s$.

Crucially, the style information is never shared directly among clients. Furthermore, only a small number of images is needed from each client to compute the average style $\bar{s}_k$. After pre-training, the source dataset is no longer used in the training process.

Style-based Clustering

Federated Learning settings often involve clients with diverse data distributions. This heterogeneity can hinder performance if models are naively aggregated across clients. For instance, self-driving cars operating in the same geographical region might collect visually similar data, while those in distant locations might encounter significantly different environments. Additionally, clients may possess limited datasets, restricting their ability to generalize solely from local training [7].

Algorithm 7 LADD Clustering Selection Algorithm

```
Require: Clients $\mathscr{C}$, target datasets $\mathscr{D}_k^T \forall k \in [K]$, function $\mathscr{G}amma$ assigning each
    client to one of the $G$ clusters, hyper-parameters $n, m, N \in \mathbb{N}_0$, $m < n$
 1: for $\text{H} \in [n]_m := \{m, m+1, \dots, n-1\}$ do
 2:     for $n \in [N]$ do
 3:         $\mathscr{G}_\text{H} \leftarrow$ K-MEANS
 4:         Compute $a_k(\mathscr{G}_\text{H}) = \text{INTRACLUSTERDIST}(\mathscr{G}_\text{H}, k) \forall k \in \mathscr{C}$
 5:     end for
 6:     $\mathscr{G}_\text{H} = \arg\min_{\mathscr{G}_\text{H}} \sum_{k \in \mathscr{C}} a_k(\mathscr{G}_\text{H})$
 7:     Define $a_k^\text{H} := a_k(\mathscr{G}_\text{H}) \forall k \in \mathscr{C}$
 8:     Compute $b_k^\text{H} := b_k(\mathscr{G}_\text{H}) = \text{INTERCLUSTERDIST}(\mathscr{G}_\text{H}, k) \forall k \in \mathscr{C}$
 9:     Compute $\bar{\sigma}(\mathscr{G}_\text{H}) = \text{SILHOUETTESCORE}(a_k^\text{H}, b_k^\text{H} \forall k \in \mathscr{C})$
10: end for
11: return $\mathscr{G} = \arg\max_\text{H} \bar{\sigma}(\mathscr{G}_\text{H})$

12: function INTRACLUSTERDIST($\mathscr{G}, k$)
13:     return $\frac{1}{|\Gamma_\mathscr{G}(k)|-1} \sum_{h \in \Gamma_\mathscr{G}(k), h \neq k} \|k, h\|_2$
14: end function
15: function INTERCLUSTERDIST($\mathscr{G}, k$)
16:     return $\min_{c \in \mathscr{G}, c \neq \mathscr{G}amma_\mathscr{G}(k)} \frac{1}{|c|} \sum_{h \in c} \|k, h\|_2$
17: end function
18: function SILHOUETTESCORE($\mathscr{G}, a, b$)
19:     $\sigma_k = \frac{b_k - a_k}{\max(a_k, b_k)}$ if $|\mathscr{G}amma_\mathscr{G}(k)| > 1$, 0 otherwise, $\forall k \in \mathscr{C}$
20:     return $\frac{1}{K} \sum_{k \in \mathscr{C}} \sigma_k$
21: end function
```

To address these challenges, this work proposes client clustering based on their visual data styles, as detailed in Algorithm 7. Client styles are transferred to the server and used to partition the client set $\mathscr{C}$ into a set of non-empty clusters $\mathscr{G}$, denoting the different latent visual domains. The number of clusters is denoted by G. Each cluster c has a centroid μ_c computed using the average styles $\bar{s}_k$ of clients k within the cluster.

Style-based Aggregation

The standard FEDAVG algorithm typically aggregates model updates from selected clients at each round. This work proposes a clustered and layer-aware aggregation approach instead. Several key elements are defined:

- w_k^t: Weights of the model for client k after E local training epochs at round t.
- $\boldsymbol{\theta}_k^t$ and $\boldsymbol{\Phi}_k^t$: Cluster-specific and global parameters of the local model $\boldsymbol{w}_k^t$, respectively. Formally, $\boldsymbol{w}_k^t = \boldsymbol{\theta}_k^t \cup \boldsymbol{\Phi}_k^t$ and $\boldsymbol{\theta}_k^t \cap \boldsymbol{\Phi}_k^t = \emptyset$.
- $\mathscr{C}^t \subseteq \mathscr{C}$: Subset of clients selected at round t.

The server performs a global aggregation of the global parameters $\boldsymbol{\Phi}_k^t$ across all selected clients in $\mathscr{C}^t$ to obtain the updated global parameter set $\boldsymbol{\Phi}^{t+1}$. In contrast, the cluster-specific parameters $\boldsymbol{\theta}_k^t$ are averaged within their respective clusters, resulting in G sets of specific parameters $\boldsymbol{\theta}_c^{t+1}$ and G corresponding models $f_c^{t+1}(\boldsymbol{x};\boldsymbol{w}_c^{t+1})$ for $c \in \mathscr{G}$, where $\boldsymbol{w}_c^{t+1} = \boldsymbol{\Phi}^{t+1} \cup \boldsymbol{\theta}_c^{t+1}$. It is important to note that the server does not need to store independent models for each cluster. It can efficiently manage this process by keeping only the cluster-specific parameters $\boldsymbol{\theta}_c^{t+1}$ and the global parameters $\boldsymbol{\Phi}^{t+1}$, loading them as needed.

At test time, given the i-th target test image, the following steps are performed:

1. Extract the style $s_{\text{test},i}$.
2. Compute the ℓ_2-norm between $s_{\text{test},i}$ and all the cluster centroids μ_c for $c \in \mathscr{G}$.
3. Identify the cluster c with the smallest ℓ_2-norm.
4. Use the corresponding model $f_c^{t+1}(\boldsymbol{x};\boldsymbol{w}_c^{t+1})$ to evaluate the test image.

By leveraging client clustering and a specific aggregation strategy, this approach aims to improve the performance of federated semantic segmentation in the presence of heterogeneous data distributions and limited client datasets.

Client-Side Unsupervised Training

Self-Supervised Training. At round t, given an image $\boldsymbol{x}$ on the k-th client, the local model $f_k^t(\boldsymbol{x};\boldsymbol{w}_k^t)$ is trained by employing hard one-hot pseudo-labels $\tilde{y}(\boldsymbol{x}) \in \mathbb{R}^{N_C \times N_p}$ with the same threshold mechanism proposed in [432], where N_C is the number of classes. To minimize the computational load on clients, this approach avoids employing client-specific teacher networks. Instead, pseudo-labels are generated using a cluster-specific teacher network, denoted as $g_c^t(\boldsymbol{x};\boldsymbol{w}_{g_c}^t)$. This network outputs predictions represented by $\hat{y}(\boldsymbol{x}) := g_c^t(\boldsymbol{x};\boldsymbol{w}_{g_c})$. The teacher network parameters,

Algorithm 8 LADD (Learning Across Domains and Devices)

```
Require: Source (labeled) dataset $\mathscr{D}^S$, clients $k \in \mathscr{C}$ with target (unlabeled) datasets
    $\mathscr{D}_k^T$, global model $f(\boldsymbol{w}) = f(\{\theta, \phi\})$

 1: procedure CLUSTERING CLIENTS IN $\mathscr{C}$ AND PRE-TRAINING OF $f$ ON $\mathscr{D}^S$
 2:     Extract the styles $\mathscr{P}_k^s$ for each $k \in \mathscr{C}$
 3:     Define the style-based clusters from $\mathscr{H}$
 4:     Train $f(\boldsymbol{w})$ on $\mathscr{D}^S$ with style-transfer from $\mathscr{P}^s = \bigcup_{k \in \mathscr{C}} \mathscr{P}_k^s$
 5: end procedure

 6: procedure ADAPTATION OF $f$ ON $\mathscr{D}^T$
 7:     Initialize cluster models $f_c(\boldsymbol{w}_c) \leftarrow f(\boldsymbol{w})$ and teachers $g_c(\boldsymbol{w}_{g_c}) \leftarrow f(\boldsymbol{w})$
 8:     for each round $t \in [T]$ do
 9:         Randomly extract $\mathscr{C}^t \subset \mathscr{C}$
10:         Let $c := \Gamma_{\mathscr{H}}(k)$
11:         for all $k \in \mathscr{C}^t$ in parallel do
12:             Set $f_k(\boldsymbol{w}_k) = f_c(\boldsymbol{w}_c)$
13:             $\phi_k^t, \theta_k^t \leftarrow$ CLIENTUPDATE($f_k$, $g_c$, $f$, $\mathscr{D}_k^T$)
14:         end for
15:         $\phi^{t+1} \leftarrow$ Aggregate $\phi_k^t$ globally
16:         $\theta_c^{t+1} \leftarrow$ Aggregate $\theta_k^t$ within the cluster $c$
17:         if $t \mod \omega \equiv 0$ then
18:             if $t \geq t_{\text{START}}$ then
19:                 $g_c^{t+1}(\boldsymbol{w}_{g_c}) =$ SWATUPDATE($g_c^t$) $\forall c$
20:             else
21:                 $g_c(\boldsymbol{w}_{g_c}) = f_c^t(\boldsymbol{w}_c^t)\ \forall c \in \mathscr{G}$
22:             end if
23:         end if
24:     end for
25: end procedure
```

denoted by $\boldsymbol{w}_{g_c}^t$, are initialized as $\boldsymbol{w}_{g_c}^0 \leftarrow \boldsymbol{w}$ for all clusters $c \in \mathscr{G}$. These parameters are subsequently updated every ω rounds according to the rule $\boldsymbol{w}_{g_c}^t \leftarrow \boldsymbol{w}_c^t$, where $\boldsymbol{w}_c^t$ represents the model parameters of cluster c at round t.

Regularization. Pseudo-labels allow the clients to mimic the presence of the labels. However, after a few training iterations, the learning curve starts dropping [433]. The initial pre-training stage aims to bridge the gap between the knowledge extracted from the source dataset $\mathscr{D}^S$ and the knowledge required for success on the target datasets $\mathscr{D}_k^T$. However, as training progresses, the pre-trained model can become overconfident in its predictions, leading to decreased effectiveness and increased misclassifications. Therefore, it becomes crucial to mitigate this trend.

This work employs a Knowledge Distillation (KD) loss, denoted as $\mathscr{L}_{\text{KD}}$ [69], based on the soft predictions of the pre-trained model, to prevent the model, $f^t(\boldsymbol{w})$, from forgetting the knowledge acquired during pre-training. Experiments demonstrate that KD alone is insufficient to prevent overfitting, as the learning curve exhibits a slight decline in the later adaptation rounds.

Inspired by the recent success of Stochastic Weight Averaging (SWA) [237] in federated learning [7], a moving average is applied to the client-specific teacher networks, g_c, after a starting round t_{start}. This update rule is defined as:

$$w_{g_c}^{t+\omega} = \frac{(w_{g_c}^t n_{g_c}^t + w_c^{t+\omega})}{(n_{g_c}^t + 1)}, \tag{5.9}$$

where $n_{g_c}^t = \frac{(t-t_{\text{start}})}{\omega}$. This technique is referred to as *SWA teacher* (SWAt). SWAt reduces noise, further stabilizes the learning curve, and allows the model to better converge to the local minimum of the total loss function:

$$\mathscr{L} = \mathscr{L}_{\text{PSEUDO}} + \lambda_{\text{KD}} \mathscr{L}_{\text{KD}}, \tag{5.10}$$

where λ_{KD} is a hyperparameter controlling the impact of the KD loss.

5.3.3 LADD in Real-World Vision Scenarios

Experiments Setup

This work evaluates the proposed framework in a synthetic-to-real transfer learning setting for autonomous driving applications, a common benchmark for domain adaptation methods. For details on the mentioned datasets, please refer to Section 3.4.2, while additional information on training hyper-parameters is provided in Section A.5.

Source Domain. The source domain dataset is the synthetic Grand Theft Auto V (GTA5) dataset. It contains 24,966 highly realistic road scenes depicting typical US-like urban and suburban environments.

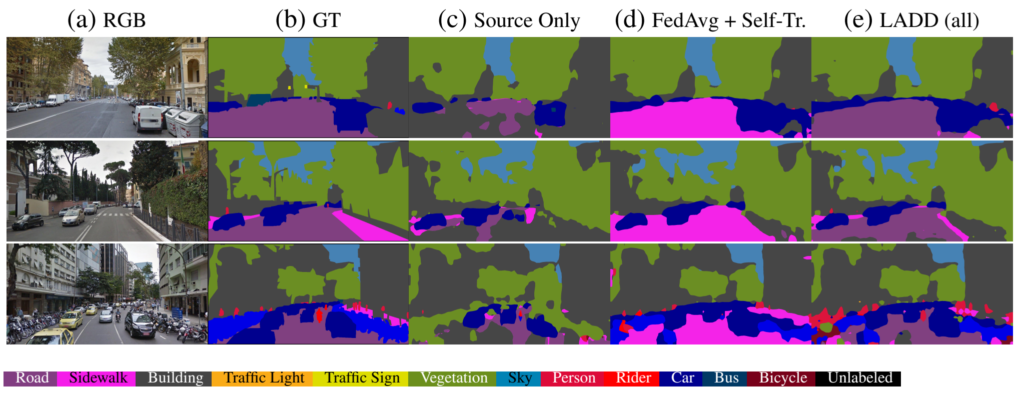


Fig. 5.15 GTA5→CrossCity qualitative results.

Target Domain. Three real-world (target) domain datasets are used in the experiments: Cityscapes, CrossCity, and Mapillary Vistas. Unlabeled training data is used from all target datasets. Results are reported on the original validation split for Cityscapes and Mapillary Vistas, and on the test split for CrossCity. The federated versions of the target datasets are described in Chapter 6.

Evaluation and Comparison Methods. To assess the effectiveness of the proposed approach in the unexplored domain adaptation setting, this work compares it with several established methods under both centralized and federated learning paradigms. The lower bound is the *source-only* approach, where only source-labeled data is used for model training. FTDA and *Oracle* establish upper bounds and assume access to supervised target data. In the centralized setting, target data is available on the server, while in the federated setting, it resides on the clients. FTDA utilizes target data for fine-tuning a pre-trained source-only model, while *Oracle* simply performs supervised FEDAVG training on the labeled target dataset. In addition, the Maximum Classifier Discrepancy (MCD) [434] method is adapted to the FL scenario following [297]. The state-of-the-art unsupervised domain adaptation (UDA) DAFormer [435] is included for comparison in Cityscapes experiments (note: evaluated in the simpler UDA setting where both source and target data are jointly available).

Experimental Results

This section introduces the results obtained with LADD on FFREEDA. Examples of qualitative results can be seen in Figure 5.15.

GTA5 $\rightarrow$ Cityscapes. The first setup involves the GTA5 to Cityscapes adaptation. Experimental results are presented in Table 5.6. Although the GTA5 dataset provides high-quality realistic images, it still suffers from a domain gap compared to real-world images like those in Cityscapes. Training on supervised source data alone (*i.e.*, *source only*) leads to a significant performance discrepancy compared to full target supervision. Even with the state-of-the-art DAFormer method in a UDA setting (*i.e.*, assuming joint availability of source supervised and aggregated target unsupervised data, which violates the setup assumptions), there is a noticeable performance drop of around 25% mIoU from the supervised oracle.

In this setup, a FL framework is assumed, with private target data distributed among multiple clients and a large-scale source dataset only available on a central server for pre-training. This introduces additional challenges not present in standard centralized domain adaptation settings. Specifically, source and target data are not accessible on the same device, and target data is available in small batches distributed across devices. Furthermore, target data is heterogeneously distributed among clients.

The increase in task complexity is evident from the performance drop of the supervised target oracle and FTDA methods (which still assume target supervision). This also applies to the MCD UDA approach, which loses almost 10% mIoU when tested in a federated setting.

The proposed method achieves robust results in this challenging setting, with an mIoU of around 36.5%. The efficient pre-training based on domain stylization, along with the self-training optimization scheme, addresses the lack of source data at the client side. Training stability, hindered by the small amount of target data available within individual clients, is improved with KD and SWAt, as indicated by the small standard deviation of results in Table 5.6.

Finally, an enhanced aggregation mechanism is provided, effectively sharing task information among clients with similar input statistics (*i.e.*, with smaller domain gaps), according to style-based client clustering. Observations in Table 5.6 show that LADD maintains a similar performance gap compared to the target oracle as DAFormer achieves in a centralized UDA setting. Competitive results are achieved with different variations of the proposed style-based clustering, such as keeping only the classifier (*i.e.*, LADD *(cls)*) or the whole network (*i.e.*, LADD *(all)*) as cluster-specific during aggregation.

Table 5.6 FFREEDA: Results on federated heterogeneous Cityscapes

Setting	Method	mIoU (%)
Centralized	Oracle	66.64 ± 0.33
	Source Only	24.05 ± 1.14
	FTDA	65.74 ± 0.48
Centralized	MCD	20.55 ± 2.66
	DAFormer	42.31 ± 0.20
Federated	Oracle	58.16 ± 1.02
	FTDA	59.35 ± 0.61
FL-UDA	MCD	10.86 ± 0.67
FFREEDA	FEDAVG † + Self-Training	35.10 ± 0.73
FFREEDA	**LADD (cls)**	$\mathbf{36.49 \pm 0.13}$
FFREEDA	**LADD (all)**	$\mathbf{36.49 \pm 0.14}$

†: Using same pre-trained model of LADD.

GTA5 $\rightarrow$ CrossCity. The performance of the proposed approach is further investigated in the GTA5$\rightarrow$CrossCity scenario. Quantitative results are reported in Table 5.7. A comparison is made with the naïve source only baseline, as well as with MCD. Due to the lack of target supervision on training images, the upper bound of the target oracle cannot be provided, nor the result of FTDA.

The diverse content and appearance of CrossCity's road scenes, due to the variable geographic origin of its samples, provide a heterogeneous target distribution. The enhanced heterogeneity with respect to the more uniform Cityscapes dataset in turn leads to a tougher challenge for federated training. For instance, the MCD method, when extended from a centralized to a federated learning framework, suffers from a substantial performance reduction. In contrast, LADD provides a much higher accuracy in a federated setting, with more than 17% gain over federated MCD, while also not requiring reuse of source data after the initial pre-training. This indicates the robustness of LADD with respect to the statistical diversity of client target data.

Finally, by allowing only a minimal amount of network parameters to be cluster-dependent, a final accuracy very close to the best result is achieved, obtained without any parameter sharing across clusters of clients. This result shows that LADD demands limited communication overhead with respect to standard FEDAVG.

GTA5 $\rightarrow$ Mapillary. The target data in this experiment originates from Mapillary Vistas, a geographically diverse dataset distributed among clients according to location. This distribution leads to even greater heterogeneity in client data compared

Table 5.7 FFREEDA: Results on federated CrossCity

Setting	**Method**	**mIoU (%)**
Centralized	Source Only	26.49 ± 1.46
	MCD	27.15 ± 0.87
FL-UDA	MCD	24.80 ± 1.56
FFREEDA	FEDAVG [†] + Self-Training	33.59 ± 1.25
FFREEDA	**LADD (cls)**	39.87 ± 0.14
FFREEDA	**LADD (all)**	**40.09 ± 0.19**

†: Using same pre-trained model of LADD.

Table 5.8 FFREEDA: Results on federated Mapillary

Setting	**Method**	**mIoU%**
Centralized	Oracle	61.46 ± 0.21
	Source Only	32.40 ± 0.71
	MCD	31.93 ± 1.89
Federated	Oracle	49.91 ± 0.49
FL-UDA	MCD	19.15 ± 0.75
FFREEDA	FedAvg[†] + Self-Training	38.97 ± 0.21
FFREEDA	**LADD (cls)**	**40.16 ± 1.02**
FFREEDA	**LADD (all)**	38.78 ± 1.82

†: Using same pre-trained model of LADD.

to previous setups. The performance gap between the supervised target oracle in centralized and federated settings is significant, with a decrease of approximately 11.5% mIoU observed (Table 5.8). This challenge is further amplified for the UDA approach using MCD, which suffers a similar mIoU decrease (12%) when adapted to the federated setting.

In contrast, the proposed method demonstrates robust performance. The best configuration of LADD, which keeps only classifier weights cluster-specific (LADD (cls)), surpasses the source-only baseline by over 8% mIoU and approaches the performance of the federated oracle. This performance improvement signifies the effectiveness of LADD in tackling domain adaptation within a distributed learning environment. Furthermore, LADD outperforms a simpler framework based on FEDAVG and self-training in its best configuration. This outcome supports the efficacy of the additional modules in LADD in addressing the domain adaptation challenge.

5.3.4 Conclusion

This work presented FFREEDA (Federated Source-Free Domain Adaptation), a new and challenging setting for source-free domain adaptation in federated learning for semantic segmentation tasks. FFREEDA utilizes a labeled dataset on a central server for pre-training, while local training on client devices leverages only unlabeled client data.

To address the challenges of FFREEDA, a novel algorithm named LADD (Learning Across Domains and Devices) was proposed. LADD incorporates various techniques, including style transfer, knowledge distillation, SWA teacher, and style-driven clustering. This approach enables effective learning of both global and personalized parameters. The achieved results demonstrate that LADD is competitive with state-of-the-art methods in related tasks.

5.4 Cluster-driven Graph Federated Learning over Multiple Domains

5.4.1 Motivation

The clustering of clients, as detailed in Sections 5.2 and 5.3, presents a major drawback: clustering may reduce heterogeneity by identifying the domains, but it deprives each cluster model of the data and supervision of others. This work proposes an alternative solution, namely FEDCG (Cluster-driven Graph Federated Learning), based on Graph Convolutional Neural Networks (GCNs). In FEDCG, clustering serves to address statistical heterogeneity, and GCNs enable sharing knowledge across clients. FEDCG: **i.** identifies the domains via an FL-compliant clustering and instantiates domain-specific modules (residual branches) for each domain; **ii.** connects the domain-specific modules through a GCN at training time to learn the interactions among domains and share knowledge; and **iii.** learns to cluster clients according to their domain of belonging in an unsupervised way, via teacher-student classifier-training iterations, and to address novel unseen test domains via their domain soft-assignment scores.

5.4.2 Cluster-driven Graph Federated Learning

FEDCG leverages clustering and its potential to reduce statistical heterogeneity by identifying homogeneous[1]. Concurrently, FEDCG is the first to model the domain-domain interaction by means of a GCN, which connects domain-specific model components. In the GCN, each node consists of domain-specific model parameters, while the adjacency matrix is composed of the inverse pairwise distances between the domain-specific parameters. In this way, FEDCG not only captures the specificity

[1]Homogeneous stands in this context for groupings that minimize intra-cluster *vs*. inter-cluster variance.

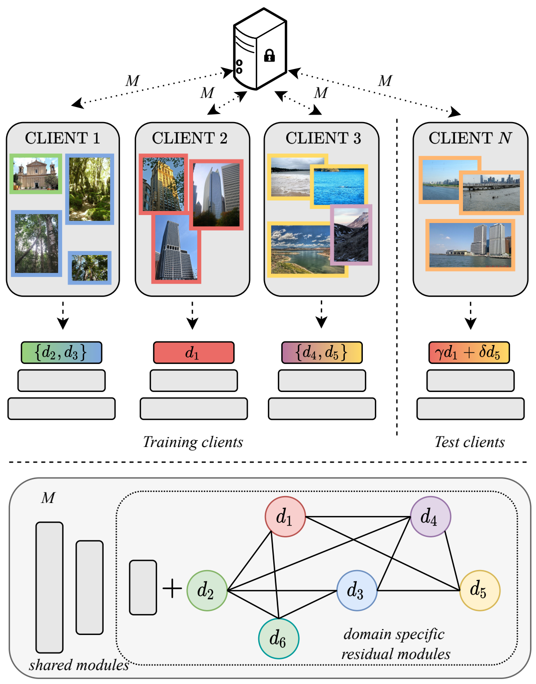


Fig. 5.16 In a federated scenario, clients and server exchange the parameters of the model M. Each client has access to its local data, which can be non-*i.i.d.* and unbalanced. Each color identifies a different distribution, *i.e.*, a domain, such as pictures of skyscrapers or sea landscapes. The model M is made of domain-agnostic layers (in gray) and a GCN containing domain- specific parameters, added as residual. According to the domains of the input images, the corresponding nodes of the GCN are activated. At test time, new domains can be addressed as a soft combination of the discovered ones, *e.g.*, skyscrapers over the sea.

of each domain but also allows each domain to benefit from the updates of others, sharing knowledge at training (Figure 5.16).

Clustering is based on unsupervised teacher-student [69] classifier-training iterations and it generalizes to unseen test-time domains. The clustering technique is based on pseudo-labels, assigned by a teacher and learned by a student, in rounds of refinements. This is accomplished within the FL training paradigm, respecting the client's privacy, and allows to estimate soft-assignments for unseen novel test domains.

Federated Clustering

Statistical heterogeneity, arising from domain variations within the data, can be mitigated by employing domain-specific modules. However, identifying these different domains presents a challenge in federated learning settings, mainly due to data partitioning (client-partitioned data hinders direct server-side clustering) and non-stationarity (optimal clusters for the training data may not generalize to the test set).

This work addresses the first challenge by proposing a clustering procedure based on two domain classifiers: a *teacher* and a *student*. These classifiers iteratively group clients' images such that the resulting groupings are easier to classify. This approach aims to overcome the limitations imposed by data distribution across clients.

Formally, assuming the data contain D domains, with D being a hyperparameter, two domain classifiers can be initialized, the teacher g_ϕ and the student g_φ parametrized by ϕ and φ respectively. Each domain-classifier is a function, mapping images to a probability vector $\mathscr{Q}$ defined over the D domains, *i.e.* $g_\cdot : \mathscr{X} \to \mathscr{Q}$. Given an input image, the teacher provides domain pseudo-labels as a target to refine student's predictions. In particular, the client student parameters φ_k is trained by iteratively minimizing the cross-entropy loss between the teacher and student domain predictions over $\mathscr{D}_k$. Thus, for a client $k \in K$, the parameters φ_k of the student are:

$$\varphi_k = \arg\min_{\varphi} -\frac{1}{n_k} \sum_{(x,y)\in\mathscr{D}_k} \log g_{\varphi}^{\hat{d}}(x)\,, \tag{5.11}$$

where $\hat{d}$ is the pseudo-label given by the teacher for x, *i.e.* $\hat{d} = \arg\max_{d\in D} g_\phi^d(x)$ and $g_*^d(x)$ denotes the probability of x to belong to the d-th domain as given by g_*. Equation (5.11) rewards the student from being able to classify according to the pseudo labels, and implicitly encourages agreement on the pseudo-labels, thus on the clustering, which most easily may be agreed upon. Then the domain classifier parameters φ are updated after each round with standard FEDAVG, *i.e.* $\varphi = \frac{1}{\sum_{i=1}^{K^t} N_k} \sum_{k\in\mathscr{C}^t} N_k \varphi_k$.

The idea behind this approach is inspired from deep clustering with self-labelling [436], *i.e.*, the teacher and the student networks would find the equilibrium once they group images in such a way so they can be more easily recognized. This reconnects to the DNNs being natural deep image priors [437], working well for image-related tasks even if just randomly initialized. And it may intuitively match that a "bad" labelling would leave no alternative to a DNN but to overfit [32], which may be hard to imitate by the student. Differently from [436], since there is no access to data and cluster labels, the teacher g_ϕ is used to provide them locally in each client. Both ϕ and φ are randomly initialized and ϕ is fixed during training. After T rounds, with T being a hyperparameter, the parameters ϕ of the teacher are updated with the current student ones φ, iteratively.

Note that, unlike previous works [438], this clustering algorithm can assign unseen data to clusters at test time, thanks to the domain classifier. In particular, the cluster assignment of a test image $\boldsymbol{x}$ corresponds to the domain probabilities given by the student g_φ. Since $g_\varphi(\boldsymbol{x})$ is soft, data belonging to unseen domains can be addressed as a combination of existing ones. Additionally, in this formulation, one client's data samples may belong to multiple clusters, considering the more general case where each client may contain more than one data distribution.

Cluster-specific Models

Since the model can identify data clusters through the previously described procedure, the function $f_{\boldsymbol{w}}$ can be specialized to each domain. Inspired by multi-domain learning [439–443], this can be achieved with domain-specific components. For simplicity, the parameters $\boldsymbol{w}$ can be split into two sets, *i.e.* $\boldsymbol{w} = \{\boldsymbol{w}_a, \boldsymbol{w}_s\}$ where $\boldsymbol{w}_a$ are the domain-agnostic parameters and $\boldsymbol{w}_s$ the domain-specific ones. Note that $\boldsymbol{w}_s$ is actually a set $\boldsymbol{w}_s = \{\boldsymbol{w}_s^d\}_{d=1}^D$ where $\boldsymbol{w}_s^d$ are the parameters specific to the d-th domain. To tailor the model to a specific domain, multiple ways exist to include $\boldsymbol{w}_s$, such as direct influence on the agnostic parameters $\boldsymbol{w}_a$ [440, 442, 443] or residual activations [439, 441]. This works follows the latter strategy, since the former relies on the robustness of $\boldsymbol{w}_a$, which is harder to guarantee in FL. Let us assume $f_{\boldsymbol{w}}$ to be a deep neural network with a set of layers L, denoting as $f_{\boldsymbol{w}}^\ell$ the function applied at layer $\ell \in L$. Given input from a domain and the features z^ℓ extracted at the previous layer, the output of the ℓ-th layer is:

$$z^\ell = f_{\boldsymbol{w}_a}^\ell(z) + \lambda_l \sum_{d=1}^{D} w_d \cdot f_{\boldsymbol{w}_s^d}^\ell(z)\,, \tag{5.12}$$

where λ_l is a learnable parameter balancing the effect of the domain-specific components and w_d is the weight of domain d. During training, the assumption is that data belongs to a single cluster, given by the pseudo-labels of the teacher, thus w_d is 1 if $d = \hat{d}$ and 0 otherwise. At test time, the aim is to having learned a model capable of dealing with data from arbitrary domains by simply combining residuals of seen ones. Thus $w_d = g_\varphi^d(x)$, weighting the impact of each domain-specific component by the student output probabilities. Note that the formulation in Equation (5.12) is general, with $f_{\boldsymbol{w}}^\ell$ being any layer of a standard convolutional neural network, and can be applied to the whole network or just some layers (the last layers in this case).

In FL, the central domain-specific parameters must be updated without access to local data and after each round. In practice, following Equation (3.5), FEDAVG is applied on both domain-agnostic and domain-specific parameters in each training round.

Knowledge Sharing between Cluster-specific Models

The goal is to learn a model that can adapt to the specificity of each domain. Here it is proposed to refine the domain-specific parameters by making them interact. Specifically, the interaction of the domain-specific parameters of each layer ℓ are modeled via a graph $\mathscr{G}^\ell = (\mathscr{V}^\ell, \mathscr{E}^\ell)$, where the nodes $i \in \mathscr{V}^\ell$ are the set of all domain-specific parameters at layer ℓ, and $e_{ij} \in \mathscr{E}^\ell$ are the edges connecting two domain nodes i and j which may interact together. This implies that if a domain has few assigned samples, its parameters will be rarely updated and thus not robust enough to capture the specificity of the domain and generalize to unseen samples of the same domain.

In particular, this work proposes to use a GCN [118] to model the interaction of domain-specific parameters. The matrix $\mathbf{V}^\ell$ collects the value of each node, *i.e.* all domain-specific parameters at layer ℓ: $\mathbf{V}^\ell = [\boldsymbol{w}_{s|l}^1, \ldots, \boldsymbol{w}_{s|l}^D]^\intercal \in^{D\times q}$, with $q = |\boldsymbol{w}_{s|l}^d|$ the number of parameters per domain. The graph-version $\hat{\mathbf{V}}^\ell$ of the domain-specific parameters $\mathbf{V}^\ell$ is computed as:

$$\hat{\mathbf{V}}^\ell = \sigma(\mathbf{A}\,\mathbf{V}^\ell\,\mathbf{W}^\ell)\,, \tag{5.13}$$

where σ is an activation function (*e.g.* ReLU), $\mathbf{A} \in^{D\times D}$ is the adjacency matrix defined across the domains, and $\mathbf{W}^\ell \in^{q\times q'}$ is a weight projection matrix, projecting the domain-specific parameters into dimension q'. Here, for simplicity, $q = q'$. In FEDCG, the domain-specific parameters of Equation (5.12) are substituted with the ones computed in Equation (5.13). Similarly to all other parameters of the network, $\mathbf{W}$ is updated at every training round with FEDAVG. In case q is large, $\mathbf{W}$ can be implemented as a multi-layer bottleneck.

The values in the adjacency matrix encode, for each edge, how close two domains are; since there are no priors on the structure of the graph, $\mathscr{G}^\ell$ is modeled as a fully-connected weighted graph. Without direct access to the data server-side, the distance among two domains is computed directly in the (domain-specific) parameter's space. In practice, the similarity $h_{i,j}$ among domains i, j is

$$h_{i,j} = \frac{1}{||\boldsymbol{w}_s^i - \boldsymbol{w}_s^j||_2}, \tag{5.14}$$

and the corresponding value $\mathbf{A}_{ij}$ in the adjacency matrix becomes

$$\mathbf{A}_{ij} = \begin{cases} \beta & \text{if } i = j \\ \frac{(1-\beta)\cdot h_{ij}}{\sum_{d=1}^{D} \mathbb{1}_{i \neq d} h_{id}} & \text{otherwise} \end{cases}$$

where β is a hyper-parameter weighing the impact of the self-connection, set to 0.5, and $\mathbb{1}_{i \neq m}$ is an indicator function being 1 when $i \neq m$ and 0 otherwise.

In this formulation, each client receives not only the set of parameters $\boldsymbol{w}$, but also the adjacency matrix. With this definition, the gradient of a domain-specific component is forced to flow to all others through the GCN. Consequently, an update on a domain-specific component will influence all domain-specific parameters, even the ones of the domains not present in the current training round. Moreover, given two domains i, j with $i \neq j$, the influence of j on i in each layer is directly proportional to the adjacency matrix value $\mathbf{A}_{ij}$. This means that the more two sets of domain-specific parameters are close, the higher is their mutual influence. Finally, while the GCN is a way to ensure information flow across domains during training, at inference one can just precompute $\hat{\mathbf{V}}^\ell$ for each layer, to save memory usage.

5.4.3 Experiments Results

FEDCG is evaluated on the CelebA and FEMNIST image classification datasets, proposed in [394].

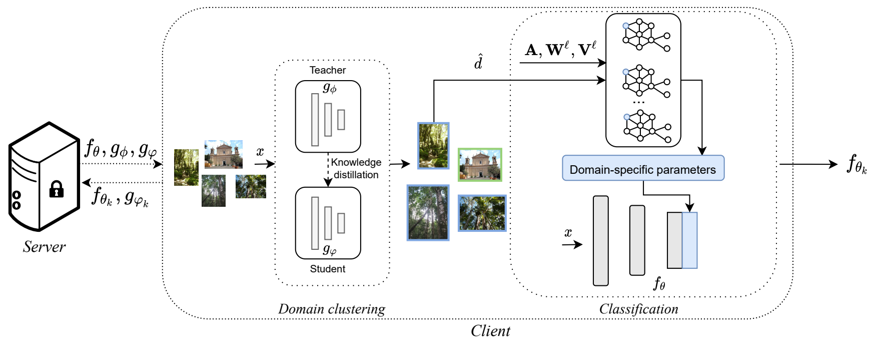


Fig. 5.17 FEDCG framework. The server sends the model f_θ to the clients selected for the federated round, together with the teacher g_ϕ and student g_φ domain classifiers. On the client-side, the domain classifier clusters the local data $\boldsymbol{x}$, producing as output the domain of belonging $\hat{d}$ of each image. At training time, the hard label $\hat{d}$ is predicted by g_ϕ and is used as input to train g_φ through a process based on knowledge distillation. At test time, $\hat{d}$ is given by g_φ and is a weighted combination of the discovered domains. In FEDCG, the network f_θ is made of a domain-agnostic part (in gray) and a residual domain-specific one (in blue). The domain-specific parameters are produced by the GCN, receiving as input $\mathbf{A}, \mathbf{W}^\ell, \mathbf{V}^\ell$ and $\hat{d}$. After training both f_θ and g_φ on its data, the client k sends back to the server the updated weights θ_k and φ_k. On the server-side, the updates are aggregated with FEDAVG.

Using Domain Information

Oracle Domain Knowledge. To establish a baseline for model evaluation, domain labels are initially defined manually by leveraging prior knowledge from the dataset's image metadata (40 attributes, Table 5.9). This approach isolates the impact of incorporating domain-specific information within the model, independent of the clustering procedure. The most balanced data subdivisions with low target feature correlation were identified from the 40 attributes, resulting in $n = 5$ selected attributes and $N = 32$ total domains: “attractive”, “heavy makeup”, “high cheekbones”, “mouth slightly open” and “wavy hair”.

Domain-Specific Models. The initial approach involved replacing the standard single server model with N separate models, each trained and tested solely on images from its assigned domain. As shown in Table 5.9, this strategy leads to a significant performance drop (33.61% vs. 86.88% FEDAVG) due to insufficient data per model, hindering generalization. This demonstrates the ineffectiveness of training a separate full model for each domain in this scenario.

Table 5.9 Ablation studies on CelebA dataset with $N = 32$ domains extracted from images meta-data. **A** is the adjacency matrix that weights the domains contributions: the symbols (eye,U,H) respectively stand for identity, uniform and weighted (with inverse Hamming distance) matrices. **W** is the weight projection matrix and ReLU the non-linear activation.

Model	**A**	**W**	**ReLU**	**Acc(%)**
Domain-specific models	-	-	-	33.61
GCN	U	✗	✗	84.39
	H	✗	✗	82.10
	U	✓	✗	87.92
	H	✓	✗	84.25
FEDCG	U	✓	✗	86.96
	H	✓	✗	88.65
	U	✓	✓	87.97
	H	✓	✓	89.57

Modeling Domain Relationships. To account for relationships between domains, a 1-layer graph convolutional network is introduced as a graph model. The impact of the GCN is evaluated by testing a simpler model variant without the weight transformation matrix **W**. Different choices for the adjacency matrix **A** were explored, including uniform weighting (U), domain weighting based on similarity (H), specifically the normalized inverse of the Hamming distance between domain numerical representations (binary metadata). Table 5.9 shows that using a GCN consistently improves performance compared to domain-specific models. While the uniform adjacency matrix performs slightly better than the weighted one in this case, both benefit from the inclusion of the projection matrix **W**. These results highlight the importance of interaction between domain-specific nodes. However, they remain unsatisfactory, falling below or just above FEDAVG performance (GCN-H with **W**). This suggests domain information is still not fully exploited within the model.

Residual Domain-Specific Layers. The use of domain-specific parameters to generate residual activations (Equation (5.12)) as implemented in FEDCG is investigated, compared to the GCN without a domain-agnostic component. Table 5.9 shows that while the model with a uniform adjacency matrix (U) experiences a performance decrease from GCN to FEDCG (87.92% vs. 86.96%), the model with a weighted adjacency matrix (H) exhibits a significant increase, from 84.25% accuracy with GCN to 88.65% with FEDCG. Two conclusions can be drawn: i) Utilizing residual layers to refine domain-agnostic activations (FEDCG) outperforms relying solely on domain-specific components (GCN). ii) When integrated as residuals, domain-specific components are more effective with weighted (H) rather than uniform (U)

Table 5.10 **Ablation studies on CelebA dataset with domains given by a priori knowledge or online clustering procedures**. In the **A** init column, "eye" stands for identity matrix and "rand" for random. The third column specifies the clustering, *i.e.* clusters generated with K-means or the teacher-student classifier ("Clf").

FedCG layers	**A** Init	Clusters	D	Soft domains	Acc(%)
All	Eye	K-means	2	✗	88.36
			3	✗	87.97
			4	✗	87.21
	Eye	Clf	2	✗	88.03
			3	✗	88.59
			4	✗	88.74
	Rand	Clf	2	✗	88.73
			3	✗	88.24
			4	✗	88.55
	Eye	Clf	2	✓	87.88
			3	✓	88.74
			4	✓	88.67
Last	Eye	Clf	2	✓	88.31
			3	✓	88.13
			4	✓	**89.18**
			32	✓	88.40

connections. This is evident from FedCG-H surpassing FedCG-U by 1.7% in accuracy. The importance of the ReLU non-linearity applied to the residual GCN output was also examined. The non-linearity improves FedCG for both uniform (+1%) and weighted (+0.9%) domain connections. The final FedCG model with a 1-layer GCN filtered by a ReLU and a weighted adjacency matrix outperforms the baseline FedAvg by 2.6 points in accuracy, demonstrating the effectiveness of the design choices.

Domain Identification

The previous section explored domain-specific component integration with oracle domain information. Here, the assumption of such knowledge is dropped, and the effectiveness of domains discovered through the clustering procedure on FedCG is investigated. These domains are identified using the teacher-student domain classifier. Results are presented in Table 5.10.

Clustered Domains. The performance of the proposed domain classifier is compared to K-means clustering applied to the parameters of models trained independently on each client. FedCG performs localized clustering, accessing only a subset of clients during each round. As shown in Table 5.10, the clustering procedure achieves performance that is either on par ($D = 2$) or superior ($D = 3,4$) to K-means clustering. Notably, as the number of clusters increases, the performance of K-means decreases (from 88.36% with $D = 2$ to 87.21 with $D = 4$), while FEDCG with the same residual GCN exhibits performance improvements (from 88.03% with $D = 2$ to 88.74 with $D = 4$). This indicates the effectiveness of the localized clustering procedure, which captures the presence of different domains within each client without requiring a dedicated model per client, unlike K-means.

Adjacency Matrix Initialization. The impact of different initialization strategies for the GCN's adjacency matrix $\mathbf{A}$ is now investigated. Two choices are considered: i) disconnected domains (identity matrix, *Eye*), ii) randomly connected domains (random adjacency matrix, *Rand*). Table 5.10 demonstrates that FEDCG's performance is not dependent on the specific initialization strategy, achieving over 88.5% for all choices with $D = 4$. However, using random initialization, the performance does not improve with an increasing number of domains. This suggests the importance of careful initialization of A as the number of domains grows. Consequently, a uniform initialization strategy is employed in subsequent experiments. This approach allows the model to refine domain-specific components independently before merging them based on their distance (see Section 5.4.2).

Soft *vs*. Hard Domain Assignment. The effect of using a soft combination of domains at test time is compared to a hard assignment derived from the domain classifier predictions. In both cases, the performances are close for all D, indicating that the domain classifier provides reliable domain predictions at test time. The soft assignment approach is used in subsequent experiments due to its higher flexibility.

Domain-Specific Layers on Last Layer Only. Finally, the application of domain-specific modules only on the last layer of the network (rather than all layers) is investigated to determine if good performance could be achieved while reducing the number of parameters required by FEDCG. As shown in the experiments, using domain-specific parameters on the last layer provides the best overall results (89.18% with $D = 4$), improving the best combination by 0.44%. Since this choice allows

Table 5.11 FEDCG comparison with the state of the art on CelebA and FEMNIST

Dataset	Model	Accuracy (%)
CelebA	FEDAVG	86.88
	FEDCG	**89.18**
FEMNIST	FEDAVG	77.81
	FEDCG	**83.41**
	FEDPROX	75.00
	SCAFFOLD	84.20

FEDCG to utilize fewer parameters while maintaining good performance, the use of domain-specific parameters will be limited to the last layers in the next section.

Comparison with the State of the Art

Table 5.11 compares FEDCG with FEDPROX, FEDAVG and SCAFFOLD on both FEMNIST and CelebA. FEDCG uses the domain-specific parameters applied on the last layer, $D = 4$, soft domain assignments at test time and the adjacency matrix initialized as identity.

FEDCG largely outperforms FEDAVG in both scenarios, achieving 89.18% accuracy compared to 86.88% of FEDAVG on CelebA, and 83.41% accuracy compared to 77.81% of FEDAVG on FEMNIST. This latter improvement (+5.6%) is remarkable given the higher complexity of the classification task in FEMNIST. Comparing FEDCG with FEDPROX and SCAFFOLD on FEMNIST, FEDCG outperforms FEDPROX by a large margin (+8.41%) while being slightly inferior to SCAFFOLD (*i.e.*-0.79%).

5.4.4 Conclusions

This work introduced FEDCG, the first cluster- driven approach addressing statistical heterogeneity in federated learning with Graph Convolutional Neural Networks. FEDCG uses an iterative clustering algorithm based on teacher and student domain classifiers. This clustering procedure serves to discover different input distributions, *i.e.*, domains, and to instantiate domain-specific parameters accordingly. The domain-specific parameters are connected through a GCN that enables their interaction and knowledge sharing during training. These parameters influence the activation of the main, domain-agnostic, network due to weighted residual activations. Thanks to the domain classifiers and connections of the GCN, new input distributions and unseen users can be addressed at test time via their do- main soft-assignment scores. Experimental results showed that FEDCG outperforms the FEDAVG on multiple benchmarks, demonstrating the efficacy of each component.

5.5 Summary

This chapter discussed the use of clustering-based techniques to group similar (Sections 5.3 and 5.4) or dissimilar clients (Section 5.2).

The main findings are the following:

- The local data distribution can be estimated without breaking the privacy constraints.
- Grouping together clients with dissimilar distributions, aiming to build homogeneous overall groups, helps speeding up convergence and reaching the centralized performance.
- In the vision domain, the style is a privacy-preserving information and can be used to model the local distributions.
- The model obtained with SWA can be used to distill knowledge into a less advanced student model in unsupervised scenarios.
- Graph Neural Networks can be used to model relationships between similar clients. This allows for knowledge sharing and fosters a degree of personalization within the federated learning framework.

Chapter 6

Novel Benchmarks for Federated Computer Vision

The computer was born to solve problems that did not exist before

BILL GATES

This chapter introduces new benchmarks designed to facilitate research on computer vision tasks within federated learning settings, aiming to support advancements in this field. Section 6.2 explores the specific requirements of federated learning for autonomous driving, particularly focusing on the task of semantic segmentation. Section 3.4.3 presents an adaptation of the Visual Place Recognition task for the federated learning environment.

6.1 Introduction

Federated learning has emerged as a powerful paradigm for training machine learning models on distributed datasets residing on various devices. This approach offers significant advantages in privacy preservation, as data remains on user devices, and communication overhead is minimized. However, applying FL effectively to computer vision (CV) tasks presents unique challenges.

The lack of standardized benchmarks specifically designed for federated CV hinders progress in this field. Traditional CV benchmarks often assume centralized training with access to vast, curated datasets. However, in federated settings, data is fragmented across devices, potentially exhibiting significant heterogeneity in distributions, labels, and image quality. This heterogeneity can lead to performance degradation in models trained via federated learning.

Developing robust and comprehensive benchmarks for federated CV is critical for several reasons:

- **Evaluating Model Performance:** standardized benchmarks provide a common ground for comparing the effectiveness of different FL algorithms on specific CV tasks. This facilitates objective evaluation and fosters innovation in the design of FL approaches for computer vision.
- **Understanding Challenges:** benchmarks can help us identify and quantify the specific challenges encountered when applying FL to CV tasks. These insights can guide research efforts towards addressing issues like data heterogeneity, privacy concerns, and communication efficiency.
- **Facilitating Algorithm Development:** By providing realistic and well-defined federated learning scenarios, benchmarks enable researchers to develop and test novel algorithms specifically tailored for the complexities of federated CV.

In conclusion, establishing well-designed benchmarks for federated CV is crucial for advancing research and development in this crucial field. These benchmarks will serve as a cornerstone for evaluating model performance, understanding the unique challenges of federated CV, and ultimately, enabling the development of robust and efficient federated learning algorithms for computer vision tasks.

This thesis introduces three novel benchmarks designed to support research in federated CV. These benchmarks address distinct challenges within this domain:

Federated Semantic Segmentation for Autonomous Driving : autonomous vehicles will likely collect sensitive data, necessitating privacy-preserving approaches like FL for semantic segmentation tasks. FEDDRIVE highlights the need for benchmarks and algorithms specifically designed for FL-based semantic segmentation in the context of autonomous driving.

Federated Learning with Unlabeled Data for Semantic Segmentation : building on FEDDRIVE, FFREEDA introduces benchmarks and methods that leverage unlabeled data on client devices during federated learning for semantic segmentation. This approach aligns with the more realistic scenario where client data may contain a significant amount of unlabeled information.

Visual Place Recognition : edge computing plays a crucial role in applications where vehicles need to recognize their location. FedVPR adapts the Visual Place Recognition (VPR) task to the FL setting and introduces a corresponding benchmark.

The works discussed in this section are published in the following venues:

- Fantauzzo*, L., Fanì*, E., Caldarola, D., Tavera, A., Cermelli, F., Ciccone, M., & Caputo, B. (2022).
  FEDDRIVE: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving.
  In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 11504-11511). IEEE. (**IROS 2022**)

- Shenaj*, D., Fanì*, E., Toldo, M., Caldarola, D., Tavera, A., Michieli, U., Ciccone, M., Zanuttigh, P., & Caputo, B. (2023).
  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* (pp. 444-454). (**WACV 2023**)

- Dutto, M., Berton, G., Caldarola, D., Fanì, E., Trivigno, G., & Masone, C. (2024).

Collaborative Visual Place Recognition through Federated Learning.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop on Federated Learning for Computer Vision*. (**CVPRW 2024**)

6.2 Federated Semantic Segmentation for Autonomous Driving

This section introduces the novel benchmarks proposed for solving semantic segmentation tasks for autonomous driving in federated scenarios.

6.2.1 Motivation

Research in autonomous driving aims at improving our safety and driving experience. In order to derive and execute trustworthy actions autonomously, vehicles must be able to perceive and understand their surroundings [444–447]. To ensure the safety of the passengers, an autonomous vehicle needs to determine precisely whether there is a danger such as an obstruction or a pedestrian, and consequently decide whether to slow down or accelerate. To do so, vehicles rely on the semantic segmentation task [448–451], whose goal is to provide a semantic prediction for every pixel of the image, giving a deep understanding of the surrounding environment. However, training robust semantic segmentation models requires having access to large scale datasets possibly representing all the conditions that can be encountered in the real world. One possible solution is to collect images from the customers' vehicles which, being already on the road, face different situations and cover multiple geographic locations, weather conditions, viewpoints, etc. While collecting the images from vehicles is an effortless process, sending them to a central server to train a model could potentially violate the users privacy. Indeed, other nodes of the communication infrastructure may access personal and privacy-protected information, thus violating the regulations in force [452–454]. A clever solution to train a model using all the clients' data while protecting their privacy is introduced by federated learning [34]. However, FL is a relatively new field and current methods mainly focus on simple vision tasks. Hence, the usage of FL for Semantic Segmentation in realistic urban environments opens the door to issues relating to the variety of witnessed scenes: images captured outdoors suffer from enormous variability in light, reflections, points of view, atmospheric conditions and types of settings, resulting in various *domains* (Figure 6.1).

Aiming to fuel the FL research on complex vision scenarios, this thesis introduces two benchmarks for SS in FL: FEDDRIVE [271] proposes the federated adaptations of

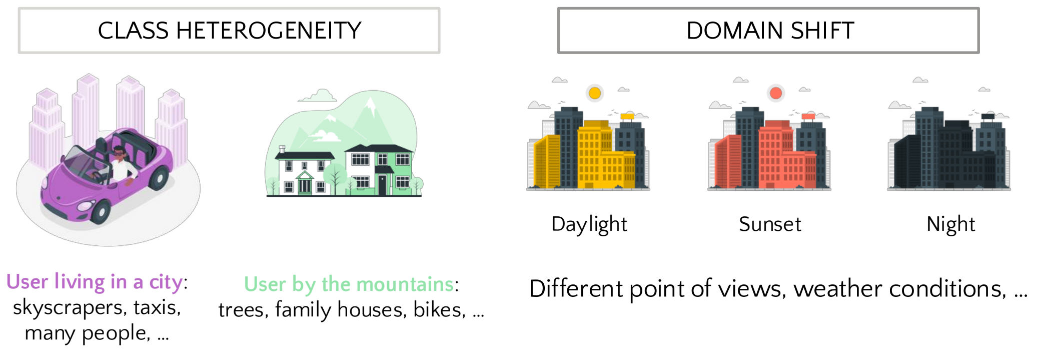

CLASS HETEROGENEITY
DOMAIN SHIFT
Daylight
Sunset
Night
User living in a city: skyscrapers, taxis, many people, ...
User by the mountains: trees, family houses, bikes, ...
Different point of views, weather conditions, ...

Fig. 6.1 Examples of distribution and domain shifts in autonomous driving scenarios. Due to personal habits and geographical locations, users may be subject to different scenery (*e.g.*, mountain landscapes *vs*. city views), or shifts in visual domains (*e.g.*, light conditions).

the Cityscapes and IDDA datasets, while studying the impact of batch normalization layers on generalization in such scenarios; FFREEDA [366] instead takes a step further in the application in real-world scenarios, removing the assumption on labeled client-side data. The considered datasets are CrossCity, GTA5, and Mapillary Vistas datasets.

6.2.2 FedDrive

© 2022 IEEE. Reprinted, with permission, from Fantauzzo, L., Fanì, E., Caldarola, D., Tavera, A., Cermelli, F., Ciccone, M., & Caputo, B. (2022, October). FEDDRIVE*: Generalizing federated learning to semantic segmentation in autonomous driving. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 11504-11511). IEEE.*

FEDDRIVE is the first benchmark introducing semantic segmentation for autonomous driving in federated scenarios, with a focus on statistical heterogeneity across clients.

Each client (vehicle) usually observes part of a meta-distribution or different factors of variation (geographical location, viewpoint, weather, illumination), which are generally different from other clients. Thus, this benchmark focuses on two key elements: i) high statistical heterogeneity, and ii) domain generalization. To address these issues, FEDDRIVE benchmarks state-of-the-art FL algorithms FEDAVG [34], SILOBN [405], FEDBN [193], and different server optimizers [290, 275]. To deal with the domain shift, the mentioned algorithms are combined with style-transfer methods [293, 455] to obtain a model capable of effectively generalizing on unseen domains.

Federated Datasets Adaptation

To investigate the different challenges of Federated Learning in autonomous driving, this work introduces a novel benchmark based on Cityscapes [396], and IDDA [174] datasets, proposing multiple federated versions for each of them, based on different levels of data heterogeneity between clients. Table 6.1 summarizes the multiple experimental settings emulating realistic scenarios.

Federated Cityscapes. Cityscapes is one of the most popular datasets for semantic segmentation and is a set of real photos taken in the streets of 50 different cities with good weather conditions. The dataset contains $2,975$ images for training and 500 for testing, providing annotations for 19 semantic classes. The first and more naïve choice for adapting the dataset to the federated scenario consists of *uniformly* splitting it among a fixed number of clients, *i.e.* each image is randomly drawn and assigned to one of the users. While this distribution allows for performance evaluation in a federated environment, it does not consider the real-world challenge of statistical heterogeneity. Therefore, an additional split, denoted as *heterogeneous*, is introduced, taking into account the information about the city where the photo was taken, resulting in a non-*i.i.d.* domain distribution across clients. Specifically, there are 8 clients for each of the 18 training cities for a total of 144 devices, with images of each city divided among its clients. The test client contains all the images belonging to cities never seen at training time for both the uniform and the heterogeneous settings.

Federated IDDA. IDDA is a synthetic dataset for semantic segmentation in the field of self-driving cars, providing annotations for 16 semantic classes with a broad variety of driving conditions, characterized by three axes: 7 towns (ranging from Urban to Rural environments), 5 viewpoints (simulating different vehicles), and 3 weather conditions (Noon, Sunset and Rainy scenarios) for a total of 105 domains. FEDDRIVE provides both a homogeneous and a heterogeneous adaptation of IDDA to FL. In the *uniform* distribution, each client has access to 48 images drawn randomly

Table 6.1 Summary of the settings in FEDDRIVE

Dataset	Setting	Distribution	# Clients	# Samples per user	Test clients
Cityscapes	-	Uniform	146	10 - 40	Unseen domains (new cities)
	-	Heterogeneous	144	10 - 45	Unseen domains (new cities)
IDDA	country	uniform	90	48	Seen + unseen (country) domains
		Heterogeneous	90	48	Seen + unseen (country) domains
	Rainy	Uniform	69	48	Seen + unseen (rainy) domains
		Heterogeneous	69	48	Seen + unseen (rainy) domains

from the whole dataset. Since such distribution is highly unrealistic, it is only used as reference for the model's performance. The *heterogeneous* federated version of IDDA requires each client to see images belonging to a single domain. In addition, the generalization capabilities of the learned model are tested on two left-out clients, one with images belonging to the already seen training domains (*seen-dom*) and one with never seen ones (*unseen-dom*). IDDA is further distinguished into two possible settings in order to analyze both *semantic* and *appearance* shift. As for the former, the *unseen-dom* test client contains images of a country town, while in the second case, the photos are taken in rainy conditions. These two setting variations are respectively identified as *country* and *rainy*.

Benchmark

The algorithms benchmarked on FEDDRIVE are FEDAVG, FEDBN and SILOBN, which leverage the BN layers to learn domain-specific features.

To further overcome the limits of decentralized datasets, **Continuous Frequency Space Interpolation** (CFSI) [293] is applied to the images. Given an image $\boldsymbol{x}_k$ from the k-th client, the frequency space signal can be derived from the Fast Fourier Transform (FFT). The low-level information (*e.g.*, color, brightness) is reflected in the amplitude spectrum of the FFT. To share the distribution information among local clients, a distribution bank $\mathscr{A} = \{\mathscr{A}_1, \dots, \mathscr{A}_K\}$ is firstly created, where each $\mathscr{A}_k = \{\mathscr{A}_k^i\}_{i=1}^{N_k}$ contains all amplitude spectrum of images from the k-th client. A continuous interpolation of the frequencies is used to transfer multi-source distribution to each local client. [293] interpolates each client image to each domain distribution. Still, due to the significant computation required by the style transfer methods, this is not scalable or feasible in a realistic setting with several domains. As a result, CFSI transformation is only applied to half of the client data and only one

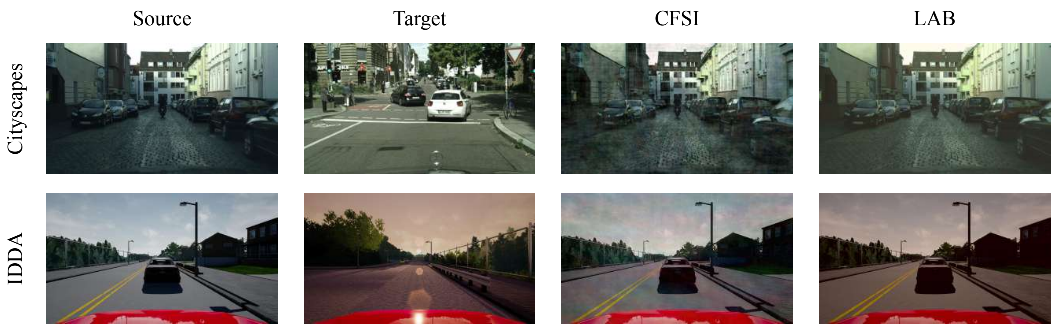


Fig. 6.2 CFSI and LAB applied to random images from Cityscapes and IDDA

target amplitude spectrum is randomly sampled from the shared distribution bank $\mathscr{A}$. The interpolation ratio λ is used to sample images uniformly in a range $[0,1]$.

Another useful technique for DG is **LAB-based Image Translation** (LAB) [455]. To share the distribution information across local clients, a shared bank $\mathscr{L} = \mathscr{L}_1, \dots, \mathscr{L}_K$ is created, where each $\mathscr{L}_k = \{\mu_k^i, \sigma_k^i\}_{i=1}^{N_k}$ is the set of means and standard deviations for each image of client k, transformed in the LAB color space. As previously done with CFSI, only half of the client data is translated. More specifically, a RGB image $\boldsymbol{x}_S^{RGB}$ of a client k is firstly converted to the LAB color space $\boldsymbol{x}_S^{LAB}$. Then, the mean μ_S and the standard deviation are computed for each channel of $\boldsymbol{x}_S^{LAB}$. A random pair of (μ_T, σ_T) is sampled from the distribution bank $\mathscr{L}$. The $\boldsymbol{x}_S^{LAB}$ image style is then translated as follows:

$$\hat{\boldsymbol{x}}_S^{LAB} = \frac{\boldsymbol{x}_S^{LAB} - \mu_S}{\sigma_S} * \sigma_T + \mu_T. \tag{6.1}$$

Following the alignment of the distribution, 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. Figure 6.2 compares CFSI and LAB.

The resulting benchmark is presented in Tables 6.2 and 6.3, with examples of qualitative results in Figure 6.3.

Overall, the improvements brought by SILOBN are limited in Cityscapes since the domain shift across cities is mostly related to semantic rather than style. Test images, in fact, belong to cities never seen during training but to the same country and are taken on similar weather conditions. For the same reason, DG methods do not contribute to performance improvement: SILOBN obtains 44.20% mIoU on

Table 6.2 FEDDRIVE: Cityscapes results

Method	**mIoU ± std (%)**
FEDAVG (uniform)	45.71 ± 0.37
FEDAVG	43.85 ± 1.24
FEDAVG + CFSI	41.50 ± 0.98
FEDAVG + LAB	39.20 ± 1.37
SILOBN	**44.20** ± **1.43**
SILOBN + CFSI	40.48 ± 1.40
SILOBN + LAB	42.23 ± 1.23

Table 6.3 FEDDRIVE: IDDA results in mIoU ± std (%).

	country		**rainy**	
Method	**seen**	**unseen**	**seen**	**unseen**
FEDAVG (uniform)	63.57 ± 0.60	49.74 ± 0.79	62.72 ± 3.65	27.61 ± 2.80
FEDAVG	42.43 ± 1.78	40.01 ± 1.26	38.18 ± 1.40	26.75 ± 2.32
FEDAVG + CFSI	54.70 ± 1.12	45.70 ± 1.73	55.24 ± 1.65	31.05 ± 2.68
FEDAVG + LAB	56.59 ± 0.90	45.68 ± 1.04	58.85 ± 0.89	26.82 ± 1.78
FEDBN	54.39	-	56.45	-
SILOBN	58.82 ± 2.93	45.32 ± 0.90	62.48 ± 1.42	50.03 ± 0.79
SILOBN + CFSI	61.22 ± 3.88	49.17 ± 1.01	63.04 ± 0.31	50.54 ± 0.88
SILOBN + LAB	**64.32** ± **0.76**	**50.43** ± **0.63**	**65.85** ± **0.91**	**53.99** ± **0.79**

average, while combining it with CFSI and LAB, it achieves, respectively, 40.48% and 42.23%. In particular, both CFSI and LAB act on the style of the image rather than on its content. A slight drop in performance is instead obtained since the network requires more time to converge with a stronger data augmentation given by the style transfer methods. Qualitative results in Figure 6.3 confirm that SILOBN with LAB does not improve the performance w.r.t. FEDAVG.

As for IDDA, on the *unseen* test client, the performances are lower, especially in the rainy setting, where the mIoU drops by 35 mIoU percentage points on average with respect to the *seen* test client. This underlines the significant domain shift introduced by the *unseen* test client. It is important to remark that country and rainy settings provide different challenges, as can be seen from Figure 6.3. The country domain has a similar style w.r.t. the training clients, since the images are taken in different weather conditions, but has other semantic characteristics, *e.g.* it rarely contains sidewalks near the road, as instead frequently happens on the training clients. On the opposite, the rainy domain has similar semantic characteristics but an essentially different appearance since there are no rainy images on training. When introducing a domain imbalance, the performance of FEDAVG has a significant drop, especially in the seen test client. On average, the performance on the seen test client drops by nearly 23 mIoU percentage points, achieving 42.43% on the country setting and 38.18% on the rainy one. This result emphasizes the domain imbalance across training clients and the high statistical heterogeneity of this setting. Differently, the performance drop on the unseen test client is limited, with a drop of about 2 mIoU percentage points on the country and a drop of about 12 mIoU percentage points on the rainy setting. The introduction of techniques able to cope with statistical heterogeneity largely improves the performance on the seen test client. FEDBN and SILOBN show an improvement, on average, of 14 mIoU percentage points on country and of 21 mIoU percentage points on rainy settings, strongly reducing the domain gap between training clients. Moreover, analyzing the results on the unseen test client, we see two different behaviors of SILOBN. On the country setting, it shows only a limited improvement with respect to FEDAVG: $40.01 \pm 1.26\%$ FEDAVG vs $45.32 \pm 0.90\%$ SILOBN. Differently, it largely improves the performance on the rainy setting: $26.75 \pm 2.32\%$ FEDAVG vs $50.03 \pm 0.79\%$ SILOBN. These results confirm that SILOBN strongly reduces the domain shift related to the appearance. Still, it does not alleviate much the semantic domain shift present in the country domain. Overall, applying CFSI and LAB style transfer methods improve the performance across domains and settings.

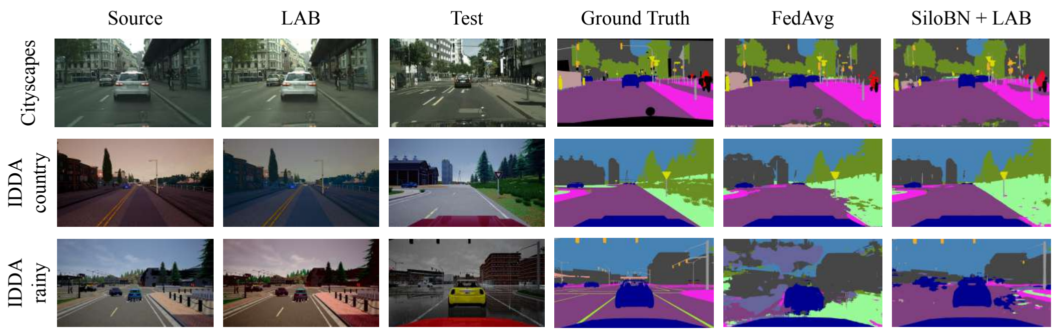


Fig. 6.3 Example of qualitative results on FEDDRIVE with various methods

6.2.3 Federated source-Free Domain Adaptation

As detailed in Section 5.3, this thesis introduces FFREEDA (Federated source-Free Domain Adaptation), a new framework enabling learning from unlabeled clients' local datasets by leveraging a labeled public dataset. This section discusses the benchmarks introduced for this specific scenario, summarized in Table 6.4.

The source dataset (*i.e.*, the public labeled server-side dataset) is the synthetic GTA5 dataset, containing $\approx 25k$ highly realistic road scenes of typical urban and suburban environments in the United States. As for the target datasets instead, three different options are proposed: Cityscapes [396], CrossCity [384] and Mapillary Vistas [397]. Cityscapes provides street-view images from 50 cities in Central Europe, for a total of 2,975 training images (unlabeled) and 500 validation images, used as test set. CrossCity includes more diverse locations and appearances, collecting driving scenes from multiple cities around the world (*i.e.*, Rome, Rio, Tokyo, and Taipei). Finally, the Mapillary Vistas dataset collects geo-localized street-view images from all around the world. This setting considers the largest number of overlapping classes among GTA5 and the real datasets (*i.e.*, 19 for Cityscapes and Mapillary, and 13 for CrossCity).

The federated partitioning of the datasets is done as follows. Cityscapes is split following FEDDRIVE [271], *i.e.*, 144 clients, where each client has between 10 and 45 samples belonging to a single city from the dataset.

CrossCity is divided by assigning 27 ± 10 images taken from the same city to each client, where the number of samples per client is uniformly sampled. The resulting distributions are balanced across cities, as shown in Figure 6.4.

In Mapillary, the provided GPS information is leveraged to discover clients with spatially near images. Starting from the original training set of 18000 images, 31 were discarded due to missing GPS coordinates. K-means was then applied to the GPS coordinates six times, one per continent. The k-Means algorithm is constrained to assign every client a random number of images in the range from 16 to 100. This

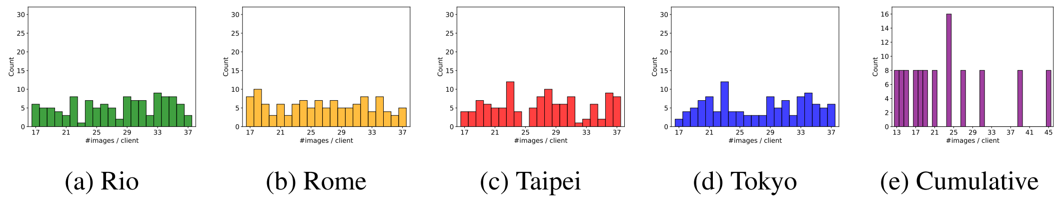


Fig. 6.4 Histogram of images per clients in the proposed federated CrossCity split

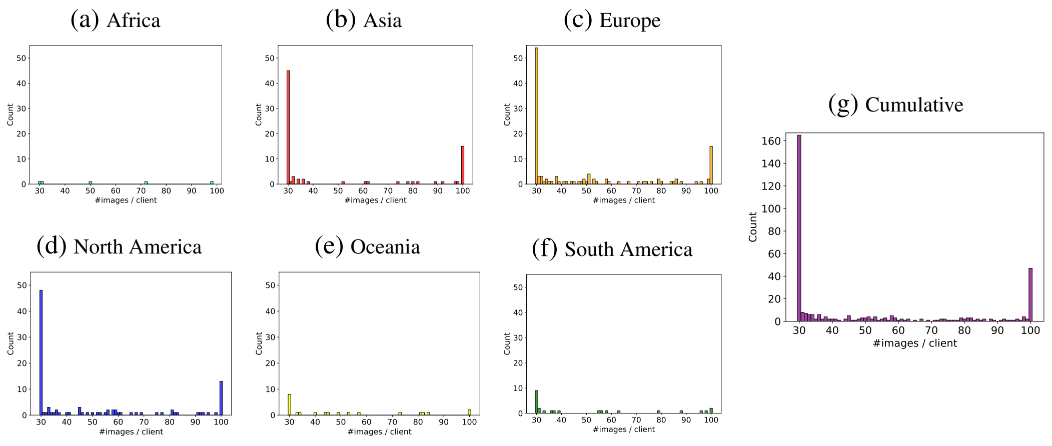


Fig. 6.5 Histogram of images per clients in the proposed federated Mapillary Vistas split

procedure resulted in 357 clients, where each client observes samples from only one continent. The final distributions of the number of images per client are shown in Figure 6.5. Unlike the other scenarios, here the variability across the distributions obtained in different continents is larger due to the highly imbalanced nature of the dataset. Also, note that the two entries with the highest values, 16 and 100, correspond to the extreme values of the constrained k-Means process.

Table 6.4 FFREEDA: Federated splits for semantic segmentation in FL. For each dataset, the following information are reported: the number of classes N_C, the size of training and test sets, the number of clients and the min-max range of images per client.

Dataset	N_C	$\lvert\mathscr{D}^T\rvert$	$\lvert\mathscr{D}^T_{test}\rvert$	K	**# Img/Client** (range)
Cityscapes	19	2,975	500	144	$[10,45]$
CrossCity	13	12,800	400	476	$[17,37]$
Mapillary	19	17,969	2,000	357	$[16,100]$

6.3 Federated Visual Place Recognition

Visual Place Recognition (VPR) [73] aims to estimate the location of an image by treating it as a retrieval problem. VPR uses a database of geo-tagged images and leverages deep neural networks to extract a global representation, called descriptor, from each image. While the training data for VPR models often originates from diverse, geographically scattered sources (geo-tagged images), the training process itself is typically assumed to be centralized. This research revisits the task of VPR through the lens of Federated Learning (FL), addressing several key challenges associated with this adaptation. VPR data inherently lacks well-defined classes, and models are typically trained using contrastive learning, which necessitates a data mining step on a centralized database. Additionally, client devices in federated systems can be highly heterogeneous in terms of their processing capabilities. The proposed FedVPR framework not only presents a novel approach for VPR but also introduces a new, challenging, and realistic task for FL research, paving the way to other image retrieval tasks in FL.

6.3.1 Introduction

The ability to recognize the place depicted in a picture is of the utmost importance for many modern applications performed by camera-equipped mobile systems. For example, in autonomous driving and mobile robotics, this ability is used for localization in instances where GPS measurement is unavailable or unreliable [456, 457], or in facilitating loop closure within SLAM (Simultaneous Localization and Mapping) pipelines [458]. Additionally, mobile phone applications heavily rely on this functionality for tasks like scene categorization [459] and augmented reality support [460]. Likewise, wearable devices leverage this capability to provide useful information to the user [461]. From a technical perspective, this task is referred to as *Visual Place Recognition* (VPR) [73] and is naturally framed as an image retrieval problem. The query image to be localized is compared via features-space k-nearest neighbor (kNN) [462] to a database of images representing the known or already-

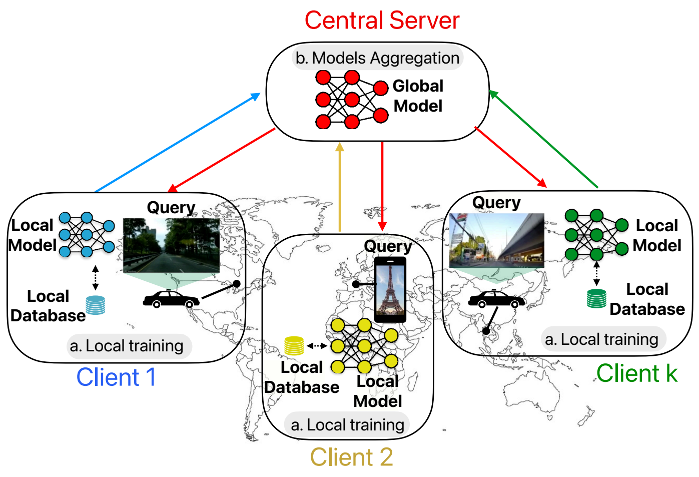


Fig. 6.6 **Federated Visual Place Recognition** (FedVPR): the training of Visual Place Recognition models is revisited from the perspective of Federated Learning, with clients distributed across geographical areas, each possessing heterogeneous computational and communication resources and availability. Instead of relying on a central database for mining, each client builds its own database of geo-tagged images and uses it for local training based on contrastive learning (step a.). Subsequently, it communicates its model weights to the server, where they are aggregated into a new global model (step b.).

visited places. Given that the database samples are usually labeled with geo-tags (such as GPS coordinates), the most similar images retrieved from the database serve as hypotheses of the queried location. This approach entails representing each image with a single vector (*global feature descriptor*) so that the kNN can efficiently compute the similarity between two images, *e.g.*, as an Euclidean distance.

Recent research on VPR has been focusing predominantly on the development of deep neural networks capable of extracting global feature descriptors that are both compact and highly informative for place recognition while leveraging large collections of data from highly heterogeneous distributions [463–468]. However, this centralized formulation assumes the images are readily available on one computer or a central server, which does not suit the distributed nature of the VPR applications previously discussed well. In an ideal scenario where mobile phones, wearable devices, and autonomous vehicles are deployed across numerous cities globally, it becomes crucial to leverage images collected by these diverse distributed devices without transferring their data to a central server, both for cost and privacy-related reasons. Furthermore, it would be beneficial to leverage the onboard computational capabilities of these devices to aid in model training.

In light of these considerations, this work aims to adapt the training paradigm of VPR models to the Federated Learning framework (Figure 6.6). Possible examples of the described scenario can be a company deploying a fleet of self-driving vehicles, wanting to improve the performances of its localization-and-mapping pipeline, a swarm of drones, or even general-purpose content-based image retrieval. However, this procedure is not straightforward. Unlike the conventional FL literature that revolves around classification problems [394, 275, 7], VPR lacks a clear division of data into classes. Instead, the collected images are labeled with continuous space annotations (commonly in the form of GPS coordinates), and models are usually trained with contrastive learning techniques [469], which are often performed in conjunction with computationally heavy mining over a large centralized database [469–472]: in a federated setting, this would be unfeasible due to (i) the low computational capacity of the clients and (ii) the privacy concerns that a centralized database would create.

The contributions can be listed as follows:

- Introduction of the first formulation of the VPR task in a federated learning framework. The importance of this formulation is twofold: for the VPR field, it opens up a new research direction with important practical implications; for the FL field, it provides a new downstream task that can broaden the horizon of the research community.
- Adaptation of the Mapillary Street-Level-Sequences (MSLS) dataset [473] to FL, to replicate realistic scenarios with varying degrees of statistical heterogeneity across clients.
- The clients' data heterogeneity is addressed through critical design decisions such as client split, local iteration scheduling, and data augmentation, achieving centralized-level performances while accounting for power and computational requirements.

6.3.2 Framework

Building upon the notation introduced in Section 2.1.2 for centralized VPR and the FL problem statement in Section 3.2, the FedVPR (Federated Visual Place Recognition) framework is defined as follows.

Each client $k \in \mathscr{C}$ has access to a privacy-protected dataset $\mathscr{D}_k$ made of N_k images $\boldsymbol{x} \in \mathbb{X}$ associated with a GPS location. While the server-side maintains the standard aggregation with FEDAVG, the client-side training requires detailing the mining procedure.

Local Mining. As discussed in Section 2.1.2, the mining process is crucial for learning as it ensures that the network is fed with informative samples during training. Differently from the centralized scenario where the model has access to the whole dataset for training, here the challenge lies in performing mining without (*i*) increasing the communication costs by exchanging continuous information between clients and server, *ii*) downloading enormous quantities of data on resource-constrained devices and (*iii*) exchanging data with other clients or the server, which could result in privacy leaks. In FedVPR, to avoid the aforementioned bottlenecks and any privacy concerns, the mining is limited to the database images previously collected by each client. The local data collection likely satisfies the requirement of having access to *hard negative* samples (*i.e.*, visually similar images of different places) to successfully train the feature extractor. However, its limited variability is an important factor that can slow down convergence and ultimately affects performances.

Hierarchical FL. Lastly, since here the clients' data distributions are linked with the users' geographical locations (*e.g.*, a device likely spends most time within a single region), this work additionally explores the hierarchical FL (H-FL) setup [317]. In H-FL, clients are grouped into K clusters according to their geographical proximity. A specialized model $F_{\mathbf{w}_c}$ is assigned to each cluster c. Once every T_c rounds, the cluster-specific models are aggregated. This implies the existence of multiple servers. FedVPR explores a dual-level framework: the first-tier servers handle inter-clusters interactions (*e.g.*, among cities or continents), while second-tier ones manage the intra-cluster exchanges (*e.g.*, between users living in the same city, or continent).

Table 6.5 Characteristics of the proposed FEDVPR datasets, associated with FEDAVG performances

FL dataset	Radius (m)	Sequences per client	Images per client	Number of clients	R@1 (%)
Centralized	-	-	-	-	66.0 ± 0.4
Random	-	64 ± 1	3655 ± 676	700	40.2 ± 0.0
Clustering	-	36 ± 32	2018 ± 1266	678	57.3 ± 1.2
Proximity	1000	17 ± 18	897 ± 808	1303	51.7 ± 1.7
	2000	33 ± 48	1834 ± 2050	713	61.0 ± 0.6
	4000	75 ± 148	4270 ± 6515	316	**66.1 ± 0.3**

Federated Mapillary Street-Level-Sequences Dataset

Our experiments center on the Mapillary Street-Level-Sequences (MSLS) dataset [473], geographically distributed across 30 cities worldwide, mimicking a FL scenario. The dataset is split into non-overlapping train, validation, and test sets. Each set comprises distinct cities: Amsterdam and Manila for validation, San Francisco and Copenhagen for testing, and the remaining cities for training. Similar to other VPR datasets, each subset is further divided into databases and queries. Queries represent images to be localized, while databases act as the system's prior knowledge of the area. Notably, the dataset excels due to its rich diversity. It encompasses a vast number of cities captured by various users, resulting in a wide range of cameras, weather conditions, times of day, and scenarios across both urban and rural environments. These characteristics perfectly align with the demands of FedVPR's use case.

This work's first contribution lies in proposing three novel splits for the MSLS dataset. These splits mimic real-world scenarios with varying data distributions across devices. Users are grouped based on geographical *proximity*, *similarity* in city features (*e.g.*, architecture), or *randomness*. Table 6.5 summarizes the introduced benchmark and Figure 6.7 reports the resulting statistics on clients and images distribution.

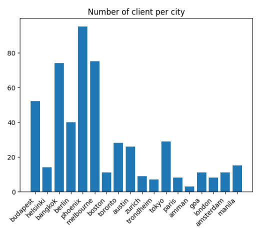
Proximity. This split emulates user movements within a neighborhood or a proximal geographical area. While clients in smaller towns may explore different localities, users in large cities like Tokyo or San Francisco are inclined to stay within their neighborhoods. The MSLS dataset is first divided geographically, with each city representing a separate entity. Within each city, clients are formed iteratively. An initial query image is chosen from a sequence available in that city. All other geographically close sequences, *i.e.*, within a given radius from the coordinates of the selected image, are then grouped with the chosen query image. This group is considered a valid client only if it contains at least two queries and two database sequences. The resulting number of training clients depends on the chosen radius, selected in $\{1000, 2000, 4000\}$ meters. Twelve clients are randomly selected from the pool of validated training clients to serve as the validation set. The test set is kept on the server side.

Clustering. The proximity split assumes similar features (*e.g.*, architecture) in nearby areas. However, distant neighborhoods might share more similarities (*e.g.*, busy streets, shops) than geographically close ones. To capture such nuances, the clustering split utilizes the K-means algorithm [89] at the city level, grouping images based on their visual and environmental characteristics. To ensure a balanced number of clients while capturing similarities, the value of K_{cl} (number of clusters) is set for each city individually. The number of clients obtained in the proximity split with a radius of 2000 meters is the reference, *i.e.*, $K_{cl} = 713$. The same selection criterion of the proximity split is then applied to define the valid clients. 12 clients are maintained for validation, and the test set is on the server.

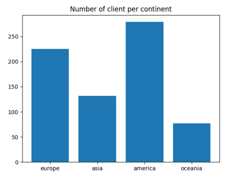
Random. Following the approach of [290, 271, 474],a *random* split of MSLS is introduced to emulate a uniform distribution and facilitate the understanding of the effects of statistical heterogeneity induced by domain shift. Each dataset's client includes images from all cities, and validation is conducted on the same local dataset. If a city does not have enough data for all clients, the existing sequences are duplicated until each client can access at least one sequence from each city. Any remaining sequences are redistributed among clients. The test set remains on the server side.

Benchmark

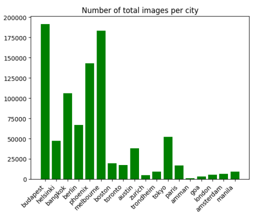
The experiments are run using a ResNet18 truncated after the third convolutional layer. The pooling layer is GeM. Each round engages 5 clients, with each client running a single local epoch. This process is repeated across a total of $T = 300$ rounds.



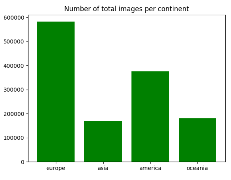
(a) Clients per city



(b) Clients per continent



(c) Images per city



(d) Images per continent

Fig. 6.7 Clients and images distributions in the federated MLSL

The Hierarchical Federated Learning (H-FL) [475, 476] experiments propose two hierarchy types, delineated by geographical proximity: *City* and *Continental* levels, where clients within the same city or continent respectively are aggregated to form the cluster-specific models. This results in 21 clusters in the former case and 4 in the latter.

Splits comparison. The vanilla FEDAVG algorithm is tested across the different introduced datasets, as illustrated in Table 6.5. Interestingly, the performance of FEDAVG on the Random FL dataset significantly lags behind that of other ones, despite the scenario closely resembling uniform splits as seen in prior works [290, 271, 474]. However, as highlighted in [477], optimal performance necessitates hard negatives to be situated within a distance range of 25 meters to a few kilometers from the query, a condition not met in this dataset where images within the same client can belong to various locations worldwide. The experiments on the Proximity and Clustering FL datasets yielded comparable performance. Interestingly, the Proximity split achieved slightly better results on average. This difference is likely because images within clients of the Proximity split have closer GPS coordinates compared to those in the Clustering split. Additionally, clients in the Proximity FL datasets with radii of 2000m and 4000m also contain a larger number of images compared to their counterparts in the Clustering datasets. The proximity experiment employing a radius of 4000m demonstrates performance levels roughly akin to the centralized baseline. However, opting for a larger radius results in fewer clients, each possessing a greater number of images and sequences, thereby resembling a cross-silo scenario [277]. Conversely, reducing the radius yields a larger number of clients but with a markedly limited quantity of images and sequences per client. Given all these considerations, the focus is shifted solely to the **proximity** FL dataset with a **2000m**

Table 6.6 Comparison of the vanilla baseline FedAvg with Hierarchical FL methods and various server optimizers. Notation: **CC** for continent-level middle servers in H-FL, **C** for city-level middle servers, SGDm for SGD with server-side momentum, T rounds, C clients participating at each round.

Algorithm	**Server Optimizer**	**R@1 (%)**
FedAvg	SGD	61.0 ± 0.6
	SGDm	61.2 ± 1.4
	Adam	61.1 ± 1.2
	AdaGrad	61.6 ± 0.3
H-FL (**CC**)	SGD	46.9 ± 1.3
FedAvg $T = 75, C = 20$	SGD	55.6 ± 0.8
H-FL (**C**)	SGD	33.3 ± 0.1
FedAvg $T = 15, C = 105$	SGD	44.2 ± 0.4

radius in the upcoming experiments. This choice strikes a balance between the number of clients and the volume of data per client.

Baselines. Table 6.6 presents a comparative analysis between FedAvg with various server-side optimizers and baselines sourced from the Hierarchical Federated Learning (H-FL) literature [475, 476]. This work distinguishes between H-FL at city (C) and continent level (CC). To ensure a fair comparison with H-FL, which selects 5 clients from each cluster per round, FedAvg is run with 20 clients per round and $T = 75$, and 105 participating clients for $T = 15$ rounds. Both H-FL experiments exhibit a reduction in recall by approximately 10%. This substantial decline in performance is associated with clusters tending to overfit the local distributions, thereby diminishing the meaningfulness of aggregation compared to training with all clients collectively. Concerning the server optimizers, AdaGrad demonstrates slightly superior performance compared to others. As a result, the standard SGD without momentum is chosen for the other baselines.

Data Quantity Skewness in FedVPR. Table 6.5 highlights significant variations in the number of sequences or images among clients (*i.e.*, *quantity heterogeneity*), particularly evident in the Proximity split with a radius of 2000m - the reference federated dataset. This paragraph analyzes how this phenomenon affects the final performance.

In heterogeneous settings, an increased number of local training steps (updates within a client over a batch of data) fosters client drift and destructive interference during aggregation (Chapter 4). Thus, a larger local dataset leads to more updates

Table 6.7 Addressing the clients' quantity heterogeneity. The R@1 (%) of the FEDAVG baseline (grey background) is compared with the ones of FEDAVG and FEDVC with a fixed number of iterations per client per round. *B* is the local mini-batch size.

Local Iterations	**Rounds**	**FEDAVG**	**FEDVC**
$\min(\lfloor \lvert\mathcal{D}_k\rvert / B \rfloor, 2500)$	300	61.0 ± 0.6	-
125	3200	66.6 ± 0.8	62.3 ± 1.1
250	1600	66.0 ± 1.7	65.9 ± 1.0
500	800	66.4 ± 1.6	$\mathbf{67.7 \pm 0.4}$
1000	400	61.7 ± 2.4	66.8 ± 0.5
2000	200	58.8 ± 1.8	65.2 ± 0.9
4000	100	57.3 ± 2.5	60.6 ± 1.0

and potentially negatively impacts the training process. Motivated by these insights, table 6.7 investigates how the data quantity skewness and the number of local training iterations affect performances of algorithms trained within the FedVPR framework, focusing on FEDAVG and the state-of-the-art algorithm FEDVC (Federated Virtual Clients) [278]. FEDVC specifically addresses variations in client data sizes by splitting large datasets into smaller clients and replicating smaller ones. This ensures all participating *virtual* clients contribute roughly the same amount of data during each training round. To prevent knowledge loss, larger clients are resampled with higher probability.

Given a fixed amount of total iterations I_{tot}, the analysis either varies the local iterations I_{loc}, or the training rounds T such that $I_{loc} \times T \times |\mathcal{C}^t| = I_{tot}$. With larger I_{loc}, smaller datasets are used multiple times within a client, while fewer iterations might lead to an incomplete view of larger datasets. We set $I_{tot} = 2,000,000$, $|\mathcal{C}|^t = 5$ and vary I_{loc} and T.

This analysis reveals that reducing the influence of data imbalances can improve performance by up to 5%. However, there exists a trade-off between communication rounds and final accuracy. As shown in Table 6.7, increasing T (more communication) while reducing the local steps leads to performance improvement. Conversely, excessively increasing local computation at the expense of the number of rounds deteriorates the final performance. Finally, FEDVC's sampling strategy, which favors larger clients, consistently improves performance when each device performs more than 500 local updates per round. However, for fewer local updates (125 and 250), FEDVC shows a decrease in accuracy.

6.4 Summary

In conclusion, this chapter introduced three novel benchmarks for studying computer vision tasks in FL, adapting both semantic segmentation for autonomous driving and visual place recognition to the federated setting.

FEDDRIVE is among the first works to propose using FL for autonomous vehicles, focusing on the semantic segmentation task. The proposed benchmark is based on the Cityscapes and IDDA datasets, and highlights the impact of style transfer techniques and batch normalization layers on the final model generalization ability.

FFREEDA proposes a more realistic task within the context of SS for FL, removing the need for labeled client-side data. The proposed benchmark introduces the federated adaptations of the CrossCity and Mapillary datasets.

FEDVPR adapts the Visual Place Recognition task to federated learning, introducing hierarchical FL to reflect the users' geographical distribution. The proposed datasets is Mapillary Street-Level-Sequences.

Chapter 7

Conclusion

Somewhere, something incredible
is waiting to be known

CARL SAGAN

This chapter discusses final considerations on the presented results (Section 7.1), highlights potential open directions and future works (Section 7.2).

7.1 Contributions Summary

This thesis investigated the critical challenge of generalization in federated learning (FL), particularly for complex real-world cross-device scenarios. These settings involve at least hundreds of clients, which autonomously collect their own data, inevitably influenced by factors like personal habits and geographical location. This leads to varying data distributions and inherent biases in the local datasets. In addition, edge devices are usually characterized by limited computational resources and varying availability. All these factors negatively impact and slow down the training process. This thesis aimed to improve the model generalization and robustness to distribution shifts, and speed up convergence in heterogeneous federated settings while maintaining communication efficiency, with a particular interest for vision-oriented solutions.

First, building upon research showing that convergence to sharp minima in the loss surface may cause poor generalization, this thesis explored the connection between the geometry of the global model's loss landscape and the lack of generalization in heterogeneous FL. This analysis revealed that FL global models tend to end up in sharp regions, especially under extreme data heterogeneity. As a consequence, this work investigated the potential benefits of promoting globally flat minima. To this end, as detailed in Chapter 4, FEDSAM and FEDGLOSS successfully leverage sharpness-aware minimization (SAM), resulting in improved global model generalization and convergence speedup in several tasks. Among their several advantages, differently from other state-of-the-art approaches, FEDSAM and FEDGLOSS enable the effective deployment in FL of both strong data augmentations and Lagrangian multipliers for local and global stationary points alignment.

Chapter 4 also revealed the importance of techniques that leverage previous global updates for improved robustness of the global model by introducing SWA and WIMA on the server side. By averaging the last server models, the proposed methods efficiently lead to more stable solutions and improved convergence speed without requiring any additional communication exchange or local computational effort. Their usage makes the federated training process more robust to potential clients' unavailability and data distribution shifts.

In addition, this research proposed alternative frameworks to the standard client-server training paradigm. By introducing various privacy-preserving techniques to infer the local data distributions - such as leveraging style information or classifier parameters - these novel methods enable the grouping of similar clients to develop tailored solutions. Moreover, this thesis challenges the conventional approach in FL of grouping similar clients to enhance performance. Instead, it proposes utilizing groups of *dissimilar* clients to achieve better model quality and convergence speed, as discussed in Chapter 5.

Furthermore, current research in FL focuses on either developing a global model that generalizes across the overall underlying distribution or creating solutions personalized to individual clients' data. This thesis argues that both approaches risk potential information loss. As a solution, this work proposes leveraging Graph Convolutional Neural Networks to effectively capture domain-specific information while facilitating knowledge exchange between similar clients (Chapter 5).

Finally, recognizing the need to bridge the gap between research and applications and aiming to promote future vision-oriented studies, novel benchmarks for complex vision tasks were developed. Chapter 6 introduces benchmarks designed for federated semantic segmentation in autonomous driving and collaborative visual place recognition, addressing the critical lack of suitable evaluation tools in these domains.

By investigating these key areas, this thesis has contributed to advancing FL for real-world vision applications. The proposed solutions pave the way for the development of more generalizable and robust FL models that can effectively handle the complexities of heterogeneous settings. Additionally, the proposed benchmarks provide a valuable foundation for continued research and development of FL algorithms for real-world computer vision tasks.

7.2 Open Directions and Future Works

This section discusses potential open directions and future works for improving generalization in heterogeneous federated learning, distinguished by the domain of application.

7.2.1 Flatness and generalization

This thesis introduced the intuition of leveraging the loss landscape perspective to analyze the behavior of local and global models in heterogeneous FL. However, these works have only scratched the surface, as many questions remain open.

First, some straightforward improvements of FEDSAM and FEDGLOSS are worth discussing. As outlined in Chapter 4, SAM requires double the computation cost at each iteration, which poses a significant challenge for deploying FEDSAM in resource-constrained devices commonly found in real-world scenarios. In centralized settings, this issue is addressed by several works that aim to improve SAM's efficiency without losing in performance, as discussed in Section 2.4. However, none of these efficiency-aware alternatives has been studied in the FL scenario. To this end, an open direction consists in finding a sharpness-aware approach the better fits the constraints of FL.

Moreover, existing global flatness-aware methods - FEDGLOSS, FEDSMOO, and FEDGAMMA- rely on stateful clients to perform operations such as the alternating direction method of multipliers for aligning local and global solutions (FEDGLOSS and FEDSMOO) or stochastic variance reduction (FEDGAMMA). This dependency may be impractical in cross-device FL scenarios involving millions of clients, many of whom are unlikely to participate in the training process more than once, potentially reducing the effectiveness of these methods. This underscores the need for alternative approaches to global flatness in FL that avoid reliance on stateful clients while maintaining communication efficiency and minimizing local computational effort compared to FEDSAM.

In addition, the proposed study on FEDGLOSS revealed that SAM enables the effective use of ADMM in FL, reducing the known risk of parameters explosion [8] by promoting smaller gradients. This lays the groundwork for future exploration on understanding whether similar stability can be achieved through alternative regularization techniques or optimization strategies.

Furthermore, recent studies in centralized learning suggest that SAM's success in achieving better generalization extends beyond its ability to select flat minima, involving several other factors [230, 231, 478, 479]. For instance, according to [478], SAM modifies the optimization trajectory by encouraging solutions that align with specific eigenvectors of the Hessian matrix, or it simplifies the learned representations by reducing the feature rank in intermediate layers of networks [231]. Similarly, Andriushchenko *et al.* [230] argue that SAM up-weights gradients from low-loss examples during training and amplifies the gradient contributions from less noisy samples, thereby prioritizing robust features.

These findings motivate further investigation into the behavior of SAM in FL. Notably, the centralized focus of these studies overlooks challenges unique to FL, such as small, imbalanced datasets, client drift, and server-side aggregation. A promising direction for future research is to explore the underlying mechanisms driving SAM's effectiveness in FL, particularly whether its use in local training implicitly regularizes client drift, how the implicit re-weighing mechanism of SAM impacts the quality of the locally learned model and the resulting global one, and how SAM locally interacts with the global trajectory implicitly memorized in the momentum term.

Lastly, by promoting convergence to flat minima, SAM leads to small gradient steps and implicitly promotes sparser solutions [227]. Such advantage could be exploited in FL scenarios to introduce both flatness-driven compression strategies for reduced message size over the network and sharpness-aware lottery ticket hypotheses for finding smaller sub-networks with equal performance [480].

7.2.2 Model Merging

Current research often uses FEDAVG as the base algorithm for further developments, relying on its basic approach of updating the global model through a weighted average of the model parameters. However, it has been extensively shown how this implementation is not robust to practical factors like the presence of data heterogeneity. A promising line of research looks at alternative model merging techniques.

From the perspective of the loss landscape, mode connectivity and neural networks subspaces [236, 481, 482] remain an under-explored area of research in FL. Mode connectivity refers to the phenomenon where different local minima, or solutions, in the loss landscape are connected by a low-loss path. This path suggests that multiple solutions may have similar generalization properties, and the model can transition between them without significant increases in loss. In the context of FL, leveraging this approach could enable the learning of local solutions that are mode connected, meaning that each solution achieves comparable loss across all clients' tasks, thus removing the need for merging altogether.

Other interesting approaches include [483], which focuses on merging models that have been trained on different tasks without the need for retraining or fine-tuning. This directly translates to the FL scenario, where multiple clients have access to different tasks or datasets.

7.2.3 Beyond Label Skew: Spurious Correlations in FL

Most of current research on heterogeneous FL focuses on the issues arising from label skew and domain generalization, disregarding other aspects, such as the presence of spurious correlations, which are particularly impactful in vision tasks. As discussed in Chapter 4, spurious correlations [198] are patterns that appear predictive in the training data but are misleading at test time (*e.g.*, the image background), and severely impact the final generalization performance. Current understanding suggests that the impact of spurious correlations in FL is still to be studied in depth and many questions remain unanswered. For instance, the presence of simpler and less complex tasks in the clients' local datasets can impact the features learned by the model and is unclear whether the learned local model may rely more on spurious information compared to the resulting global model.

In addition, recent research [484] shows that SAM implicitly balances the quality of diverse features, with positive implications for imbalanced dataset affected by the presence of spurious information. Thus, a clear path of research would consist in understanding the role of spurious correlations in heterogeneous FL, their impact on SAM's behavior and on the loss landscape geometry, and how the two aspects combine when using methods like FEDSAM and FEDGLOSS (Chapter 4).

To address this gap and further the practical application of FL, the research community should prioritize the development of robust and comprehensive FL benchmarks that explicitly study spurious correlations.

References

[1] Charlie Giattino, Edouard Mathieu, Veronika Samborska, and Max Roser. Data page: Cumulative number of notable ai systems by domain. https://ourworldindata.org/grapher/cumulative-number-of-notable-ai-systems-by-domain, 2023. Data adapted from Epoch, part of the publication "Artificial Intelligence".

[2] Charlie Giattino, Edouard Mathieu, Veronika Samborska, and Max Roser. Data page: Datapoints used to train notable artificial intelligence systems. https://ourworldindata.org/grapher/artificial-intelligence-number-training-datapoints, 2023. Data adapted from Epoch, part of the publication "Artificial Intelligence".

[3] Epoch. Training computation (petaflop). Dataset: Parameter, Compute and Data Trends in Machine Learning, 2024. with major processing by Our World in Data.

[4] Charlie Giattino, Edouard Mathieu, Veronika Samborska, and Max Roser. Data page: Hardware and energy cost to train notable ai systems. https://ourworldindata.org/grapher/hardware-and-energy-cost-to-train-notable-ai-systems, 2023. Data adapted from Epoch. In: Artificial Intelligence. Retrieved from Our World in Data.

[5] Epoch AI. Parameter, compute and data trends in machine learning. https://epochai.org/data/epochdb/visualization, 2023. Published online at epochai.org.

[6] 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.

[7] 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.

[8] Farshid Varno, Marzie Saghayi, Laya Rafiee Sevyeri, Sharut Gupta, Stan Matwin, and Mohammad Havaei. Adabest: Minimizing client drift in federated learning via adaptive bias estimation. In *European Conference on Computer Vision*, pages 710–726. Springer, 2022.

[9] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[10] Eugene Cheng-Xi Aw, Garry Wei-Han Tan, Tat-Huei Cham, Ramakrishnan Raman, and Keng-Boon Ooi. Alexa, what's on my shopping list? transforming customer experience with digital voice assistants. *Technological Forecasting and Social Change*, 180:121711, 2022.

[11] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, et al. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 958–979, 2024.

[12] OpenAI. Chatgpt. https://chat.openai.com, 2023.

[13] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S Yu, and Lichao Sun. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226*, 2023.

[14] Junyi Chai and Anming Li. Deep learning in natural language processing: A state-of-the-art survey. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1–6. IEEE, 2019.

[15] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2515–2528, 2022.

[16] Chenshuang Zhang, Chaoning Zhang, Sheng Zheng, Mengchun Zhang, Maryam Qamar, Sung-Ho Bae, and In So Kweon. A survey on audio diffusion models: Text to speech synthesis and enhancement in generative ai. *arXiv preprint arXiv:2303.13336*, 2023.

[17] 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.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, 2016.

[19] Shikun Zhang, Omid Jafari, and Parth Nagarkar. A survey on machine learning techniques for auto labeling of video, audio, and text data. *arXiv preprint arXiv:2109.03784*, 2021.

[20] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[21] OpenAI. Dall-e. https://openai.com/dall-e, 2023.

[22] Sai Munikoti, Ian Stewart, Sameera Horawalavithana, Henry Kvinge, Tegan Emerson, Sandra E Thompson, and Karl Pazdernik. Generalist multimodal ai: A review of architectures, challenges and opportunities. *arXiv preprint arXiv:2406.05496*, 2024.

[23] Devon Myers, Rami Mohawesh, Venkata Ishwarya Chellaboina, Anantha Lakshmi Sathvik, Praveen Venkatesh, Yi-Hui Ho, Hanna Henshaw, Muna Alhawawreh, David Berdik, and Yaser Jararweh. Foundation and large language models: fundamentals, challenges, opportunities, and social impacts. *Cluster Computing*, 27(1):1–26, 2024.

[24] Nobel Prize Outreach AB. Press release: The 2024 nobel prize in physics, 2024. Accessed: 31 October 2024.

[25] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[26] Nobel Prize Outreach AB. Press release: The 2024 nobel prize in chemistry, 2024. Accessed: 31 October 2024.

[27] Vivek Ramanujan, Thao Nguyen, Sewoong Oh, Ali Farhadi, and Ludwig Schmidt. On the connection between pre-training data diversity and fine-tuning robustness. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Lalit Pandey, Samantha Wood, and Justin Wood. Are vision transformers more data hungry than newborn visual systems? *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[30] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[31] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 561–578. Springer, 2020.

[32] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[33] David Reinsel. How you contribute to today's growing datasphere and its enterprise impact, 2019. IDC Blog, Accessed: 2024-10-21.

[34] 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.

[35] EU. Regulation (EU) 2016/679 of the European Parliament and of the Council 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). *General Data Protection Regulation*, 679, 2016.

[36] Abigail Goldsteen, Gilad Ezov, Ron Shmelkin, Micha Moffie, and Ariel Farkash. Data minimization for gdpr compliance in machine learning models. *AI and Ethics*, 2(3):477–491, 2022.

[37] Prakhar Ganesh, Cuong Tran, Reza Shokri, and Ferdinando Fioretto. The data minimization principle in machine learning. *arXiv preprint arXiv:2405.19471*, 2024.

[38] Kallista Bonawitz, Peter Kairouz, Brendan Mcmahan, and Daniel Ramage. Federated learning and privacy. *Communications of the ACM*, 65(4):90–97, 2022.

[39] Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

[40] Lane, Nicholas and Sani, Lorenzo and Iacob, Alex. Introducing flowerllm, 2024. Accessed: 2024-10-28.

[41] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[42] Google. Learn how google improves speech models, 2024. Accessed: 2024-10-22.

[43] Karen Hao. How apple personalizes siri without hoovering up your data, 2019. Accessed: 2024-10-22.

[44] 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.

[45] 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, 2020.

[46] 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.

[47] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[48] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[49] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[50] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[51] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[52] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

[53] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[54] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *International Conference on Learning Representations*, 2019.

[55] Lin Chen, Yifei Min, Mingrui Zhang, and Amin Karbasi. More data can expand the generalization gap between adversarially robust and standard models. In *International Conference on Machine Learning*, pages 1670–1680. PMLR, 2020.

[56] Rie Johnson and Tong Zhang. Inconsistency, instability, and generalization gap of deep neural network training. *Advances in Neural Information Processing Systems*, 36, 2024.

[57] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[58] H Jabbar and Rafiqul Zaman Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70(10.3850):978–981, 2015.

[59] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[60] Daniel Bashir, George D Montañez, Sonia Sehra, Pedro Sandoval Segura, and Julius Lauw. An information-theoretic perspective on overfitting and underfitting. In *AI 2020: Advances in Artificial Intelligence: 33rd Australasian Joint Conference, AI 2020, Canberra, ACT, Australia, November 29–30, 2020, Proceedings 33*, pages 347–358. Springer, 2020.

[61] Rahul Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *International conference on artificial intelligence and statistics*, pages 143–151. PMLR, 2018.

[62] Jake Lever, Martin Krzywinski, and Naomi Altman. Points of significance: model selection and overfitting. *Nature methods*, 13(9):703–705, 2016.

[63] Irving John Good. Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 14(1):107–114, 1952.

[64] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[65] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 7:87–93, 2018.

[66] Hongshan Yu, Zhengeng Yang, Lei Tan, Yaonan Wang, Wei Sun, Mingui Sun, and Yandong Tang. Methods and datasets on semantic segmentation: A review. *Neurocomputing*, 304:82–103, 2018.

[67] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020.

[68] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation? In *Bmvc*, volume 27, pages 10–5244. Bristol, 2013.

[69] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[70] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[71] Yu-Feng Li, Lan-Zhe Guo, and Zhi-Hua Zhou. Towards safe weakly supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):334–346, 2019.

[72] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *ieee transactions on robotics*, 32(1):1–19, 2015.

[73] Carlo Masone and Barbara Caputo. A survey on deep visual place recognition. *IEEE Access*, 9:19516–19547, 2021.

[74] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2021.

[75] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[76] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5396–5407, 2022.

[77] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in Neural Information Processing Systems*, 16, 2003.

[78] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[79] Zoubin Ghahramani. Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer, 2003.

[80] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 485–585, 2009.

[81] Dmitry Krotov and John J. Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, 116(16):7723–7731, March 2019.

[82] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[83] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[84] Jiaqi Liu, Guoyang Xie, Jinbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. Deep industrial image anomaly detection: A survey. *Machine Intelligence Research*, 21(1):104–135, 2024.

[85] Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[86] Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[87] Gbeminiyi John Oyewole and George Alex Thopil. Data clustering: application and trends. *Artificial Intelligence Review*, 56(7):6439–6475, 2023.

[88] Lior Rokach and Oded Maimon. Clustering methods. *Data mining and knowledge discovery handbook*, pages 321–352, 2005.

[89] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[90] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020.

[91] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A survey on self-supervised learning: Algorithms, applications, and future trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[92] Zejian Shi, Minyong Shi, and Chunfang Li. The prediction of character based on recurrent neural network language model. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 613–616. IEEE, 2017.

[93] Milind Soam and Sanjeev Thakur. Next word prediction using deep learning: A comparative study. In *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 653–658. IEEE, 2022.

[94] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[95] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[96] Hans-Dieter Block. The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34(1):123, 1962.

[97] 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.

[98] Abhinav Goel, Caleb Tung, Yung-Hsiang Lu, and George K Thiruvathukal. A survey of methods for low-power deep learning and computer vision. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.

[99] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, cnns and vision transformers: a review. *IEEE Access*, 2023.

[100] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[101] Jasmin Bharadiya. A comprehensive survey of deep learning techniques natural language processing. *European Journal of Technology*, 7(1):58–66, 2023.

[102] Xiaodong He and Li Deng. Deep learning for image-to-text generation: A technical overview. *IEEE Signal Processing Magazine*, 34(6):109–116, 2017.

[103] Dennis W Ruck, Steven K Rogers, and Matthew Kabrisky. Feature selection using a multilayer perceptron. *Journal of neural network computing*, 2(2):40–48, 1990.

[104] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[105] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[106] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[107] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[108] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[109] Stephen Grossberg. Recurrent neural networks. *Scholarpedia*, 8(2):1888, 2013.

[110] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. *Advances in Neural Information Processing Systems*, 20, 2007.

[111] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE workshop on automatic speech recognition and understanding (ASRU)*, pages 167–174. IEEE, 2015.

[112] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[113] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[114] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[115] 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.

[116] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[117] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[118] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

[119] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[120] Xinhua Zhang. Empirical risk minimization. *Encyclopedia of Machine Learning and Data Mining*, pages 392–393, 2017.

[121] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[122] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29, 2016.

[123] Aatila Mustapha, Lachgar Mohamed, and Kartit Ali. An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field. In *Smart Applications and Data Analysis: Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26, 2020, Proceedings 3*, pages 349–359. Springer, 2020.

[124] William Carlisle Thacker. The role of the hessian matrix in fitting models to measurements. *Journal of Geophysical Research: Oceans*, 94(C5):6177–6196, 1989.

[125] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257. PMLR, 2016.

[126] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[127] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *International Conference on Learning Representations*, 2019.

[128] David C Plaut and Geoffrey E Hinton. Learning sets of filters using back-propagation. *Computer Speech & Language*, 2(1):35–61, 1987.

[129] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[130] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in Neural Information Processing Systems*, 31, 2018.

[131] Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322, 2022.

[132] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *International Conference on Learning Representations*, 2020.

[133] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in Neural Information Processing Systems*, 31, 2018.

[134] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2018.

[135] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. In *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022.

[136] Pan Zhou and Jiashi Feng. Understanding generalization and optimization performance of deep cnns. In *International Conference on Machine Learning*, pages 5960–5969. PMLR, 2018.

[137] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[138] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[139] Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. *Advances in Neural Information Processing Systems*, 32, 2019.

[140] Donghyun Kim, Kaihong Wang, Stan Sclaroff, and Kate Saenko. A broad study of pre-training for domain generalization and adaptation. In *European Conference on Computer Vision*, pages 621–638. Springer, 2022.

[141] Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, and Amanda Gonsalves. Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441, 2020.

[142] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[143] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 466–477, 2021.

[144] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[145] 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.

[146] 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.

[147] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2022.

[148] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. 2008.

[149] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8156–8164, 2018.

[150] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2720–2729, 2019.

[151] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[152] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[153] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[154] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018.

[155] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[156] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[157] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

[158] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[159] Guillaume Leclerc and Aleksander Madry. The two regimes of deep network training. *arXiv preprint arXiv:2002.10376*, 2020.

[160] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33:18261–18271, 2020.

[161] Ali Ramezani-Kebrya, Ashish Khisti, and Ben Liang. On the generalization of stochastic gradient descent with momentum, 2021.

[162] Samy Jelassi and Yuanzhi Li. Towards understanding how momentum improves generalization in deep learning. In *International Conference on Machine Learning*, pages 9965–10040. PMLR, 2022.

[163] Rebecca C Fitzgerald, Antonis C Antoniou, Ljiljana Fruk, and Nitzan Rosenfeld. The future of early cancer detection. *Nature medicine*, 28(4):666–677, 2022.

[164] Asma Cherif, Arwa Badhib, Heyfa Ammar, Suhair Alshehri, Manal Kalkatawi, and Abdessamad Imine. Credit card fraud detection in the era of disruptive technologies: A systematic review. *Journal of King Saud University-Computer and Information Sciences*, 35(1):145–174, 2023.

[165] Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16:449–475, 2013.

[166] Rangachari Anand, Kishan G Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. An improved algorithm for neural network classification of imbalanced training sets. *IEEE transactions on neural networks*, 4(6):962–969, 1993.

[167] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[168] Samira Pouyanfar, Yudong Tao, Anup Mohan, Haiman Tian, Ahmed S Kaseb, Kent Gauen, Ryan Dailey, Sarah Aghajanzadeh, Yung-Hsiang Lu, Shu-Ching Chen, et al. Dynamic sampling in convolutional neural networks for imbalanced data classification. In *2018 IEEE conference on multimedia information processing and retrieval (MIPR)*, pages 112–117. IEEE, 2018.

[169] Hansang Lee, Minseok Park, and Junmo Kim. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE international conference on image processing (ICIP)*, pages 3713–3717. IEEE, 2016.

[170] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

[171] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[172] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[173] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization in vision: A survey. *arXiv preprint arXiv:2103.02503*, 2, 2021.

[174] 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.

[175] Ashish Rauniyar, Desta Haileselassie Hagos, Debesh Jha, Jan Erik Håkegård, Ulas Bagci, Danda B Rawat, and Vladimir Vlassov. Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions. *IEEE Internet of Things Journal*, 2023.

[176] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[177] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in Neural Information Processing Systems*, 31, 2018.

[178] Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. On the generalization effects of linear transformations in data augmentation. In *International Conference on Machine Learning*, pages 10410–10420. PMLR, 2020.

[179] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 3(1):91–99, 2022.

[180] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[181] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[182] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. In *2018 IEEE international conference on big data (Big Data)*, pages 3873–3882. IEEE, 2018.

[183] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, and David Owen. The rising costs of training frontier ai models. *arXiv preprint arXiv:2405.21015*, 2024.

[184] Nobel Dhar, Bobin Deng, Dan Lo, Xiaofeng Wu, Liang Zhao, and Kun Suo. An empirical analysis and resource footprint study of deploying large language models on edge devices. In *Proceedings of the 2024 ACM Southeast Conference*, ACM SE '24, page 69–76, New York, NY, USA, 2024. Association for Computing Machinery.

[185] Yi Sheng, Junhuan Yang, Yawen Wu, Kevin Mao, Yiyu Shi, Jingtong Hu, Weiwen Jiang, and Lei Yang. The larger the fairer? small neural networks can achieve fairness for edge devices. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 163–168, 2022.

[186] Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.

[187] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Just dial: Domain alignment layers for unsupervised domain adaptation. In *Image Analysis and Processing-ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part I 19*, pages 357–369. Springer, 2017.

[188] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *Proceedings of the IEEE international conference on computer vision*, pages 5067–5075, 2017.

[189] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Adagraph: Unifying predictive and continuous domain adaptation through graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6568–6577, 2019.

[190] 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.

[191] 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.

[192] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 68–83. Springer, 2020.

[193] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *International Conference on Learning Representations*, 2021.

[194] Mattia Segu, Alessio Tonioni, and Federico Tombari. Batch normalization embeddings for deep domain generalization. *Pattern Recognition*, 135:109115, 2023.

[195] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[196] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations*, 2019.

[197] Hengyue Liang, Le Peng, and Ju Sun. Selective classification under distribution shifts. *arXiv preprint arXiv:2405.05160*, 2024.

[198] Brian D Haig. What is a spurious correlation? *Understanding Statistics: Statistical Issues in Psychology, Education, and the Social Sciences*, 2(2):125–132, 2003.

[199] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *International Conference on Learning Representations*, 2023.

[200] Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35:38516–38532, 2022.

[201] 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.

[202] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[203] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[204] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[205] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

[206] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[207] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[208] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*, 2018.

[209] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[210] Carlo Baldassi, Clarissa Lauditi, Enrico M Malatesta, Gabriele Perugini, and Riccardo Zecchina. Unveiling the structure of wide flat minima in neural networks. *Physical Review Letters*, 127(27):278301, 2021.

[211] Huanran Chen, Shitong Shao, Ziyi Wang, Zirui Shang, Jin Chen, Xiaofeng Ji, and Xinxiao Wu. Bootstrap generalization ability from loss landscape perspective. In *European Conference on Computer Vision*, pages 500–517. Springer, 2022.

[212] Xingxuan Zhang, Renzhe Xu, Han Yu, Hao Zou, and Peng Cui. Gradient norm aware minimization seeks first-order flatness and improves generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20247–20257, June 2023.

[213] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *International Conference on Machine Learning*, 2021.

[214] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. In *International Conference on Learning Representations*, 2022.

[215] Dara Bahri, Hossein Mobahi, and Yi Tay. Sharpness-aware minimization improves language model generalization. *arXiv preprint arXiv:2110.08529*, 2021.

[216] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *International Conference on Learning Representations*, 2022.

[217] Wenxuan Zhou, Fangyu Liu, Huan Zhang, and Muhao Chen. Sharpness-aware minimization with dynamic reweighting. *arXiv preprint arXiv:2112.08772*, 2021.

[218] Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12360–12370, 2022.

[219] Jinseong Park, Hoki Kim, Yujin Choi, and Jaewook Lee. Differentially private sharpness-aware training. In *International Conference on Machine Learning*, pages 27204–27224. PMLR, 2023.

[220] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[221] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. *Advances in Neural Information Processing Systems*, 35:23439–23451, 2022.

[222] Renkun Ni, Ping-yeh Chiang, Jonas Geiping, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. K-sam: Sharpness-aware minimization at the speed of sgd. *arXiv preprint arXiv:2210.12864*, 2022.

[223] Maximilian Mueller, Tiffany Vlaar, David Rolnick, and Matthias Hein. Normalization layers are all that sharpness-aware minimization needs. *Advances in Neural Information Processing Systems*, 36, 2024.

[224] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *International Conference on Learning Representations*, 2022.

[225] Yixuan Zhou, Yi Qu, Xing Xu, and Hengtao Shen. Imbsam: A closer look at sharpness-aware minimization in class-imbalanced recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11345–11355, 2023.

[226] Yun Yue, Jiadi Jiang, Zhiling Ye, Ning Gao, Yongchao Liu, and Ke Zhang. Sharpness-aware minimization revisited: Weighted sharpness as a regularization term. *arXiv preprint arXiv:2305.15817*, 2023.

[227] Peng Mi, Li Shen, Tianhe Ren, Yiyi Zhou, Tianshuo Xu, Xiaoshuai Sun, Tongliang Liu, Rongrong Ji, and Dacheng Tao. Systematic investigation of sparse perturbed sharpness-aware minimization optimizer. *arXiv preprint arXiv:2306.17504*, 2023.

[228] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.

[229] Gonçalo Mordido, Sarath Chandar, and François Leduc-Primeau. Sharpness-aware training for accurate inference on noisy dnn accelerators. *arXiv preprint arXiv:2211.11561*, 2022.

[230] Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pages 639–668. PMLR, 2022.

[231] Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. Sharpness-aware minimization leads to low-rank features. *arXiv preprint arXiv:2305.16292*, 2023.

[232] Yan Dai, Kwangjun Ahn, and Suvrit Sra. The crucial role of normalization in sharpness-aware minimization. *arXiv preprint arXiv:2305.15287*, 2023.

[233] Yihao Zhang, Hangzhou He, Jingyu Zhu, Huanran Chen, Yifei Wang, and Zeming Wei. On the duality between sharpness-aware minimization and adversarial training. *International Conference on Machine Learning*, 2024.

[234] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[235] Ammar Mohammed and Rania Kora. A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 35(2):757–774, 2023.

[236] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[237] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *Conference on Uncertainty in Artificial Intelligence*, 2018.

[238] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.

[239] Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. When do flat minima optimizers work? *Advances in Neural Information Processing Systems*, 35:16577–16595, 2022.

[240] Jean Kaddour. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

[241] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of

multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[242] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *NeurIPS*, 2018.

[243] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[244] Peng Xu, Bryan He, Christopher De Sa, Ioannis Mitliagkas, and Chris Re. Accelerated stochastic power iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 58–67. PMLR, 2018.

[245] Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition, October 2018.

[246] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[247] Liangqi Yuan, Ziran Wang, Lichao Sun, S Yu Philip, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 2024.

[248] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the second workshop on distributed infrastructures for deep learning*, pages 1–8, 2018.

[249] John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? on the impact of pre-training and initialization in federated learning. *Advances in Neural Information Processing Systems Workshop*, 2022.

[250] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in Neural Information Processing Systems*, 35:19332–19344, 2022.

[251] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. *International Conference on Learning Representations*, 2023.

[252] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[253] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In *Artificial Intelligence and Statistics*, pages 509–517. PMLR, 2017.

[254] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[255] Sawsan AbdulRahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2020.

[256] Paolo Bellavista, Luca Foschini, and Alessio Mora. Decentralised learning in federated deployment environments: A system-level survey. *ACM Computing Surveys (CSUR)*, 54(1):1–38, 2021.

[257] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

[258] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. Spreadgnn: Serverless multi-task federated learning for graph neural networks. *arXiv preprint arXiv:2106.02743*, 2021.

[259] Hong Xing, Osvaldo Simeone, and Suzhi Bi. Federated learning over wireless device-to-device networks: Algorithms and convergence analysis. *IEEE Journal on Selected Areas in Communications*, 39(12):3723–3741, 2021.

[260] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of healthcare informatics research*, 5:1–19, 2021.

[261] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *International Conference on Learning Representations*, 2018.

[262] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[263] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 2023.

[264] 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.

[265] Tomisin Awosika, Raj Mani Shukla, and Bernardi Pranggono. Transparency and privacy: the role of explainable ai and federated learning in financial fraud detection. *IEEE Access*, 2024.

[266] Zichen Xu, Li Li, and Wenting Zou. Exploring federated learning on battery-powered devices. In *Proceedings of the ACM Turing Celebration Conference-China*, pages 1–6, 2019.

[267] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, and Albert Y Zomaya. Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. *IEEE Internet of Things Journal*, 9(12):10257–10271, 2021.

[268] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1):24–29, 2022.

[269] Shiva Raj Pokhrel and Jinho Choi. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE, 2020.

[270] 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.

[271] 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 *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11504–11511. IEEE, 2022.

[272] Vishnu Pandi Chellapandi, Liangqi Yuan, Christopher G Brinton, Stanislaw H Żak, and Ziran Wang. Federated learning for connected and automated vehicles: A survey of existing approaches and challenges. *IEEE Transactions on Intelligent Vehicles*, 2023.

[273] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[274] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[275] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *International Conference on Learning Representations*, 2021.

[276] Kate Donahue and Jon Kleinberg. Optimality and stability in federated learning: A game-theoretic approach. *Advances in Neural Information Processing Systems*, 34:1287–1298, 2021.

[277] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[278] 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.

[279] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[280] 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.

[281] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *International Conference on Learning Representations*, 2020.

[282] 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. *Advances in Neural Information Processing Systems*, 2020.

[283] 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.

[284] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[285] Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022.

[286] Zili Lu, Heng Pan, Yueyue Dai, Xueming Si, and Yan Zhang. Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 2024.

[287] Bingyan Liu, Nuoyan Lv, Yuanchun Guo, and Yawen Li. Recent advances on federated learning: A systematic survey. *Neurocomputing*, page 128019, 2024.

[288] 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.

[289] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[290] 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.

[291] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. Fedcm: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.

[292] Geeho Kim, Jinkyu Kim, and Bohyung Han. Communication-efficient federated learning with acceleration of global momentum. *arXiv preprint arXiv:2201.03172*, 2022.

[293] 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.

[294] Junming Chen, Meirui Jiang, Qi Dou, and Qifeng Chen. Federated domain generalization for image recognition via cross-client style transfer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 361–370, 2023.

[295] Yuwei Sun, Ng Chong, and Hideya Ochiai. Feature distribution matching for federated domain generalization. In *Asian Conference on Machine Learning*, pages 942–957. PMLR, 2023.

[296] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *International Conference on Learning Representations*, 2020.

[297] 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.

[298] Umberto Michieli and Mete Ozay. Are all users treated fairly in federated learning systems? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2318–2322, 2021.

[299] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *International Conference on Learning Representations*, 2019.

[300] Emre Ozfatura, Kerem Ozfatura, and Deniz Gündüz. Fedadc: Accelerated federated learning with drift control. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 467–472. IEEE, 2021.

[301] Rudrajit Das, Anish Acharya, Abolfazl Hashemi, Sujay Sanghavi, Inderjit S Dhillon, and Ufuk Topcu. Faster non-convex federated learning via global and local momentum. In *Uncertainty in Artificial Intelligence*, pages 496–506. PMLR, 2022.

[302] Geeho Kim, Jinkyu Kim, and Bohyung Han. Communication-efficient federated learning with accelerated client gradient. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12385–12394, 2024.

[303] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *Advances in Neural Information Processing Systems Workshop*, 2019.

[304] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153, 2022.

[305] Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Accelerating heterogeneous federated learning with closed-form classifiers. *International Conference on Machine Learning*, 2024.

[306] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[307] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. *Advances in Neural Information Processing Systems*, 32, 2019.

[308] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[309] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *Advances in Neural Information Processing Systems*, 2020.

[310] Durmus Alp Emre Acar, Yue Zhao, Ruizhao Zhu, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Debiasing model updates for improving personalized federated training. In *International conference on machine learning*, pages 21–31. PMLR, 2021.

[311] Wenbo Zheng, Lan Yan, Chao Gou, and Fei-Yue Wang. Federated meta-learning for fraudulent credit card detection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4654–4660, 2021.

[312] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[313] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[314] Luca Corinzia, Ami Beuret, and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[315] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34:15434–15447, 2021.

[316] Jiayi Chen and Aidong Zhang. Fedmsplit: Correlation-adaptive federated multi-task learning across multimodal split networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 87–96, 2022.

[317] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[318] 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.

[319] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674, 2021.

[320] Yihan Yan, Xiaojun Tong, and Shen Wang. Clustered federated learning in heterogeneous environment. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[321] Martin Papenberg and Gunnar W Klau. Using anticlustering to partition data sets into equivalent parts. *Psychological Methods*, 26(2):161, 2021.

[322] Shenglai Zeng, Zonghang Li, Hongfang Yu, Yihong He, Zenglin Xu, Dusit Niyato, and Han Yu. Heterogeneous federated learning via grouped sequential-to-parallel training. In *International Conference on Database Systems for Advanced Applications*, pages 455–471. Springer, 2022.

[323] Apple Machine Learning Research. Personalized hey siri, April 2018.

[324] Zhiyong Chen and Shugong Xu. Learning domain-heterogeneous speaker recognition systems with personalized continual federated learning. *EURASIP Journal on Audio, Speech, and Music Processing*, 2023(1):33, 2023.

[325] Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 21(8):2818–2832, 2020.

[326] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.

[327] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[328] Fahad Sabah, Yuwen Chen, Zhen Yang, Muhammad Azam, Nadeem Ahmad, and Raheem Sarwar. Model optimization techniques in personalized federated learning: A survey. *Expert Systems with Applications*, page 122874, 2023.

[329] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[330] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020.

[331] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. *International Conference on Learning Representations*, 2022.

[332] Zihao Zhao, Yuzhu Mao, Yang Liu, Linqi Song, Ye Ouyang, Xinlei Chen, and Wenbo Ding. Towards efficient communications in federated learning: A contemporary survey. *Journal of the Franklin Institute*, 360(12):8669–8703, 2023.

[333] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020, 2016.

[334] Yanjun Han, Ayfer Özgür, and Tsachy Weissman. Geometric lower bounds for distributed parameter estimation under communication constraints. In *Conference On Learning Theory*, pages 3163–3188. PMLR, 2018.

[335] Leighton Pate Barnes, Yanjun Han, and Ayfer Ozgur. Lower bounds for learning distributions under communication constraints via fisher information. *Journal of Machine Learning Research*, 21(236):1–30, 2020.

[336] Emre Ozfatura, Kerem Ozfatura, and Deniz Gündüz. Time-correlated sparsification for communication-efficient federated learning. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 461–466. IEEE, 2021.

[337] Runmeng Du, Daojing He, Zikang Ding, Miao Wang, Sammy Chan, and Xuru Li. Gsasg: Global sparsification with adaptive aggregated stochastic gradients for communication-efficient federated learning. *IEEE Internet of Things Journal*, 2024.

[338] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International conference on artificial intelligence and statistics*, pages 2021–2031. PMLR, 2020.

[339] Divyansh Jhunjhunwala, Advait Gadhikar, Gauri Joshi, and Yonina C Eldar. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3110–3114. IEEE, 2021.

[340] Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

[341] Xinyan Dai, Xiao Yan, Kaiwen Zhou, Han Yang, Kelvin KW Ng, James Cheng, and Yu Fan. Hyper-sphere quantization: Communication-efficient sgd for federated learning. *arXiv preprint arXiv:1911.04655*, 2019.

[342] Mohammad Mohammadi Amiri, Deniz Gunduz, Sanjeev R Kulkarni, and H Vincent Poor. Federated learning with quantized global model updates. *arXiv preprint arXiv:2006.10672*, 2020.

[343] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Towards understanding biased client selection in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 10351–10375. PMLR, 2022.

[344] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[345] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

[346] Changlin Song, Divya Saxena, Jiannong Cao, and Yuqing Zhao. Feddistill: Global model distillation for local model de-biasing in non-iid federated learning. *arXiv preprint arXiv:2404.09210*, 2024.

[347] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *IEEE 8th European Symposium on Security and Privacy*, pages 175–199. IEEE, 2023.

[348] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. Is federated learning a practical pet yet? *arXiv:2301.04017*, 2023.

[349] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[350] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 Association for Computing Machinery Special Interest Group on Security, Audit and Control (SIGSAC) conference on computer and communications security*, pages 603–618, 2017.

[351] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the Association for Computing Machinery*, 63:139–144, 2020.

[352] 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*, pages 691–706. IEEE, 2019.

[353] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10132–10142, 2022.

[354] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[355] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proceedings of the Twenty-Third International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[356] Ligeng Zhu, Song Liu, and Yin Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[357] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv:2202.00580*, 2022.

[358] Haokun Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4):94, 2021.

[359] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems*, 37(9):5880–5901, 2022.

[360] Anass El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE Access*, 10:22359–22380, 2022.

[361] 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.

[362] Najeeb Moharram Jebreel, Josep Domingo-Ferrer, Alberto Blanco-Justicia, and David Sánchez. Enhanced security and privacy via fragmented federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2022.

[363] Sheng Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519. ACM, 2016.

[364] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[365] Junbo Wang, Amitangshu Pal, Qinglin Yang, Krishna Kant, Kaiming Zhu, and Song Guo. Collaborative machine learning: Schemes, robustness, and privacy. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–18, 2022.

[366] Donald Shenaj, Giulia Rizzoli, and Pietro Zanuttigh. Federated learning in computer vision. *IEEE Access*, 2023.

[367] 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 *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[368] 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 *International workshop on machine learning in medical imaging*, pages 133–141. Springer, 2019.

[369] 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.

[370] 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.

[371] Chris Xing Tian, Haoliang Li, Yufei Wang, and Shiqi Wang. Privacy-preserving constrained domain generalization for medical image classification. *arXiv preprint arXiv:2105.08511*, 2021.

[372] Holger R Roth, Ken Chang, Praveer Singh, Nir Neumark, Wenqi Li, Vikash Gupta, Sharut Gupta, Liangqiong Qu, Alvin Ihsani, Bernardo C Bizzo, et al. Federated learning for breast density classification: A real-world implementation. 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 181–191. Springer, 2020.

[373] Pragati Baheti, Mukul Sikka, KV Arya, and R Rajesh. Federated learning on distributed medical records for detection of lung nodules. In *VISIGRAPP (4: VISAPP)*, pages 445–451, 2020.

[374] Amelia Jiménez-Sánchez, Mickael Tardy, Miguel A González Ballester, Diana Mateus, and Gemma Piella. Memory-aware curriculum federated learning for breast cancer classification. *Computer Methods and Programs in Biomedicine*, 229:107318, 2023.

[375] Pengfei Guo, Puyang Wang, Jinyuan Zhou, Shanshan Jiang, and Vishal M Patel. Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2423–2432, 2021.

[376] Chun-Mei Feng, Yunlu Yan, Shanshan Wang, Yong Xu, Ling Shao, and Huazhu Fu. Specificity-preserving federated learning for mr image reconstruction. *IEEE Transactions on Medical Imaging*, 2022.

[377] Gokberk Elmas, Salman UH Dar, Yilmaz Korkmaz, Emir Ceyani, Burak Susam, Muzaffer Ozbey, Salman Avestimehr, and Tolga Çukur. Federated learning of generative image priors for mri reconstruction. *IEEE Transactions on Medical Imaging*, 2022.

[378] Xuanang Xu, Hannah H Deng, Tianyi Chen, Tianshu Kuang, Joshua C Barber, Daeseung Kim, Jaime Gateno, James J Xia, and Pingkun Yan. Federated cross learning for medical image segmentation. In *Medical Imaging with Deep Learning*, pages 1441–1452. PMLR, 2024.

[379] Bingjie Yan, Jun Wang, Jieren Cheng, Yize Zhou, Yixian Zhang, Yifan Yang, Li Liu, Haojiang Zhao, Chunjuan Wang, and Boyi Liu. Experiments of federated learning for covid-19 chest x-ray images. In *Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part II 7*, pages 41–53. Springer, 2021.

[380] Sadaf Naz, Khoa T Phan, and Yi-Ping Phoebe Chen. A comprehensive review of federated learning for covid-19 detection. *International Journal of Intelligent Systems*, 37(3):2371–2392, 2022.

[381] 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.

[382] 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.

[383] Jiahuan Luo, Xueyang Wu, Yun Luo, Anbu Huang, Yunfeng Huang, Yang Liu, and Qiang Yang. Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*, 2019.

[384] Shangchao Su, Bin Li, Chengzhi Zhang, Mingzhao Yang, and Xiangyang Xue. Cross-domain federated object detection. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1469–1474. IEEE, 2023.

[385] Umberto Michieli, Marco Toldo, and Mete Ozay. Federated learning via attentive margin of semantic feature representations. *IEEE Internet of Things Journal*, 10(2):1517–1535, 2022.

[386] 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.

[387] Divyansh Aggarwal, Jiayu Zhou, and Anil K Jain. Fedface: Collaborative learning of face recognition model. In *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8. IEEE, 2021.

[388] 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.

[389] 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.

[390] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[391] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584, 2006.

[392] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[393] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[394] 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. *Workshop on Data Privacy and Confidentiality*, 2019.

[395] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[396] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[397] 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.

[398] 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.

[399] 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.

[400] Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary street-level sequences: A dataset for lifelong place recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2626–2635, 2020.

[401] 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.

[402] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[403] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[404] Irwan Bello, William Fedus, Xianzhi Du, Ekin Dogus Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34, 2021.

[405] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[406] Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. Generalized federated learning via sharpness aware minimization. In *International Conference on Machine Learning*, pages 18250–18280. PMLR, 2022.

[407] Yan Sun, Li Shen, Tiansheng Huang, Liang Ding, and Dacheng Tao. Fedspeed: Larger local interval, less communication round, and higher generalization accuracy. *International Conference on Learning Representations*, 2023.

[408] Rong Dai, Xun Yang, Yan Sun, Li Shen, Xinmei Tian, Meng Wang, and Yongdong Zhang. Fedgamma: Federated learning with global sharpness-aware minimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[409] Yifan Shi, Yingqi Liu, Kang Wei, Li Shen, Xueqian Wang, and Dacheng Tao. Make landscape flatter in differentially private federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24552–24562, 2023.

[410] Yifan Shi, Li Shen, Kang Wei, Yan Sun, Bo Yuan, Xueqian Wang, and Dacheng Tao. Improving the model consistency of decentralized federated learning. In *International Conference on Machine Learning*, pages 31269–31291. PMLR, 2023.

[411] Xueying Zhang, Yun Li, Tianyu Gong, Zhe Wang, and Fuyan Wang. Fedksam*: A two-phase federated learning intrusion detection algorithm. In *2024 43rd Chinese Control Conference (CCC)*, pages 8827–8831, 2024.

[412] Hoang Phan, Lam Tran, Ngoc N Tran, Nhat Ho, Dinh Phung, and Trung Le. Improving multi-task learning via seeking task-based flat regions. *arXiv preprint arXiv:2211.13723*, 2022.

[413] Yan Sun, Li Shen, Shixiang Chen, Liang Ding, and Dacheng Tao. Dynamic regularized sharpness aware minimization in federated learning: Approaching global consistency and smooth landscape. *International Conference on Machine Learning*, 2023.

[414] Ming Jiang, Shi Chen, Jinhui Yang, and Qi Zhao. Fantastic answers and where to find them: Immersive question-directed visual attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2980–2989, 2020.

[415] 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.

[416] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv:2003.02133*, 2020.

[417] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th Association for Computing Machinery workshop on artificial intelligence and security*, pages 27–38, 2017.

[418] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020.

[419] Michael R Garey and David S Johnson. “strong”np-completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery*, 25(3):499–508, 1978.

[420] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[421] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6430–6439, 2019.

[422] Douglas Steinley. K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, 2006.

[423] Riccardo Zaccone, Andrea Rizzardi, Debora Caldarola, Marco Ciccone, and Barbara Caputo. Speeding up heterogeneous federated learning with sequentially trained superclients. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3376–3382. IEEE, 2022.

[424] The TensorFlow Federated Authors. Tensorflow federated stack overflow dataset, 2019.

[425] Shreyansh Jain and Koteswar Rao Jerripothula. Federated learning for commercial image sources. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6534–6543, 2023.

[426] Richard Bellman. Dynamic programming. *Science*, 153:34–37, 1966.

[427] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[428] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 2017.

[429] Xiaoling Xia, Cui Xu, and Bing Nan. Inception-v3 for flower classification. In *2nd international conference on image, vision and computing*, pages 783–787. IEEE, 2017.

[430] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1215–1224, 2021.

[431] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pages 3973–3983. PMLR, 2020.

[432] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4085–4095, 2020.

[433] Francois Fleuret et al. Uncertainty reduction for model adaptation in semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9613–9623, 2021.

[434] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3723–3732, 2018.

[435] 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.

[436] Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020.

[437] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

[438] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning. *arXiv preprint arXiv:2005.01026*, 2020.

[439] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 506–516, 2017.

[440] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

[441] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[442] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[443] Massimiliano Mancini, Elisa Ricci, Barbara Caputo, and Samuel Rota Bulo. Boosting binary masks for multi-domain learning through affine transformations. *Machine Vision and Applications*, 31(6):1–14, 2020.

[444] 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 *IROS*, 2021.

[445] 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 *IROS*, 2021.

[446] Timothy Ha, Gunmin Lee, Dohyeong Kim, and Songhwai Oh. Road graphical neural networks for autonomous roundabout driving. In *IROS*, 2021.

[447] Viswadeep Lebakula, Bo Tang, Christopher Goodin, and Cindy L. Bethel. Shape estimation of negative obstacles for autonomous navigation. In *IROS*, 2021.

[448] 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. *T-PAMI*, 40, 2017.

[449] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*, 34, 2021.

[450] Gabriel L Oliveira, Wolfram Burgard, and Thomas Brox. Efficient deep models for monocular road segmentation. In *IROS*, 2016.

[451] 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(11):3051–3068, 2021.

[452] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. End-to-end federated learning for autonomous driving vehicles. In *IJCNN*, 2021.

[453] 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)*, 2022.

[454] Yijing Li, Xiaofeng Tao, Xuefei Zhang, Junjie Liu, and Jin Xu. Privacy-preserved federated learning for autonomous driving. *IEEE TITS*, 2021.

[455] Jianzhong He, Xu Jia, Shuaijun Chen, and Jianzhuang Liu. Multi-source domain adaptation with collaborative learning for semantic segmentation. *CoRR*, 2021.

[456] Zhuoqun Liu, Fan Guo, Heng Liu, Xiaoyue Xiao, and Jin Tang. CMLocate: A cross-modal automatic visual geo-localization framework for a natural environment without gnss information. *IET Image Processing*, 17(12):3524–3540, 2023.

[457] Lauri Suomela, Jussi Kalliola, Harry Edelman, and Joni-Kristian Kämäräinen. Placenav: Topological navigation through place recognition. *arXiv preprint arXiv:2309.17260*, 2023.

[458] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, 2024.

[459] Mike Chatzidakis, Junye Chen, Oliver Chick, Eric Circlaeays, Sowmya Gopalan, Yusuf Goren, Kristine Guo, Michael Hesse, Omid Javidbakht, Vojta Jina, Kalu Kalu, Anil Katti, Albert Liu, Richard Low, Audra McMillan, Joey Meyer, Steve Myers, Alex Palmer, David Park, Gianni Parsa, Paul Pelzl, Rehan Rishi, Michael Scaria, Chiraag Sumanth, Kunal Talwar, Karl Tarbe, Shan Wang, and Mayank Yadav. Learning iconic scenes with differential privacy. *Expo Talk at International Conference on Machine Learning*, 2023.

[460] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking localization and mapping for augmented reality. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Eur. Conf. Comput. Vis.*, pages 686–704, Cham, 2022. Springer Nature Switzerland.

[461] Ruiqi Cheng, Kaiwei Wang, Jian Bai, and Zhijie Xu. Unifying visual localization and scene recognition for people with visual impairment. *IEEE Access*, 8:64284–64296, 2020.

[462] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[463] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4878–4888, June 2022.

[464] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Gsv-cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 513:194–203, 2022.

[465] Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11080–11090, October 2023.

[466] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Mixvpr: Feature mixing for visual place recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2998–3007, 2023.

[467] María Leyva-Vallina, Nicola Strisciuglio, and Nicolai Petkov. Data-efficient large scale place recognition with graded similarity supervision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 23487–23496, 2023.

[468] Sergio Izquierdo and Javier Civera. Optimal transport aggregation for visual place recognition, 2023.

[469] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(6):1437–1451, 2018.

[470] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Learned contextual feature reweighting for image geo-localization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3251–3260, 2017.

[471] Liu Liu, Hongdong Li, and Yuchao Dai. Stochastic attraction-repulsion embedding for large scale image localization, 2019.

[472] Yixiao Ge, Haibo Wang, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-supervising fine-grained region similarities for large-scale image localization. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 369–386, Cham, 2020. Springer International Publishing.

[473] Frederik Warburg, Søren Hauberg, Manuel López-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary street-level sequences: A dataset for lifelong place recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2623–2632, 2020.

[474] Eros Fanì, Marco Ciccone, and Barbara 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.

[475] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B. Letaief. Hierarchical federated learning with quantization: Convergence analysis and system design. *IEEE Transactions on Wireless Communications*, 22(1):2–18, 2023.

[476] Shunfeng Chu, Jun Li, Kang Wei, Yuwen Qian, Kunlun Wang, Feng Shu, and Wen Chen. Design of two-level incentive mechanisms for hierarchical federated learning, 2023.

[477] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark, 2023.

[478] Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How sharpness-aware minimization minimizes sharpness? In *The Eleventh International Conference on Learning Representations*, 2023.

[479] Kayhan Behdin and Rahul Mazumder. Sharpness-aware minimization: An implicit regularization perspective. *stat*, 1050:23, 2023.

[480] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[481] Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *International Conference on Machine Learning*, pages 11217–11227. PMLR, 2021.

[482] Thang Doan, Seyed Iman Mirzadeh, and Mehrdad Farajtabar. Continual learning beyond a single model. In *Conference on Lifelong Learning Agents*, pages 961–991. PMLR, 2023.

[483] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024.

[484] Jacob Mitchell Springer, Vaishnavh Nagarajan, and Aditi Raghunathan. Sharpness-aware minimization enhances feature quality via balanced learning. *International Conference on Learning Representations*, 2024.

[485] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[486] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[487] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. Fedjax: Federated learning simulation with jax. *arXiv preprint arXiv:2108.02117*, 2021.

[488] 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.

[489] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020.

[490] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[491] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[492] Debora Caldarola, Barbara Caputo, and Marco Ciccone. Window-based model averaging improves generalization in heterogeneous federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2263–2271, 2023.

[493] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. FedJAX: Federated learning simulation with JAX. *arXiv preprint arXiv:2108.02117*, 2021.

[494] Andrej Karpathy. mingpt. https://github.com/karpathy/minGPT, 2022.

[495] Wikimedia Foundation. Wikimedia downloads.

Appendix A

Implementation Details

A.1 Appendix for *Improving Generalization in Federated Learning by Seeking Flat Minima*

This section provides a detailed description of the datasets and models employed in this work. Additionally, information regarding the selected hyper-parameters and their fine-tuning schedule is provided. To enhance robustness and reliability, all results presented throughout the paper are averaged over the last 100 training rounds. Unless explicitly mentioned, the PyTorch framework [485] was used, and experiments were conducted on a single NVIDIA GeForce GTX 1070 GPU.

A.1.1 Models

CIFAR10 and CIFAR100

Dirichlet-based CIFAR. When using the Dirichlet-based version of the CIFAR datasets, the model of choice is a Convolutional Neural Network (CNN) similar to LeNet5 [112] on both datasets, following the setting of [278]. The network has two 64-channels convolutional layers with kernel of size 5×5, each followed by a 2×2 max-pooling layer, ended by two fully connected layers with 384 and 192 channels respectively and a linear classifier.

CIFAR-PAM. The chosen model is ResNet18, replacing Batch Normalization [115] layers with group normalization (GN) ones [116], as suggested by [486], with two groups for each GN layer. Experiments were run using `FedJAX` [487] on a cluster with NVIDIA V100 GPUs.

Landmarks-User-160k

Similarly to [278], MobileNetV2 [415] pre-trained on ImageNet [488] is used for training on LANDMARKS-USER-160K, with GN layers in place of BN. Since no details on the model are available, the network feature multiplier is set to $\alpha = 1$ and the number of groups of the GN layers to 8. No bottleneck layer before the classifier is applied, as specified in [278]. To reduce training time, the code is developed using `Flax` [489] for both pre-training and centralized baselines, and `FedJAX` [487] for the implementation of the federated algorithms. Both libraries are based on `JAX` [490] and allow for efficient data parallelization. Implementation of the MobileNetV2 backbone used for all the experiments is available here[1]. All large-scale classification experiments have been performed using an NVIDIA DGX A100 40GB.

The model trained on ImageNet reaches $\approx 68\%$ top-1 accuracy on the validation set, due to the presence of GN layers, which tend to perform slightly worse than BN when trained on ImageNet, and reduced hyperparemters tuning. The pre-training on ImageNet was run on 8 GPUs with a total batch size of 2048 images.

Cityscapes and IDDA

As proposed by the authors of FedDrive, the lightweight BiSeNetv2 [451] is employed for training, accounting for possible lower computational capabilities of the edge devices.

A.1.2 Hyper-parameters Tuning

Each dataset comes with its own hyper-parameters setup. The final choices of training hyper-parameters are summarized in Table A.1. Table A.2 and A.3 respectively show the values used for SAM/ASAM and SWA.

[1]https://github.com/rwightman/efficientnet-jax/tree/a65811fbf63cb90b9ad0724792040ce93b749303

Table A.1 Best performing training parameters

Dataset	η_l	**Batch size**	**Weight decay**	**Epochs**	**Client momentum**	**Rounds**	**Clients per round**
CIFAR10	0.01	64	$4 \cdot 10^{-4}$	1	0	$10k$	$\{5, 10, 20\}$
CIFAR100	0.01	64	$4 \cdot 10^{-4}$	1	0	$20k$	$\{5, 10, 20\}$
CIFAR100-PAM	0.01	20	$4 \cdot 10^{-4}$	1-2	0.9	$10k$	$\{10, 20\}$
LANDMARKS-USER-160K	0.1	64	$4 \cdot 10^{-5}$	5	0	$5k$	10
CITYSCAPES (unif.)	0.05	8	$5 \cdot 10^{-4}$	2	0.9	$1.5k$	5
CITYSCAPES (het.)	0.05	8	$5 \cdot 10^{-4}$	2	0.9	$1.5k$	5
IDDA (country)	0.1	8	0	2	0.9	$1.5k$	5
IDDA (rainy)	0.1	8	0	2	0.9	$1.5k$	5

CIFAR10 and CIFAR100

For both datasets, the training hyper-parameters follow the choice of [278]. The client learning rate η_l is tuned between the values $\{0.01, 0.1\}$ and set to 0.01, the batch size is 64, $E \in \{1, 2\}$ is tested for the number of local epochs and the former is chosen. As for the weight decay the value $4 \cdot 10^{-4}$ leads to better performances than 0. The local optimizer is SGD with no momentum. No learning rate scheduler is used for simplicity. The local loss function is the cross-entropy loss. As for the server-side, when testing FEDAVGM, the server-side momentum $\beta = 0.9$. The training proceeds for $10k$ rounds on CIFAR10 and $20k$ rounds on CIFAR100.

Mixup and Cutout. Following the setup of [181], $\alpha_{\text{mixup}}= 1$, resulting in λ uniformly distributed between 0 and 1. As for Cutout instead, the cutout size is 16×16 pixels for CIFAR10 and 8×8 for CIFAR100, as proposed by [180].

SAM and ASAM. The parameter ρ of SAM is searched in $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$. As for ASAM, the value of ρ is tuned in $\{0.05, 0.1, 0.2, 0.5, 0.7, 1.0, 2.0\}$ and $\eta \in \{0.0, 0.01, 0.1, 0.2\}$. The choices made for each dataset and α are shown in Table A.2. There is no distinction of values as clients vary per round.

SWA. As shown in Section 4.3.5, SWA's starting round is tested in $\{5\%, 25\%, 50\%, 75\%\}$ of the rounds budget and, as expected [237], the best contribution is given if applied from 75% of the training onwards. The value of the learning rate γ_1 is set to 0.01 and $\gamma_2 \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, selecting $\gamma_2 = 10^{-4}$. The cycle

Table A.2 FEDSAM and FEDASAM hyper-parameters

Dataset	Distribution	SAM	ASAM	
		ρ	ρ	η
	α= 0	0.1	0.7	0.2
CIFAR10	α= 0.05	0.1	0.7	0.2
	α= 100	0.02	0.05	0.2
	α= 0	0.02	0.5	0.2
CIFAR100	α= 0.5	0.05	0.5	0.2
	α= 1000	0.05	0.5	0.2
CIFAR100-PAM	α= 0.1	0.05	0.5	0/0.2
LANDMARKS-USER-160K	-	0.05	0.5	0/0.2
CITYSCAPES	het/unif	0.01	0.1	0.2
IDDA	het/unif	0.01	0.5	0.2

length c is tested in $\{5, 10, 20\}$ and set to 10 for CIFAR10 and 20 for CIFAR100. Table A.3 summarizes the choices.

CIFAR100-PAM

The hyper-parameters follow the same choice of [275] (see Table A.1). Accuracies are reported at $5K$ and $10K$ communication rounds.

Mixup and Cutout. Same as CIFAR100.

SAM and ASAM. The hyper-parameters search replicates the on used for the Dirichlet-based CIFAR100. The best values of ρ for SAM and ASAM in all configurations are 0.05 and 0.5, respectively. For ASAM, $\eta = 0.2$ is the best choice when cutout or no augmentations are applied, while $\eta = 0$ works best in the case of Mixup.

SWA. Same as CIFAR100.

Landmarks-User-160k

In contrast with the setting proposed by the original paper [278], here, FEDAVGM with momentum $\beta = 0.9$ is unstable with 10 participating clients and requires reducing the server learning rate to 0.1 to train the model. Better performance and faster convergence can be obtained with 50 clients per round and $\beta = 0.9$. However, to maintain consistency with other experiments and deal with limited resources, 10 clients per round are selected. All hyper-parameters are described in Table A.1.

SAM/ASAM. The parameter ρ of SAM is searched in $\{0.01, 0.05, 0.1\}$. As for ASAM, the value of ρ is tuned in $\{0.1, 0.3, 0.5\}$ and $\eta \in \{0.0, 0.1, 0.2\}$.

SWA. SWA starting round is tested at both the 75% and 100% of training, *i.e.* the 3750-*th* and 5000-*th* rounds. The cycle lenghts are tested as $c \in \{5, 10, 20\}$ and learning rate $\gamma_2 \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. The best performing learning rates (γ_1, γ_2) are respectively $(10^{-1}, 10^{-3})$ and the cycle length is 5.

Cityscapes and IDDA

For both Cityscapes and IDDA, the choice of hyper-parameters follows [271]. The clients' initial learning rate is 0.05 on Cityscapes and 0.1 on IDDA, the weight decay is $5 \cdot 10^{-4}$ on Cityscapes, while it is not used on IDDA, 2 local epochs, the client optimizer is SGD with momentum 0.9. Differently from [271], here mixed precision is not used, thus the batch size is reduced from 16 to 8. A polynomial learning rate scheduler is applied locally, following [451]. The optimization is based on the Online Hard-Negative Mining [491], which selects the 25% of the pixels having the highest cross-entropy loss. The training is spanned across $1.5k$ rounds.

SAM and ASAM. The parameter ρ of SAM is searched in $\{0.01, 0.05, 0.1\}$. As for ASAM, the value of ρ is tuned in the set $\{0.05, 0.1, 0.5\}$ and $\eta \in \{0.0, 0.1, 0.2\}$.

SWA. Following the setup established for the CIFAR datasets, SWA starts at the 75% of training, *i.e.* the $1125th$ round. The learning rates (γ_1, γ_2) are respectively $(10^{-1}, 10^{-3})$ for IDDA and $(5 \cdot 10^{-2}, 5 \cdot 10^{-4})$ for Cityscapes. The cycle length is 5 for both datasets.

Table A.3 SWA hyper-parameters

Dataset	c	γ_1	γ_2	Start round
CIFAR10	10	10^{-2}	10^{-4}	7500
CIFAR100	20	10^{-2}	10^{-4}	15000
CIFAR100-PAM	5	10^{-2}	10^{-4}	15000
LANDMARKS-USER-160K	5	10^{-1}	10^{-3}	3750/5000
CITYSCAPES	5	$5 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	1125
IDDA	5	10^{-1}	10^{-3}	1125

A.2 Appendix for *Beyond Local Sharpness: Communication-Efficient Global Sharpness-aware Minimization for Federated Learning*

The model setup follows the one described in Section A.1.1.

A.2.1 Hyper-parameters Tuning

Table A.4 reports the training hyper-parameters associated to each dataset and model pairing. Table A.5 instead summarizes the hyper-parameters search grid tested for each method (in bold the chosen ones). All runs are averaged over 3 seeds. In addition, following previous works [288, 7, 492], each result is the averaged accuracy of the last 100 rounds to reduce the noise typical of heterogeneous FL settings.

Experiments with FEDGLOSS revealed that a larger local ρ value led to higher final accuracy, while a smaller ρ facilitated faster convergence during the initial training stages. Based on this observation, the local ρ value is scheduled to change over the first T_s training rounds as follows:

$$\rho(t) = \begin{cases} \rho_0 + {}^{(\rho-\rho_0)}/_{T_s} \cdot t & \text{if } t \leq T_s \\ \rho & \text{otherwise,} \end{cases}$$

starting from $\rho_0 = 0.001$.

Table A.4 General training hyper-parameters common to all methods, distinguished by dataset and model architecture. Symbols: local epochs E, local learning rate η, weight decay wd, client-side momentum β_l, batch size B.

Dataset	**Model**	**Rounds**	**Clients per round**	**Client optimization**				
				E	η	wd	β_l	B
CIFAR10	CNN	10000	5	1	10^{-2}	$4 \cdot 10^{-4}$	0	64
CIFAR100	CNN	20000	5	1	10^{-2}	$4 \cdot 10^{-4}$	0	64
	ResNet18GN	10000	10	1	10^{-2}	10^{-5}	0.7	64
LANDMARKS-USER-160K	MobileNetv2	1300	50	5	0.1	$4 \cdot 10^{-5}$	0	64

Table A.5 Search grid used to find optimal hyper-parameters for each combination of method, dataset and model. Best performing values in **bold**.

Method	**HParam**	**CIFAR10**		**CIFAR100**		**LANDMARKS-USER-160K**
		CNN	ResNet18	CNN	ResNet18	
FEDSAM	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**, 0.02, 0.05]	[0.005, **0.01**, 0.02, 0.05]	[**0.01**, 0.02, 0.05]	[**0.05**]
FEDPROX	μ	[0.001, 0.01, **0.1**]	[0.001, **0.01**, 0.1]	[0.001, 0.01, **0.1**]	[0.001, 0.01, **0.1**]	[0.001, 0.01, **0.1**]
FEDDYN	α	[0.001, **0.01**, 0.1]	[0.001, **0.01**, 0.1]	[0.001, **0.01**, 0.1]	[**0.01**]	[**0.001**, 0.01]
	ρ (SAM-based only)	[**0.15**]	[**0.01**]	[0.01, **0.02**]	[**0.01**]	[**0.05**]
FEDSPEED	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.005, **0.01**, 0.02, 0.05]	[**0.01**]	[**0.05**]
	α	[0.9, **0.95**, 0.99]	[0.9, **0.95**]	[0.9, **0.95**, 0.99]	[0.9, **0.95**]	[**0.95**]
	λ	[10, **100**, 1000]	[10, 100, **1000**]	[10, 100, **1000**]	[10, 100, **1000**]	[100, **1000**]
FEDGAMMA	ρ	[**0.15**]	[**0.01**]	[**0.01**]	[**0.01**]	[**0.05**]
FEDSMOO	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.005, 0.01, 0.05, **0.1**, 0.2]	[**0.01**]	[0.05, **0.1**, 0.2, 0.3]
	β	[5, **10**, 100]	[1, 2, **5**, 10]	[10, **100**]	[5, **10**, 100]	[10, **50**, 100, 1000]
FEDGLOSS (ours)	ρ_s	[0.01, 0.1, **0.15**]	[**0.01**, 0.05, 0.1, 0.5]	[**0.01**, 0.05, 0.1, 0.2]	[**0.01**, 0.05, 0.1, 0.5]	[**0.005**, 0.01, 0.02]
	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.05, 0.1, **0.2**]	[**0.01**]	[0.05, 0.1, 0.2, **0.3**]
	β	[5, **10**, 100]	[**1**, 2, 5, 10]	[10, **100**]	[**5**, 10, 100]	[10, **50**, 100]
	T_s	[1000, **2000**, 4000]	[**0**]	[1000, 2000, 5000, 10000, **15000**]	[**0**]	[**0**]

A.3 Appendix for *Window-based Model Averaging Improves Generalization in Heterogeneous Federated Learning*

This section provides the implementation details for the experiments using WIMA (Window-based Model Averaging), introduced in Section 4.5.

Large-scale experiments were performed using an NVIDIA DGX A100, while the others run on one NVIDIA GeForce GTX 1070. The code was built starting from the FedJAX framework [493]. All runs are averaged over 3 seeds.

A.3.1 Training Details

On the server side, the standard FedAvg with $\eta_g = 1$ without momentum is the aggregation algorithm of choice, unless otherwise specified. The clients locally train with SGD.

Experiments on the Dirichlet's CIFAR datasets are run for $10k$ rounds, selecting 10 clients at each round, *i.e*. with 10% participation rate. In the local training, the learning rate is 0.1 from $\{0.1, 0.01\}$, momentum 0 from $\{0, 0.9\}$, weight decay 0 unless otherwise specified, batch size 100 among $\{32, 64, 100, 128\}$, and the trainining is run for 1 local epoch chosen from $\{1, 2, 4\}$

For CIFAR100/PAM, $T = 10k$ rounds with 20% participation rate, using learning rate 0.05, weight decay $4e$-4, batch size 20, server-side momentum 0.9 from [7].

For FEMNIST the client learning rate is 0.1 from $\{0.1, 0.01, 0.001\}$, momentum 0 from $\{0, 0.9\}$, weight decay 0, batch size 10 from $\{10, 20, 32\}$. $T = 1,500$ rounds with 10 clients per round ($\approx$ 0.3% participation rate), performing 1 local epoch each.

GLDV2 follows the setup of [7] except for the batch size equal to 50. The model is trained for $3k$ rounds with 10 clients selected at the time.

In SHAKESPEARE, local learning rate is 1, momentum 0, weight decay 0, batch size 4, 1 epoch from [282]. Training is spanned over $1,500$ rounds with 10 clients per round ($\approx$ 1.4% participation rate).

The WIMA parameter W is set to 100 for all settings except for GLDV2, where $W = 370$.

For all datasets, the reported final results are averaged over the last 100 rounds for increased robustness [7].

A.3.2 SOTA Algorithms Hyper-parameters

This section provides details on the tuning intervals for the state-of-the-art (SOTA) algorithms used for comparison.

WIMA is applied on top of methods proposed for addressing statistical heterogeneity in FL. Looking at momentum-based approaches, the selected algorithms are FEDAVGM [290] (server-side momentum $\beta = 0.9$, $\eta_g \in \{0.1, 1\}$), MIME SGD ($\eta_g \in \{0.1, 1\}$) and SGDM [282], *i.e*. with momentum 0.9, MIMELITE SGDM [282] ($\eta_g \in \{0.1, 1\}$, momentum 0.9), FEDCM [291] ($\alpha_{\text{CM}} \in \{0.05, 0.1, 0.5\}$) and FEDACG [292] ($\beta_{\text{ACG}} \in \{0.01, 0.001\}$, $\lambda_{\text{ACG}} \in \{0.8, 0.85, 0.9\}$). Other methods are SCAFFOLD [280], FEDPROX [45] ($\mu_{\text{PROX}} \in \{0.1, 0.01, 0.001\}$), FEDDYN [288] ($\alpha_{\text{DYN}} \in \{0.01, 0.001\}$) and ADABEST [8] ($\mu_{\text{ADABEST}} \in \{0.01, 0.02\}$, $\beta_{\text{ADABEST}} \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$) to reduce the client drift.

Lastly, WIMA is compared with SWA applied from 75% of training onward, for which $c \in \{10, 20\}$ and the second learning rate is equal to $\eta_l \cdot 10^{-2}$, following [7].

A.4 Appendix for *Accelerating Federated Learning via Sequential Training of Grouped Heterogeneous Clients*

This section provides the implementation details for the experiments presented in Section 5.2. The used framework is PyTorch [485], and all the experiments were conducted on one NVIDIA GeForce GTX 1070 with 3 different seeds. As proposed by previous works [7, 288] accounting for the learning trends instability, the results are averaged over the last 100 rounds.

A.4.1 Datasets and Models

CIFAR100 and CIFAR10. This work utilizes the widely used CIFAR datasets (CIFAR10 and CIFAR100) as benchmarks for image classification tasks in FL. These datasets consist of 10 classes (CIFAR10) and 100 classes (CIFAR100), respectively. Following the protocol established in [290], the class distribution for each client is sampled from a Dirichlet distribution with varying concentration parameters $\alpha \in \{0, 0.2, 0.5\}$. This, combined with the number of clients ($K = 500$), creates a realistic scenario where clients have small and unbalanced datasets, reflecting the inherent statistical heterogeneity in FL.

A Convolutional Neural Network (CNN) similar to LeNet-5 is employed, following the setup in [278, 7]. The network architecture consists of two convolutional layers, each with 64 channels and a 5x5 kernel size. Each convolutional layer is followed by a 2x2 max-pooling layer. These layers are then connected to two fully-connected (FC) layers with 284 and 192 channels, respectively. Finally, a linear classifier with a size dependent on the number of classes (10 for CIFAR10 and 100 for CIFAR100) is used for output.

FEMNIST. The FEMNIST dataset [394] is a federated version of the EMNIST dataset and consists of images containing 62 different digits and uppercase/lowercase English letters. The data is distributed among 3,500 clients based on the author's identity, resulting in a naturally heterogeneous dataset (NIID split) due to variations in handwriting styles. Additionally, the number of training examples and covered letters varies between authors, further increasing the heterogeneity. An IID split proposed in [394] is also adopted, where each of the 3,500 clients has a balanced local dataset.

The CNN architecture used in [275] is employed for training. The network consists of two convolutional layers with 32 and 64 channels, respectively. These layers are followed by a 2x2 max-pooling layer and two dropout layers with probabilities of 0.25 and 0.5. A fully-connected layer with 128 channels is placed between the dropout layers. Finally, a linear classifier with 62 channels (corresponding to the number of possible characters) is used for output.

SHAKESPEARE. The SHAKESPEARE language modeling dataset, built from the works of William Shakespeare, is included for next-character prediction tasks as described in [394]. Each client represents a Shakespearean character and has access only to their lines from the corresponding play. Non-*i.i.d.* (NIID) and *i.i.d.* splits are adopted following the approach in [394]. The experiment is limited to 100 clients, each with 2k training examples, as established in [288]. The inherent heterogeneity in the NIID split arises from the distinct speaking styles of various characters.

A Long Short-Term Memory (LSTM) network architecture is employed for training, following the setup in [288]. The network first transforms an 80-character sequence into an 80x8 matrix using learned embeddings. This matrix is then fed into a two-layer LSTM with a hidden size of 100. The output is passed to a fully-connected layer with 90 channels (corresponding to the 86 characters in the Shakespeare dataset along with 4 special characters for padding, out-of-vocabulary words, and beginning/end of line markers).

As for the data preprocessing, sequences of 80 characters are selected, and the task is to predict the next 80 characters. This is achieved by shifting the input sequence by one position (treating the last character as the predicted output) and inserting special characters when necessary.

Table A.6 Best performing training hyperparameters on FEDSEQ

Dataset	**Client lr**	**Batch size**	**Weight decay**	**Server lr**
CIFAR10	0.01	64	$4 \cdot 10^{-4}$	1
CIFAR100	0.01	64	$4 \cdot 10^{-4}$	1
FEMNIST	0.01	20	0	1
SHAKESPEARE	1	100	0	1
STACKOVERFLOW	$10^{-1/2}$	16	0	1

STACKOVERFLOW. The STACKOVERFLOW dataset [424], sourced from the StackOverflow website, consists of questions and answers. Each client holds a maximum of 1,000 sentences from a single user, leading to heterogeneity due to varying user typing styles. This dataset presents the most challenging scenario in terms of the number of devices ($K = 40,000$) used in the experiments. To balance efficiency with a reasonable accuracy estimate, test examples are restricted to 10,000 random samples per round. The final test accuracy is obtained using the entire test dataset.

A LSTM network architecture is employed for training, following the setup in [275]. The network processes sequences of 20 characters. These sequences are first converted into a 20x96 matrix using a learned embedding layer. The resulting matrix is then fed into a single-layer LSTM with 670 channels. Finally, the output is passed through two fully-connected layers with channel sizes of 96 and $10,004$, respectively. The larger layer size corresponds to the number of characters in the vocabulary (excluding special tokens) plus four special tokens for padding, out-of-vocabulary words, and end-of-sentence markers.

Regarding the data preprocessing, each client's dataset is limited to a maximum of $1,000$ sentences. Sentences are further truncated to contain at most 20 words. Special tokens for padding and out-of-vocabulary words are inserted when necessary. Finally, each word is converted to its corresponding index based on the $10,000$ most frequent words in the dataset. Words not found in this vocabulary are replaced with the special out-of-vocabulary token.

A.4.2 Hyper-parameters Tuning

Table A.6 summarizes the chosen hyper-parameters for each dataset. For all methods, local training happens for one epoch with SGD (without momentum and learning rate scheduler) and in FEDSEQ $E_S = E_k = 1$. Cosine annealing is used in the centralized experiments.

CIFAR datasets. Following the hyper-parameters choices of [278] on both datasets, the client learning rate is set to 0.01, the weight decay to $4 \cdot 10^{-4}$, momentum 0 and batch size 64. We run the experiments for $T = 10k$ rounds on CIFAR10 and $T = 20k$ on CIFAR100, accounting for the tasks' difficulty. For the centralized scenario we also add a momentum of 0.9, training for 300 epochs. In FEDSEQ and FEDASYNCSEQ, ψ_{t2v} is the distribution estimator and ϕ_{greedy} the grouping method, with $|\mathscr{D}_S|_{min} = 1000$ and $K_{S,max} = 11$. In FEDSEQ2PAR, ψ_{t2v} and ϕ_{icg} are chosen, with f_{exp}, $\alpha_{gr}= 4.6 \cdot 10^{-4}$ and $\beta_{gr} = 5$. For ψ_{t2v}, ResNet18 pre-trained on ImageNet is used as in [421].

As for the SOTA algorithms, FEDPROX is tested with $\mu \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and choose $\mu = 0.01$. In FEDDYN $\alpha_{dyn}= 0.1$ is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}\}$. To ensure convergence in the most heterogeneous scenarios, the gradient is clipped to 30.

FEMNIST. Client learning rate is tested in $\{0.05, 0.01, 0.005, 0.001\}$ and the weight decay in $\{0, 1 \cdot 10^{-4}\}$, choosing respectively 0.01 and 0, with momentum 0 and batch size 20.The federated experiments are run for 1500 rounds.

The best performing parameters for the clustering methods in FEDSEQ and FEDASYNCSEQ are ψ_{t2v} and ϕ_{kmeans}, with $|\mathscr{D}_S|_{min} = 4120$ and $K_{S,max} = 21$, while FEDSEQ2PAR uses ψ_{t2v} and ϕ_{icg}, with f_{exp}, $\alpha_{gr}= 3 \cdot 10^{-3}$ and $\beta_{gr} = 5$.

FedProx's μ is tested between $\{10^{-4}, 10^{-3}, 10^{-2}\}$ with 10^{-3} being the best performing, while for FedDyn $\alpha_{dyn}\in \{10^{-3}, 10^{-2}, 1.5 \cdot 10^{-2}\}$ and choose 10^{-3}, with gradient clipping of 30 because of convergence issues.

SHAKESPEARE. Shakespeare's hyper-parameters follow the implementation of [288] : the client's learning rate is set to 1, with weight decay 10^{-4} to prevent overfitting. Momentum is 0 and the batch size is 100. $T = 250$ in FL and $E = 75$ in the centralized setting.

FEDSEQ and FEDASYNCSEQ use ψ_{t2v} and ϕ_{greedy} with $|\mathscr{D}_S|_{min} = 8000$ and $K_{S,max} = 5$, while for FEDSEQ2PAR the best performing hyper-parameters are f_{exp},

α_{gr}= $1.8 \cdot 10^{-2}$ and $\beta_{gr} = 5$. GPT-2 from [494] is used as pre-trained model for ψ_{t2v}. Since there was no character-level language model that was not already trained on the Shakespeare dataset, charGPT was trained for $100k$ epochs with learning rate $5 \cdot 10^{-4}$ on cleaned articles from English Wikipedia [495], following [494]'s implementation. The model is made of 6 hidden layers, 6 attention heads (referred to as 'gpt-mini') [2]

For FEDPROX, $\mu = 10^{-3}$ from $\{10^{-4}, 10^{-3}, 10^{-2}\}$, and α_{dyn}= 10^{-3} for FEDDYN from $\{10^{-3}, 10^{-2}, 1.5 \cdot 10^{-2}\}$.

A.5 Appendix for *Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning*

Table A.7 LADD datasets training hyper-parameters

Dataset	T	K^t	η	λ_{KD}	ω	t_{START}
CrossCity	1000	4	$1.0 \cdot 10^{-2}$	20	1	400
Cityscapes	300	5	$5e-5$	10	5	200
Mapillary	100	$1.0 \cdot 10^{-2}$	6	5	50	

A.5.1 Training Details

Server Pre-Training. The model was pre-trained on the GTA5 dataset using a power-law decreasing learning rate schedule with a starting value of $\eta = 5 \cdot 10^{-3}$ and a power of 0.9. SGD optimizer with momentum of 0.9 and no weight decay was employed. The pre-training phase lasted for $N_{\text{pre-train}} = 15k$ steps. Each client computes the style on all its images using a window of size 3×3 and sends the mean style to the server, before the pre-training starts.

Federated Adaptation. Experiments were conducted on the CrossCity, Cityscapes, and Mapillary datasets. Details on the hyper-parameters can be found in Table A.7

[2]Model available at https://huggingface.co/andrea-silvi/charGPT_pretrained.

Data augmentation techniques including random scaling $(0.7, 2)$, random cropping at 1024×512, color jittering with brightness, contrast and saturation equal to 0.5, and image normalization were applied. Rescaling was forced to width equal to 1024.

Appendix B

Project Funding and Computational Resources

This thesis has been partially funded by the ERC projects 853489-DEXIM, 802554-SPECGEO, 637076-RoboExNovo and the MIUR grant "*Dipartimenti di eccellenza 2018-2022*". It was supported by *Consorzio Interuniversitario Nazionale per l'Informatica* (CINI), the FAIR — Future Artificial Intelligence Research and European Union (EU) Next-GenerationEU and the Blanceflor Foundation (travel fund).

Computational resources were partly provided by CINECA (Iscra B "FedVis", Iscra C "Fed-PERS", Iscra C "FedUDAS") and Politecnico di Torino (HPC@PoliTo).