Sparsity in Federated Learning: A Survey

Alessio Mora[1] , **Adriano Guastella**[1] , **Lorenzo Sani**[2,3] , **Paolo Bellavista**[1] and **Nicholas D. Lane**[2,3]
[1]Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna
[2]Department of Computer Science and Technology, University of Cambridge
[3]Flower Labs, UK
{alessio.mora, adriano.guastella2, paolo.bellavista}@unibo.it, {ls985, ndl32}@cam.ac.uk

Abstract

Conventional Federated Learning (FL) pipelines focus on the collaborative training of a global dense model across client devices. Sparsity has been increasingly adopted in FL, during or after local optimization, for a range of objectives, including reducing communication and computation costs, supporting unlearning, enhancing privacy guarantees, and improving local personalization. In this survey, we introduce a novel taxonomy of sparse FL methods that systematically organizes the existing literature according to their core objectives and methodological choices. Using this taxonomy, we analyze and categorize prior work, highlighting the underlying intuitions, technical mechanisms, benefits, and limitations of each class of approaches. Finally, we identify open challenges, expose research gaps, and extract guidance to help practitioners understand and adopt sparsity mechanisms in FL.

1 Introduction

Federated Learning (FL) has emerged as a standard paradigm for distributed training on private data, particularly in settings involving edge devices [McMahan *et al.*, 2017]. Beyond controlled academic benchmarks, FL is increasingly used in real-world deployments involving mobile devices, multi-center medical studies, and industrial IoT systems, where models are trained across heterogeneous client populations while maintaining data locality [You *et al.*, 2025; Hanser *et al.*, 2025]. These deployment scenarios introduce practical constraints that distinguish FL from centralized learning, including resource-constrained devices, heterogeneous data distributions, privacy requirements, and personalization–generalization trade-offs [Kairouz *et al.*, 2021].

Sparsity has long been studied in centralized deep learning [Hoefler *et al.*, 2021; Cheng *et al.*, 2024] as a means to reduce parameter redundancy, memory footprint, and computational cost, typically through pruning, structured sparsity, or sparse architectural design. These ideas have naturally been extended to the federated setting, initially with the goal of reducing communication overhead by transmitting sparse updates or compressed models. However, the role of sparsity in FL extends beyond compression.

Unlike centralized training, sparsity in FL must be managed across a federation of autonomous clients that may observe heterogeneous data, participate intermittently, and evolve their local models independently before aggregation. As a result, sparsity introduces new design challenges that are inherently federated: *how sparse structures are coordinated across clients, how sparsity patterns interact with client drift, how sparse models or updates are aggregated without destroying sparsity properties, and how consistency is maintained across communication rounds*. In this sense, sparsity in FL is not merely a local optimization choice, but a system-level design dimension.

Driven by these system-level considerations, recent work has explored sparsity in FL for objectives beyond communication efficiency [Sattler *et al.*, 2019]. Sparse mechanisms have been leveraged to reduce local computational requirements at resource-constrained devices [Guastella *et al.*, 2025], to enhance privacy by restricting information flow through selective parameter updates or constrained representations [Shen *et al.*, 2025], to enable federated unlearning by identifying and pruning parameter subsets associated with specific data contributions [Zhong *et al.*, 2025], and to support personalization by adapting sparse subnetworks to local data while preserving a shared global structure [Bai *et al.*, 2025]. In this light, sparsity has emerged as a versatile mechanism to address core FL challenges across efficiency, privacy, unlearning, and personalization.

Contribution and related surveys. While substantial progress has been made, the literature on sparse FL remains fragmented. Existing sparsity surveys focus on centralized deep learning [Hoefler *et al.*, 2021; Cheng *et al.*, 2024], while general FL surveys [Woisetschläger *et al.*, 2024] typically treat sparsity only as an efficiency mechanism, without addressing its broader roles or federated-specific challenges. To narrow this gap, our contributions are threefold:

- We introduce a **federated-specific taxonomy** for sparse FL methods that organizes existing approaches along dimensions intrinsic to distributed and privacy-preserving training, including the motivation for sparsity, the object being pruned, the mechanism used to induce sparsity, the stage of the FL workflow at which sparsity is applied, the degree of coordination across clients, and the server-side aggregation strategy.

- We systematically **categorize and analyze** prior work using this taxonomy, distilling common design principles, dominant patterns, and technical limitations.
- We **identify underexplored areas** and promising research directions for sparse FL, and provide **practitioner-oriented guidance** for selecting techniques under real-world deployment constraints.

Organization. Section 2 introduces preliminaries and the taxonomy; Section 3 reviews sparse FL methods by motivation; Section 4 outlines current trends and open issues; Section 5 distills practical guidance for selecting sparse FL techniques; Section 6 concludes.

2 Preliminaries and Taxonomy

This section formalizes the notion of sparsity used throughout the proposed taxonomy and introduces the basics of FL. These preliminaries establish a common vocabulary for classifying existing sparse FL methods.

2.1 Sparsity Primitives

We first formalize the notion of sparsity independently of any specific learning paradigm. Let $x \in \mathbb{R}^d$ denote a generic d-dimensional vector. We say that x is *sparse* if a large fraction of its entries are zero.

Sparsity mask. Sparsity is commonly represented through the application of a binary mask $m \in \{0,1\}^d$. The masked version of x is defined as

$$x^{(m)} = m \odot x,$$

where $\odot$ denotes element-wise multiplication.

The support of the mask is given by $\mathrm{supp}(m) = \{i \mid m_i = 1\}$. The *density* of the mask is defined as

$$\rho(m) = \frac{\|m\|_0}{d},$$

where $\|m\|_0$ denotes the number of non-zero entries in m. The corresponding *sparsity level* is $s(m) = 1 - \rho(m)$, often reported as a percentage (e.g., 90% sparse). Equivalently, sparsity can be interpreted as an explicit constraint on the number of active (non-zero) elements, i.e., $\|m\|_0 \leq \rho d$ for a target density $\rho \in (0,1]$.

In the remainder of this section, these definitions will be instantiated in the context of FL, where multiple d-dimensional vectors naturally arise.

2.2 Federated Learning

Let $\mathcal{K} = \{1, \ldots, K\}$ denote the set of clients and let $w \in \mathbb{R}^d$ represent the parameters of a shared global model maintained by the server [McMahan *et al.*, 2017]. The global model is iteratively trained across communication rounds. In conventional FL formulations, neither w nor the client-side updates derived from it are assumed to satisfy any sparsity constraint; all such vectors are treated as dense.

We explicitly describe the FL workflow in a step-wise manner, as these stages will serve as reference points for the taxonomy introduced later in the survey. A standard FL procedure, such as FedAvg [McMahan *et al.*, 2017], can be abstracted through the following sequence of per-round steps:

Step 0: Initialization (server side). The global model w^{init} is initialized once before federated training starts.
Step 1: Before local training (client side). At round t, the server broadcasts the current global model w^t to a subset of clients, which initialize their local models accordingly.
Step 2: Local training (client side). Each participating client $k \in S_t \subseteq \mathcal{K}$ performs local optimization on its private data, producing a locally updated model w_k^t. The *model update* generated by client k at round t is defined as $\Delta_k^t \triangleq w_k^t - w^t$.
Step 3: After local training (client side). Clients transmit either w_k^t or Δ_k^t to the server; before communication, local models or updates may optionally be transformed.
Step 4: Aggregation (server side). The server aggregates the received client updates to compute the next global model as $w^{t+1} = w^t + \sum_{k \in S_t} n_k \, \Delta_k^t$, where n_k are aggregation weights, typically proportional to local dataset sizes.

Steps 1-4 repeat until convergence.

2.3 Taxonomy

To systematically organize existing sparse FL methods, we introduce a taxonomy (visualized in Figure 1) that characterizes approaches along multiple complementary dimensions. The taxonomy answers the following questions: *why* sparsity is introduced, *what* is made sparse, *how* sparsity is technically realized, *when* and by *which entity* (client or server) it is applied, *how* sparsity patterns are coordinated or decided, and *how* sparse information is aggregated.

Challenge. We first classify methods according to their primary motivation for introducing sparsity: *(i)* efficiency (communication, computation), *(ii)* privacy enhancement, *(iii)* unlearning, and *(iv)* personalization.

Object of sparsity. This dimension captures *what* is made sparse: model parameters (w), model updates (Δ), activations (a), or learned gates (g). Gates are trainable scalar variables associated with structural units (e.g., neurons, channels) and are jointly optimized with model weights; units with low gate values are pruned.

Technique. This describes *how* sparsity is technically realized and is further decomposed into three components:

- **Sparsity regime:** describes how sparsity is applied during model training. We distinguish: (i) *one-shot pruning*, where the model is trained densely and sparsified in a single post-hoc step; (ii) *static sparse training*, where the model is pruned once and trained only on the remaining parameters, without weight regrowth; and (iii) *dynamic sparse training*, where training proceeds under a sparse mask that is updated online (e.g., through prune–grow cycles), allowing weights to regrow into previously pruned positions.
- **Pruning criterion:** common criteria to estimate importance of model parameters include: (i) *weight-based* (WEI), where importance depends only on the current parameter values; (ii) *gradient-based* (GRAD), where importance is derived from gradients or their statistics; (iii) *activation-based* (ACT), where scores are computed from feature activations or activation discrepan-

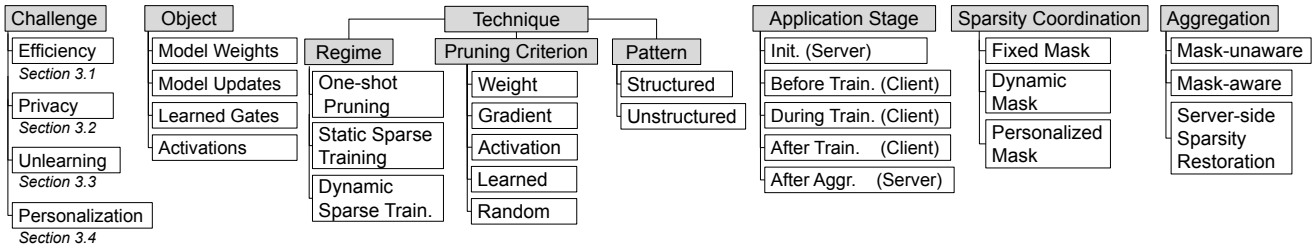

Figure 1: Taxonomy of sparse FL methods.

cies; (iv) *learned* (LRN), where learnable gates are optimized jointly with model weights and low-gate units are pruned; (v) *random* (RND), where sparsity patterns are sampled independently of model or data statistics.

- **Pattern:** we distinguish between *unstructured* sparsity (UNS), where individual parameters can be removed independently, and *structured* (STR) sparsity, where entire groups of parameters (e.g., channels, neurons, or blocks) are removed according to predefined patterns, typically enabling more practical computational speedups.

Application stage. This dimension specifies *when* and by *whom* sparsity is applied within the FL workflow. We map sparsity application to the stages defined in Section 2.2: server-side initialization (S-INIT), client-side before local training (C-BLT), client-side during local training (C-DLT), client-side after local training and before communication (C-ALT), and after server-side aggregation (S-AAG).

Sparsity coordination. This dimension describes *how* sparsity patterns or masks are synchronized across the federation. We distinguish: (i) *fixed global masks* (FIX), where a single mask is predefined (or initialized) and remains identical for all clients throughout training; (ii) *coordinated dynamic masks* (DYN), where masks are updated during training but periodically synchronized at the server; and (iii) *personalized masks* (PERS), where each client maintains its private mask with no synchronization.

Aggregation rule. Finally, we classify methods according to *how* sparse entities are aggregated at the server. We distinguish: (i) *mask-unaware*, where sparse updates or models are aggregated as dense vectors without accounting for sparsity patterns; (ii) *mask-aware aggregation*, where the server exploits client-side masks during averaging (e.g., averaging only over overlapping active parameters, or using union/intersection rules on supports); and (iii) *server-side sparsity restoration*, where the server performs standard FedAvg to obtain a dense intermediate model and subsequently reintroduces sparsity.

3 Sparsity in FL

This section presents the surveyed works organized by motivation, the first axis of our taxonomy (Section 2.3). Table 1 provides a compact characterization of the literature.

3.1 Efficiency-Oriented Sparsity

We first consider sparsity applied for communication and/or computational efficiency. Within this category, we group methods according to the *training regime* dimension of our taxonomy: *one-shot sparsity*, *static sparse training*, and *dynamic sparse training*. We dedicate the closing paragraph to reviewing strategies that prevent post-aggregation sparsity erosion (i.e., loss of sparsity after server aggregation).

One-shot sparsity. These methods train densely and sparsify after local training, reducing communication but not local floating-point operations (FLOPs).

A representative example is Sparse Ternary Compression (STC) [Sattler *et al.*, 2019], which applies unstructured Top-K magnitude pruning to client updates followed by ternary quantization of retained values, together with residual error accumulation to mitigate information loss. Structured variants such as FedSCR [Wu *et al.*, 2020] exploit the observation that Top-K supports often cluster within convolutional channels and filters, and therefore, prune structured groups rather than individual parameters. A recent line of work applies one-shot pruning in the context of federated large language models (LLM), with FedSpaLLM [Bai *et al.*, 2025] performing unstructured importance-based pruning to dense LLM weights before communicating them back to the server.

Static sparse training. *Static sparse training* determines a sparsity mask once ahead of local training, usually at initialization or after a short warm-up phase, and trains only the remaining parameters throughout the FL procedure. This reduces both computational cost and uplink communication cost (updates are inherently sparse).

Several works adopt this approach, often inspired by the Lottery Ticket Hypothesis (LTH) [Frankle and Carbin, 2019]. PruneFL [Jiang *et al.*, 2023a] performs initial pruning at a powerful client, and periodically reconfigures the global mask at the server based on aggregated importance statistics, but masks remain unchanged during local training phases. FLASH [Babakniya *et al.*, 2023] initializes a sensitivity-driven consensus mask on a subset of clients and supports two modes: *SPDST*, where clients perform static sparse training, and *JMWST*, where the server periodically restores the consensus mask after aggregation (dynamic training with server-side sparsity restoration). FedSNIP [Bustincio *et al.*, 2025] uses gradient-based importance scoring to prune networks at initialization without regrowth. BiPruneFL [Lee and Jang, 2025] also falls under static sparse training: it identifies sparse binary subnetworks at initialization using the multi-prize LTH, then trains them with binary operations.

Dynamic sparse training. *Dynamic sparse training* allows weights to regrow into previously pruned positions, typi-

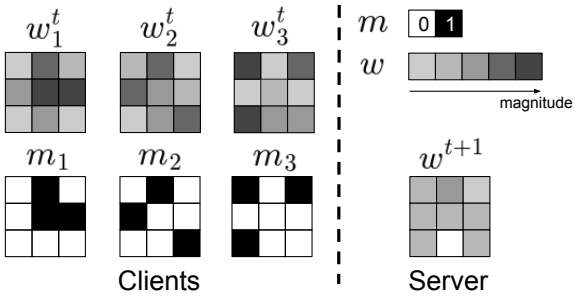

Figure 2: Illustration of sparsity erosion after aggregation. Each client applies the same sparsity level locally (3 active parameters out of 9), yet heterogeneous sparsity patterns yield a significantly denser aggregated model under FedAvg. For clarity, we use magnitude-based pruning in the illustrative examples, i.e., parameters with the smallest absolute values are set to zero.

cally through prune-grow cycles, prune-restore procedures, or structured participation decisions. This adaptivity helps accommodate heterogeneous client data and evolving model representations, but introduces challenges related to mask divergence and aggregation.

FedDST [Bibikar *et al.*, 2022] exemplifies this regime: clients prune and regrow weights during local training, while the server enforces a target per-layer sparsity after aggregation, coordinating mask evolution across rounds. SparsyFed [Guastella *et al.*, 2025] and ZeroFL [Qiu *et al.*, 2022] reduce FLOPs by enforcing sparse weights and sparse activations during training, although the resulting speedups may depend on sparse kernel support. NTK-BAFFLE [Zhang *et al.*, 2025] takes a more radical approach: it eliminates backpropagation and combines Neural Tangent Kernel-based foresight pruning with backpropagation-free FL, reducing memory and compute on constrained devices.

Preserving sparsity across aggregation. A fundamental challenge in FL with sparse communication arises from the decentralized and heterogeneous nature of client data. Even when clients employ the same sparsification mechanism, differences in local data distributions and participation lead to substantially misaligned sparsity patterns [Jiang *et al.*, 2023b]. As a result, FedAvg aggregation of locally sparse models or updates typically produces a denser global model, as illustrated in Figure 2. This loss of sparsity has two major consequences: (i) clients must repeatedly download dense global models, increasing downlink communication cost, and (ii) sparsity must be re-induced from scratch during each round of local training.

To mitigate this issue, existing methods can be grouped according to the *aggregation rule* dimension of our taxonomy, as presented below.

(i) Mask-unaware aggregation. Several works retain standard FedAvg without explicitly accounting for sparsity patterns. This choice mainly arises in two settings. First, when sparsity is used only to reduce uplink communication or to serve objectives beyond efficiency (discussed in 3.2, 3.3, and 3.4), the densification of the global model after aggregation is not a concern. Second, when clients share a consensus mask and perform static sparse training, sparsity is preserved by construction under FedAvg. Such shared masks can be initialized once and remain unchanged [Huang *et al.*, 2022], periodically reconfigured using server-side statistics as in PruneFL [Jiang *et al.*, 2023a] and FLASH [Babakniya *et al.*, 2023].

(ii) Mask-aware aggregation. A second class of methods modifies the aggregation operator to account for heterogeneous client masks. In FedDST [Bibikar *et al.*, 2022], each parameter is averaged only over clients for which it is active. Majority voting or union/intersection rules can also be used to form a global mask before averaging, as in [Stripelis *et al.*, 2022] and FedSpaLLM [Bai *et al.*, 2025]. These approaches prevent sparsity erosion by aligning or respecting client-side supports during aggregation.

(iii) Server-side sparsity restoration. A third family performs standard FedAvg to obtain a dense intermediate model and then restores sparsity on the server. A representative example is [Shah and Lau, 2021], which solves an ℓ_1-regularized reconstruction problem and then applies soft-thresholding. FedDST [Bibikar *et al.*, 2022] also includes a restoration step: after mask-aware aggregation, the server enforces a target per-layer sparsity by magnitude-based pruning, yielding a new global sparse model.

Other strategies. Other approaches promote implicit consensus without explicit masks: SparsyFed [Guastella *et al.*, 2025] aligns Top-K supports through sparsity-inducing local optimization, while [Isik *et al.*, 2023] treats masks themselves as the primary object of optimization and communication.

3.2 Enhanced Privacy

Privacy preservation is a central challenge in FL, as exchanged model updates can leak sensitive information about clients' private data. Unlike efficiency-oriented sparsity, which aims to reduce communication or computation, privacy-oriented sparsity targets inference attacks (e.g., gradient inversion, membership inference), often prioritizing information suppression or randomization over model compression [Shen *et al.*, 2025].

Sparsity-based privacy mechanisms. Theoretical analysis indicates that sparsity alone offers limited privacy protection [Chu *et al.*, 2025]. PriPrune [Chu *et al.*, 2025] shows that model updates remain vulnerable even at 90% sparsity, and proposes *pseudo-pruning*, where clients transmit heavily pruned updates while keeping dense local models, enabling privacy–utility trade-offs via information asymmetry. Building on this insight, FedPM [Isik *et al.*, 2023] leverages stochastic Bernoulli masks, where random sparsification yields privacy amplification under Rényi differential privacy [Mironov, 2017] (DP), and aggregation operates on probability masks rather than deterministic updates. MemDefense [Shen *et al.*, 2025] adopts a targeted strategy, pruning parameters most responsible for membership inference leakage while minimally affecting task accuracy, providing lightweight protection for constrained devices.

For stronger guarantees, sparsity is combined with cryptographic or robustness-oriented defenses. Beguier et al. [Beguier *et al.*, 2021] integrate top-K sparsification with additive secret sharing to enable efficient secure aggregation against malicious adversaries, while QuanCryptFL [Mia and Amini,

	Work	Object	Technique			Stage	Coordination	Aggregation
			Regime	Criterion	Pattern			
Efficiency	STC [Sattler *et al.*, 2019]	w, Δ	one-shot	WEI	UNS	C-ALT, S-AAG	PERS	mask-unaware
	FedSCR [Wu *et al.*, 2020]	Δ	one-shot	WEI	STR	C-ALT	PERS	mask-unaware
	FedSparsify [Stripelis *et al.*, 2022]	w	dynamic	WEI	UNS	C-ALT / S-AAG	DYN	mask-aware
	ZeroFL [Qiu *et al.*, 2022]	w, a	dynamic	WEI, ACT	UNS, STR	C-DLT, C-ALT	DYN	mask-unaware
	FedDST [Bibikar *et al.*, 2022]	w	dynamic	WEI	UNS	C-BLT, S-AAG	DYN	mask-aware, restoration
	PruneFL [Jiang *et al.*, 2023a]	w	dynamic	GRAD	UNS	S-INIT, S-AAG	DYN	mask-unaware
	FLASH–S [Babakniya *et al.*, 2023]	w	static	WEI	UNS	S-INIT, C-DLT	FIX	mask-unaware
	FLASH–J [Babakniya *et al.*, 2023]	w	dynamic	WEI	UNS	S-INIT, C-DLT, S-AAG	DYN	restoration
	FedSpaLLM [Bai *et al.*, 2025]	w	one-shot	WEI, GRAD	UNS	C-ALT, S-AAG	DYN	mask-aware
	SparsyFed [Guastella *et al.*, 2025]	w, a	dynamic	WEI	UNS	C-DLT, C-ALT	DYN	mask-unaware
	NTK-BAFFLE [Zhang *et al.*, 2025]	w	one-shot	LRN	UNS	S-INIT, C-DLT	FIX	mask-unaware
	FedSNIP [Bustincio *et al.*, 2025]	w	static	GRAD	UNS	S-INIT, C-DLT	FIX	mask-unaware
	BiPruneFL [Lee and Jang, 2025]	w	static	LRN	UNS	S-INIT, C-DLT	FIX	mask-unaware
Privacy	Beguier et al. [Beguier *et al.*, 2021]	Δ	one-shot	WEI	UNS	C-ALT	PERS	mask-aware
	SparseFed [Panda *et al.*, 2022]	Δ	one-shot	WEI	UNS	S-AAG	DYN	mask-unaware
	FedPM [Isik *et al.*, 2023]	w, g	dynamic	LRN	UNS	C-DLT	DYN	mask-aware
	PriPrune [Chu *et al.*, 2025]	w, Δ	one-shot	GRAD	UNS	C-ALT	PERS	mask-unaware
	QuanCryptFL [Mia and Amini, 2025]	w	one-shot	WEI	UNS	C-ALT	PERS	mask-unaware
	MemDefense [Shen *et al.*, 2025]	w	one-shot	GRAD	UNS	C-ALT	PERS	mask-unaware
Unlearning	Wang et al. [Wang *et al.*, 2022]	w	one-shot	ACT	STR	S-AAG	FIX	mask-unaware
	FedOrtho [Gong *et al.*, 2025]	w	one-shot	ACT	STR	S-AAG	FIX	mask-unaware
	FedUP [Romandini *et al.*, 2025]	w	one-shot	WEI	UNS	S-AAG	PERS	mask-unaware
	FUSED [Zhong *et al.*, 2025]	w	static	RND	UNS	S-INIT, C-DLT	FIX	mask-unaware
Personalization	FedMask [Li *et al.*, 2021]	w	dynamic	WEI, LRN	STR	C-DLT, C-ALT	DYN	mask-aware
	DisPFL [Dai *et al.*, 2022]	w	dynamic	WEI, GRAD	UNS	C-BLT, C-DLT, C-ALT	PERS	mask-aware
	FedRolex [Alam *et al.*, 2022]	w	dynamic	RND	STR	S-INIT, C-DLT, S-AAG	DYN	mask-aware
	ASPFL [Jiang *et al.*, 2023b]	w	dynamic	WEI, GRAD	UNS	S-INIT, C-DLT	PERS	mask-aware
	pFedGate [Chen *et al.*, 2023]	w, g	dynamic	LRN	STR	C-DLT	PERS	mask-unaware
	SpaFL [Kim *et al.*, 2024]	w	dynamic	LRN	UNS	C-DLT	PERS	mask-unaware
	FLASC [Kuo *et al.*, 2024]	Δ	dynamic	WEI	UNS	C-ALT, S-AAG	PERS	mask-unaware

Table 1: Overview of sparse FL methods organized according to the proposed taxonomy (Section 2.3). **Object**: model weights (w), model updates (Δ), activations (a), learned gates (g). **Regime**: one-shot, static, dynamic. **Criterion**: weight-based (WEI), gradient-based (GRAD), activation-based (ACT), learned (LRN), random (RND). **Pattern**: unstructured (UNS), structured (STR). **Stage**: server init. (S-INIT), client before training (C-BLT), client during training (C-DLT), client after training (C-ALT), server after aggregation (S-AAG). **Coordination**: fixed (FIX), dynamic (DYN), personalized (PERS). **Aggregation**: mask-unaware, mask-aware, sparsity restoration.

2025] reduces the overhead of fully homomorphic encryption by coupling it with low-bit quantization and unstructured pruning. Sparsity can also actively limit adversarial impact: SparseFed [Panda *et al.*, 2022] exploits the tension between large gradients needed to survive top-K selection and norm clipping, yielding certified robustness guarantees.

Privacy-preserving aggregation of sparse updates. Privacy-focused aggregation differs fundamentally from efficiency-driven sparsification. Rather than preserving sparsity across rounds, these methods tolerate sparsity erosion if it strengthens protection. Cryptographic approaches aggregate updates entirely in encrypted or secret-shared domains [Beguier *et al.*, 2021; Mia and Amini, 2025], ensuring the server never observes individual updates, even if the aggregated model becomes dense. Probabilistic methods such as FedPM [Isik *et al.*, 2023] aggregate distributions instead of realizations, while PriPrune [Chu *et al.*, 2025] operates transparently over standard aggregation rules.

3.3 Effective Unlearning

Federated unlearning (FU) aims to remove the contribution of specific training data from a federated model, such as the data of an entire client, a subset of samples, or particular classes [Romandini *et al.*, 2024]. Unlike standard FL, FU must selectively alter a trained model to eliminate the influence of targeted data while preserving overall utility.

Inspired by centralized machine unlearning (e.g., [Foster *et al.*, 2024]), FU methods leverage sparsity to identify and modify parameters that encode information about the forget data [Wang *et al.*, 2022; Zhong *et al.*, 2025; Romandini *et al.*, 2025; Gong *et al.*, 2025]. In contrast to efficiency-oriented sparsity, which prunes low-importance weights to reduce communication or computation, unlearning-oriented sparsity targets high-impact components associated with the data to be removed.

Existing sparsity-based FU methods differ primarily in *where* the forgettable information is assumed to reside and thus *at what granularity* sparsity is applied. Accordingly, current approaches can be grouped into three categories: (i) channel-level methods that remove discriminative feature channels, (ii) parameter-level methods that suppress individual high-impact weights, and (iii) module-level methods that confine deletions to specific architectural adapters.

Channel-level forgetting. Wang et al. [Wang *et al.*, 2022] perform class-level unlearning by pruning convolutional channels with high term–frequency discrimination scores computed from client-side activations, followed by light fine-tuning. Gong et al. [Gong *et al.*, 2025] introduce FedOrtho, which enforces kernel orthogonality to decouple

feature spaces and then soft-prunes forget-dominant channels based on aggregated activation discrepancies, enabling single-step class forgetting.

Parameter-level forgetting. FedUP [Romandini *et al.*, 2025] introduces a server-side pruning approach for adversarial settings where malicious clients do not cooperate. FedUP identifies high-impact parameters by measuring discrepancies between benign and malicious client updates, and selectively zeros them at the server without requiring historical updates, auxiliary data, or additional client-side computation.

Module-level forgetting. Zhong et al. [Zhong *et al.*, 2025] propose FUSED, which applies sparsity to trainable adapters rather than modifying base model parameters directly. Sensitive layers are identified via offline calibration, after which sparse adapters are trained on frozen backbones using only retained clients. This design reduces communication and limits parameter modifications, but relies on calibration heuristics and manually tuned sparsity ratios.

3.4 Improved Personalization

Personalized federated learning (PFL) addresses statistical heterogeneity by allowing clients to deviate from a single global model. Sparsity naturally supports personalization by activating different parameter subsets across clients while maintaining a shared parameter space. Existing methods can be broadly grouped into *mask-based personalization*, operating at client or finer granularity, and *model-heterogeneous* approaches that accommodate diverse client capacities.

Mask-based personalization. Most approaches assign each client a personalized sparse mask over shared weights. FedMask [Li *et al.*, 2021] learns heterogeneous structured masks over frozen parameters and aggregates only overlapping mask entries, preserving client-specific sparse structures. DisPFL [Dai *et al.*, 2022] extends this idea to decentralized settings via sparse-to-sparse gossip averaging over mask intersections, with masks refined through pruning and gradient-based regrowth. ASPFL [Jiang *et al.*, 2023b] introduces adaptive layer-wise sparsity with drop–grow dynamics, leading to implicit clustering as clients with similar data converge to similar sparse topologies. Finer-grained personalization further conditions sparsity below the client level: pFedGate [Chen *et al.*, 2023] predicts block-wise masks from input batches using private gating layers, while SpaFL [Kim *et al.*, 2024] communicates only global per-neuron thresholds, keeping weights local and yielding emergent personalized sparsity with minimal communication.

Model-heterogeneous personalization. Sparsity also enables participation under heterogeneous computational budgets. FedRolex [Alam *et al.*, 2022] trains rolling submodels aligned with client capacity, ensuring that all server parameters are updated over time. For large language models, FLASC [Kuo *et al.*, 2024] sparsifies LoRA adapters only during communication, with asymmetric upload and download sparsity to handle bandwidth constraints, while FedSpaLLM [Bai *et al.*, 2025] employs ℓ_0-norm aggregation to prevent highly sparse clients from diluting important parameters.

Aggregation under heterogeneous sparsity. Personalized sparse FL fundamentally alters aggregation, shifting it from enforcing consensus to selectively transferring knowledge across partially overlapping supports. Standard FedAvg leads to sparsity erosion when masks diverge, motivating alternatives that aggregate only overlapping parameters [Dai *et al.*, 2022; Li *et al.*, 2021], weight updates by client participation [Jiang *et al.*, 2023b], or avoid weight aggregation entirely [Kim *et al.*, 2024]. Nonetheless, a core tension remains: increasing sparsity heterogeneity reduces aggregation benefits, while enforcing alignment constrains personalization.

4 Current Trends and Open Issues

To inform practitioners and highlight open challenges, we analyze experimental practices across the 29 sparse FL methods surveyed, revealing strong standardization but limited realism in evaluations. We also discuss gaps in theoretical foundations that currently limit principled design of sparse FL.

Hardware evaluation gap. A critical limitation is the lack of hardware realism. Only 17.2% of surveyed papers (5/29) report wall-clock or on-device measurements [Li *et al.*, 2021; Jiang *et al.*, 2023a; Chen *et al.*, 2023; Mia and Amini, 2025; Bibikar *et al.*, 2022]; the vast majority rely on theoretical FLOPs as efficiency proxies. Moreover, just 13.8% (4/29) include actual edge device timing, typically on Raspberry Pi or NVIDIA Jetson. As shown in Table 1, 79.3% (23/29) of methods employ *unstructured* sparsity, yet such patterns provide limited or inconsistent speedups on commodity hardware where *structured* patterns are required for real acceleration. This mismatch between evaluation methodology and deployment reality undermines efficiency claims.

Beyond low-resolution benchmarks and small architectures. Sparse FL evaluation is heavily concentrated on low-resolution vision benchmarks and small convolutional models. CIFAR-10 alone appears in 86.2% of surveyed works (25/29), the CIFAR family in 93.1% (27/29), and MNIST variants in 51.7% (15/29), reflecting continued reliance on early FL testbeds [McMahan *et al.*, 2017]. Consistently, ResNet-18 and generic CNNs account for 75.9% of evaluated architectures, while transformers and Vision Transformers [Dosovitskiy *et al.*, 2021] appear in only 17.2% (5/29). This co-limitation of data and model scale raises questions about whether observed sparsity gains extend to higher-dimensional inputs, richer distributions, and architectures with attention and large feed-forward blocks. Beyond vision, task coverage remains sparse: language modeling appears in 10.3% of works [Li *et al.*, 2021; Panda *et al.*, 2022; Kuo *et al.*, 2024], only one study targets LLM-scale models [Bai *et al.*, 2025], and medical imaging is addressed once [Mia and Amini, 2025].

Oversimplified data and client heterogeneity. Non-IID data is predominantly simulated via label skew using Dirichlet partitioning with concentration parameters $\alpha \in [0.1, 1.0]$. While this convention aids reproducibility, it may oversimplify real-world heterogeneity. Client configurations also cluster narrowly: 69.0% (20/29) of studies use 100 clients with 10% participation per round, while only two works scale

Challenges	Regime	Stage	Pattern	Coordination	Aggregation
Communication	One-shot	C-ALT / S-AAG	UNS	Fixed / None	Mask-unaware or Restoration
Computation	Static / Dynamic	C-BLT / C-DLT	STR	Fixed / Dynamic	Mask-aware or Restoration
Privacy	One-shot	C-ALT	UNS	Personalized	Mask-unaware
Unlearning	One-shot	S-AAG	STR or UNS	Personalized	Mask-unaware
Personalization	Dynamic	C-DLT	STR or UNS	Personalized	Mask-aware

Table 2: Practitioner mapping from challenges to recommended sparsity configurations along key taxonomy dimensions.

beyond 1,000 clients. As analyzed in Section 3, data heterogeneity directly influences local mask evolution, and systematic studies on how sparsity patterns interact with realistic client scales and participation rates remain limited.

Limited theoretical understanding. Despite early progress, theoretical foundations for sparse FL remain incomplete. Existing analyses only partially capture how sparse masks interact with data heterogeneity, client participation, and server aggregation. While recent efforts provide convergence guarantees under dynamic client-specific sparsity [Zhou *et al.*, 2023] or adaptive pruning [Yu *et al.*, 2025], these results address isolated settings. Beyond convergence, there is no general theory describing sparsity–utility trade-offs, aggregation bias, or coverage requirements across heterogeneous deployments. Similar gaps exist for privacy and unlearning: the extent to which sparsity amplifies, preserves, or weakens guarantees remains poorly understood.

5 Practitioner Guidance

The characterization of sparsity varies depending on the challenge it targets. As practical guidance, we extract *design principles* that relate the main dimensions of our taxonomy to relevant deployment scenarios. Table 2 summarizes common configurations observed in the surveyed literature.

Identify the primary motivation. Sparse FL can be driven by four main motivations: efficiency, privacy enhancement, unlearning, or personalization. Identifying the dominant goal narrows the viable design space for regime, application stage, pattern, and coordination.

Select an appropriate sparsity regime. The *regime* should reflect the deployment goal: *one-shot* sparsity suits pure communication efficiency; *static sparse training* reduces computation and memory throughout training; *dynamic sparse training* provides finer adaptation under non-identical data or evolving client populations. Privacy- and unlearning-oriented approaches typically train dense models and apply one-shot pruning after training, since modifying the optimization dynamics can be unnecessary.

Place sparsity at the correct stage of the FL workflow. The application stage determines which resources benefit. C-BLT/C-DLT reduce client-side computation and memory; C-ALT primarily reduces uplink communication; S-AAG can reduce downlink communication by restoring sparsity after densification. For privacy, C-ALT enables asymmetric disclosure, while dense training preserves gradients locally. For unlearning, S-AAG is natural for class- or client-level removal without re-running local training.

Match sparsity pattern to the bottleneck. *Unstructured* sparsity maximizes compression for communication efficiency and is common in privacy settings due to its less informative structure. *Structured* sparsity (e.g., channels, blocks) aligns with hardware execution units and suits computation- or memory-constrained edge devices. Unlearning can leverage *structured* sparsity when the influence of specific forget data emerges from clustered model parameters, but can also exploit *unstructured* sparsity for selective removal of fine-grained contributions. Personalization may adopt either pattern depending on whether the goal is communication reduction (unstructured) or on-device efficiency (structured).

Choose mask coordination based on data heterogeneity. Under homogeneous data, fixed global masks minimize coordination overhead. In heterogeneous client or data settings, dynamic coordination prevents model collapse, while personalized masks decouple model structures at the cost of reduced parameter overlap. In privacy-oriented and unlearning-oriented settings, mask divergence can be desirable to reduce shared exposure or to localize forgettable information. Personalization naturally benefits from mask divergence for device-adaptive model capacity.

Couple sparsity with aggregation logic. Standard FedAvg implicitly densifies sparse client updates, preventing sparsity from persisting across rounds. As a result, efficiency-oriented deployments typically require mask-aware aggregation or server-side sparsity restoration to preserve communication and computation benefits over time. Similar issues arise under data and model heterogeneity, including personalized FL settings, where naive aggregation likewise disrupts sparsity patterns. By contrast, privacy– and unlearning–oriented deployments often tolerate or even benefit from densification. In privacy-focused settings, server-side reconstruction noise or dense aggregation may strengthen secrecy or DP guarantees. In unlearning-oriented pipelines, pruning is commonly applied as a one-shot post-aggregation operation.

6 Conclusion

Sparsity supports multiple roles in FL, from communication and computation efficiency to privacy, unlearning, and personalization. We introduced a federated-specific taxonomy to organize this space and compare existing methods. Using this framework, we reviewed the literature along its key design dimensions, identified common patterns and methodological gaps, and analyzed experimental practices and deployment assumptions. We further distilled practitioner-oriented design principles and highlighted open challenges to guide future research and real-world adoption.

References

[Alam *et al.*, 2022] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in neural information processing systems*, 35:29677–29690, 2022.

[Babakniya *et al.*, 2023] Sara Babakniya, Souvik Kundu, Saurav Prakash, Yue Niu, and Salman Avestimehr. Revisiting sparsity hunting in federated learning: Why does sparsity consensus matter? *Trans. Mach. Learn. Res.*, 2023, 2023.

[Bai *et al.*, 2025] Guangji Bai, Yijiang Li, Zilinghan Li, Liang Zhao, and Kibaek Kim. Fedspallm: Federated pruning of large language models. In *Proceedings of NAACL-HLT 2025*, pages 8361–8373. ACL, 2025.

[Beguier *et al.*, 2021] Constance Beguier, Mathieu Andreux, and Eric W. Tramel. Efficient sparse secure aggregation for federated learning, 2021.

[Bibikar *et al.*, 2022] Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 6080–6088, 2022.

[Bustincio *et al.*, 2025] Rómulo Bustincio, Allan M. de Souza, Joahannes B. D. da Costa, Luis F. G. Gonzalez, and L. F. Bittencourt. Reducing communication overhead through one-shot model pruning in federated learning. *Annals of Telecommunications*, 80(9):901–913, 2025.

[Chen *et al.*, 2023] Daoyuan Chen, Liuyi Yao, Dawei Gao, Bolin Ding, and Yaliang Li. Efficient personalized federated learning via sparse model-adaptation. 2023.

[Cheng *et al.*, 2024] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):10558–10578, 2024.

[Chu *et al.*, 2025] Tianyue Chu, Mengwei Yang, Nikolaos Laoutaris, and Athina Markopoulou. Priprune: Quantifying and preserving privacy in pruned federated learning. *ACM Trans. Model. Perform. Evaluation Comput. Syst.*, 10(2):11:1–11:30, 2025.

[Dai *et al.*, 2022] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. In *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pages 4587–4604. PMLR, 2022.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[Foster *et al.*, 2024] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 12043–12051. AAAI Press, 2024.

[Frankle and Carbin, 2019] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[Gong *et al.*, 2025] Qinghui Gong, Xue Yang, and Xiaohu Tang. Orthogonal soft pruning for efficient class unlearning. *CoRR*, abs/2506.19891, 2025.

[Guastella *et al.*, 2025] Adriano Guastella, Lorenzo Sani, Alex Iacob, Alessio Mora, Paolo Bellavista, and Nicholas Donald Lane. Sparsyfed: Sparse adaptive federated learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*, 2025.

[Hanser *et al.*, 2025] Thierry Hanser, Ernst Ahlberg, Alexander Amberg, Lennart T Anger, Chris Barber, Richard J Brennan, Alessandro Brigo, Annie Delaunois, Susanne Glowienke, et al. Data-driven federated learning in drug discovery with knowledge distillation. *Nature Machine Intelligence*, pages 1–14, 2025.

[Hoefler *et al.*, 2021] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22:241:1–241:124, 2021.

[Huang *et al.*, 2022] Tiansheng Huang, Shiwei Liu, Li Shen, Fengxiang He, Weiwei Lin, and Dacheng Tao. Achieving personalized federated learning with sparse local models. *CoRR*, abs/2201.11380, 2022.

[Isik *et al.*, 2023] Berivan Isik, Francesco Pase, Deniz Gündüz, Tsachy Weissman, and Michele Zorzi. Sparse random networks for communication-efficient federated learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023.

[Jiang *et al.*, 2023a] Yuang Jiang, Shiqiang Wang, Víctor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Trans. Neural Networks Learn. Syst.*, 34(12):10374–10386, 2023.

[Jiang *et al.*, 2023b] Yuhui Jiang, Xingjian Lu, Haikun Zheng, and Wei Mao. ASPFL: efficient personalized federated learning for edge based on adaptive sparse training. In *IEEE International Conference on Web Services, ICWS 2023*, pages 269–277. IEEE, 2023.

[Kairouz *et al.*, 2021] 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.

[Kim *et al.*, 2024] Minsu Kim, Walid Saad, Merouane Debbah, and Choong S Hong. Spafl: Communication-efficient

federated learning with sparse models and low computational overhead. *Advances in Neural Information Processing Systems*, 37:86500–86527, 2024.

[Kuo *et al.*, 2024] Kevin Kuo, Arian Raje, Kousik Rajesh, and Virginia Smith. Federated lora with sparse communication. *arXiv preprint arXiv:2406.05233*, 2024.

[Lee and Jang, 2025] Sangmin Lee and Hyeryung Jang. Bipruneﬂ: Computation and communication efficient federated learning with binary quantization and pruning. *IEEE Access*, 13, 2025.

[Li *et al.*, 2021] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, page 42–55, 2021.

[McMahan *et al.*, 2017] 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, 2017.

[Mia and Amini, 2025] Md Jueal Mia and M Hadi Amini. Quancrypt-fl: Quantized homomorphic encryption with pruning for secure federated learning. *IEEE Transactions on Artificial Intelligence*, pages 1–16, 2025.

[Mironov, 2017] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, page 263–275, 2017.

[Panda *et al.*, 2022] Ashwinee Panda, Saeed Mahloujifar, Arjun Nitin Bhagoji, Supriyo Chakraborty, and Prateek Mittal. Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification. In *International Conference on Artificial Intelligence and Statistics*, pages 7587–7624. PMLR, 2022.

[Qiu *et al.*, 2022] Xinchi Qiu, Javier Fernández-Marqués, Pedro P. B. de Gusmao, Yan Gao, Titouan Parcollet, and Nicholas Donald Lane. Zerofl: Efficient on-device training for federated learning with local sparsity. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

[Romandini *et al.*, 2024] Nicolò Romandini, Alessio Mora, Carlo Mazzocca, Rebecca Montanari, and Paolo Bellavista. Federated unlearning: A survey on methods, design guidelines, and evaluation metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[Romandini *et al.*, 2025] Nicolò Romandini, Cristian Borcea, Rebecca Montanari, and Luca Foschini. Fedup: Efficient pruning-based federated unlearning for model poisoning attacks. *arXiv preprint arXiv:2508.13853*, 2025.

[Sattler *et al.*, 2019] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019.

[Shah and Lau, 2021] Suhail Mohmad Shah and Vincent KN Lau. Model compression for communication efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):5937–5951, 2021.

[Shen *et al.*, 2025] Meng Shen, Jin Meng, Ke Xu, Shui Yu, and Liehuang Zhu. Memdefense: Defending against membership inference attacks in iot-based federated learning via pruning perturbations. *IEEE Transactions on Big Data*, 11(5):2135–2147, 2025.

[Stripelis *et al.*, 2022] Dimitris Stripelis, Umang Gupta, Greg Ver Steeg, and Jose Luis Ambite. Federated progressive sparsification (purge, merge, tune)+. *arXiv preprint arXiv:2204.12430*, 2022.

[Wang *et al.*, 2022] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM web conference 2022*, pages 622–632, 2022.

[Woisetschläger *et al.*, 2024] Herbert Woisetschläger, Alexander Erben, Shiqiang Wang, Ruben Mayer, and Hans-Arno Jacobsen. A survey on efficient federated learning methods for foundation model training. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8317–8325, 2024.

[Wu *et al.*, 2020] Xueyu Wu, Xin Yao, and Cho-Li Wang. Fedscr: Structure-based communication reduction for federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1565–1577, 2020.

[You *et al.*, 2025] Linlin You, Zihan Guo, Chau Yuen, Calvin Yu-Chian Chen, Yan Zhang, and H Vincent Poor. A framework reforming personalized internet of things by federated meta-learning. *Nature communications*, 16(1):3739, 2025.

[Yu *et al.*, 2025] Dongxiao Yu, Yuan Yuan, Yifei Zou, Xiao Zhang, Yu Liu, Lizhen Cui, and Xiuzhen Cheng. Pruning-based adaptive federated learning at the edge. *IEEE Transactions on Computers*, 74(5):1538–1548, 2025.

[Zhang *et al.*, 2025] Pengyu Zhang, Yingjie Liu, Yingbo Zhou, Xian Wei, and Mingsong Chen. When foresight pruning meets zeroth-order optimization: Efficient federated learning for low-memory devices. *Journal of Systems Architecture*, 168:103526, 2025.

[Zhong *et al.*, 2025] Zhengyi Zhong, Weidong Bao, Ji Wang, Shuai Zhang, Jingxuan Zhou, Lingjuan Lyu, and Wei Yang Bryan Lim. Unlearning through knowledge overwriting: Reversible federated unlearning via selective sparse adapter. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 30661–30670, 2025.

[Zhou *et al.*, 2023] Hanhan Zhou, Tian Lan, Guru Venkataramani, and Wenbo Ding. Every parameter matters: ensuring the convergence of federated learning with dynamic heterogeneous models reduction. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.