pFLAIR: PERSONALIZED FEDERATED LEARNING USING ATTENTION-INFORMED MODEL RECOMBINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized federated learning (pFL) has emerged as a key approach for addressing statistical heterogeneity, while retaining the privacy and collaboration benefits of federated learning. However, most existing pFL methods place the bulk of personalization on the client side, often adding extra local modules, inner-loop optimization, or per-client adaptation steps, which can be computationally and memory intensive, and contrary to a core practical motivation of FL: keeping client workloads and on-device model footprints small. This is especially limiting under label skew, the most realistic heterogeneity regime, where clients observe different subsets or frequencies of classes reflecting natural specialization in real-world deployments. We propose pFLAIR , a pFL framework that shifts a substantial part of personalization to the server by learning an explicit, trainable mixing mechanism that determines how client knowledge should be combined. Crucially, unlike several competitive pFL methods that rely on communicating explicit label frequencies or class-conditioned statistics to achieve their best performance, pFLAIR operates without requiring explicit label-distribution statistics. This avoids exposing such statistics to the server, which is the very premise of federated learning. Across heterogeneous settings, pFLAIR consistently outperforms strong pFL baselines, with particularly large gains under challenging label skew, and degrades far more gracefully than competing methods as the number of clients grows, demonstrating a better scalability for federated deployments.

1 INTRODUCTION

Federated Learning (FL) enables a population of clients to train models collaboratively while keeping raw data local, typically by alternating local optimization with server-side aggregation (McMahan et al., 2017; Li et al., 2020). In real deployments, however, the "single global model" objective is often misaligned with practice: clients vary in data distributions, compute, and connectivity, leading to statistical and systems heterogeneity that can destabilize optimization and harm local performance (Li et al., 2020). A particularly prevalent and challenging regime is label skew, where each client observes only a subset of classes or highly imbalanced label frequencies. Under such skew, local updates can become conflicting, and naive global averaging may yield a model that is simultaneously suboptimal for many clients.

Personalized Federated Learning (pFL) addresses this mismatch by aiming to learn a collection of client-specific models rather than a single global solution (Tan et al., 2022a). Different methods can be used in order to fulfill this requirement. Prior work achieves personalization through meta-learning or adaptation objectives (Dinh et al., 2020), by decoupling shared representations from lightweight personalized components (Oh et al., 2022; Chen & Chao, 2022), or by learning client-dependent mixing weights to enable selective collaboration (Zhang et al., 2021; Huang et al., 2021b). While effective, many of these approaches place algorithmic burden on the client adaptation steps, additional personalized states, or repeated fine-tuning, which can be costly for resource-constrained devices and runs counter to a practical motivation of FL: keeping client participation lightweight. This asymmetry is fundamental: clients are typically resource-constrained edge devices (phones, IoT sensors, embedded nodes) while the coordinating server is well-provisioned. Shifting this burden to the server is therefore a natural design goal.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This paper pursues exactly this: we propose PFLAIR , a server-oriented pFL framework tailored to realistic label-skew regimes. Clients communicate compact, task-relevant summaries and the server learns how to recombine client models via an attention-guided mechanism. At a high level, the server builds client descriptors from head parameters, injects class semantics through lightweight conditioning, and performs class-wise multi-head attention across clients, which is aggregated into a mixing matrix that synthesizes a personalized model for each client in parameter space. Because the heavy lifting (descriptor learning, conditioning, attention, and recombination) happens centrally, PFLAIR keeps clients on a standard supervised loop with minimal on-device personalization steps, relocating the personalization machinery to the server where compute is abundant.

The server-side viewpoint is especially appealing under label skew, where the usefulness of transfer is inherently class dependent: clients may be compatible for some classes but not others. PFLAIR operationalizes this intuition through class-indexed attention heads that can share selectively when overlap is beneficial and suppress transfer when it is harmful, while remaining deliberately generic: no assumptions are made about which classes should overlap, and the learned attention adapts case-by-case. Crucially, PFLAIR is designed to remain effective as the federation grows: clients run essentially the same local training loop regardless of the number of participants, and only upload compact summaries, while the server performs the expensive coordination once per round. As the client population grows (e.g., from tens to hundreds of clients), PFLAIR maintains strong accuracy and degrades much more gracefully than prior pFL methods that rely on costly client-side adaptation or repeated fine-tuning. Empirically, this design yields consistent improvements over strong pFL baselines in realistic skewed settings (by several percentage points in accuracy), while simultaneously lowering client compute relative to methods of comparable accuracy and without sending potentially sensitive information.

2 Related works

Client-side personalization. Most pFL methods place personalization on the client. Meta-learning approaches such as Per-FedAvg (Fallah et al., 2020) learn initializations for fast local adaptation, at the cost of repeated inner-loop updates. Model-decoupling methods decompose the network into shared and personalized components: FedBABU (Oh et al., 2022) freezes the head during training, FedRoD (Chen & Chao, 2022) maintains dual heads, PerFedMask (Setayesh et al., 2023) learns per-client masking vectors, and pFedClub (Wang et al., 2024) selectively swaps layers. FedMR (Hu et al., 2024) recombines layers across clients but targets global generalization rather than personalization. pFedFDA (McLaughlin & Su, 2024) adapts a generative classifier to each client's feature distribution. Regularization-based methods such as pFedMe (Dinh et al., 2020), FedProx (Li et al., 2020), and SCAFFOLD (Karimireddy et al., 2019) couple local and global objectives via proximal penalties. While effective, all these approaches add computation, memory, or adaptation steps on device.

Prototype and representation-guided methods. A second family communicates representation-space summaries to guide local learning. FedProto (Tan et al., 2022b) shares class prototypes, GPFL (Zhang et al., 2023b) adds global and personalized feature-level guidance, FedDBE (Zhang et al., 2023a) eliminates domain bias, FedPAC (Xu et al., 2023) combines feature alignment with classifier collaboration, and CCVR (Luo et al., 2021) calibrates the classifier with virtual representations. These methods rely on communicated statistics that can be weakly aligned across clients under strong label skew and may leak distributional information.

Adaptive aggregation, graphs, and hypernetworks. Instead of uniform averaging, some methods adapt aggregation weights. FedFOMO (Zhang et al., 2021) learns per-client mixture weights, FedAMP (Huang et al., 2021b) uses attentive message passing, FedALA (Zhang et al., 2023c) adapts local aggregation, pFedGAT (Zhou et al., 2025) and FedAGHN (Song et al., 2025) learn graph-based collaboration structures, and FedMosaic (Abourayya et al., 2025) enables per-example trust-aware collaboration through co-training. A related line uses hypernetworks to generate personalized weights from client embeddings (Zhang et al., 2025; Li et al., 2025), sharing with PFLAIR the philosophy of compressing client state into compact descriptors but differing in that they generate models from scratch rather than recombining existing ones. Privacy-aware approaches such as FedDPA (Yang et al., 2023) further highlight the tension between richer communicated signals and the attack surface exposed to the server. Critically, existing client-level attention and similarity-based aggregation methods typically represent the compatibility between a pair of clients with a single scalar weight, a coarse signal when clients are compatible on some classes but not others.

Contribution of pFLAIR . The novelty of pFLAIR compared to prior works in those three axes is as follows: First, personalization is learned entirely on the server, relieving clients of on-device personalization steps: the aggregation rule is a trainable function of compact client summaries, optimized end-to-end from local loss feedback, keeping clients on a standard supervised loop. Second, pFLAIR computes class-wise attention with learned projections on class-conditioned tokens, producing a mixing matrix informed by fine-grained per-class compatibility rather than a single global similarity score allowing better knowledge sharing across clients. Third, FiLM-based fusion of projected client descriptors with class embeddings addresses the cross-client alignment problem that raw prototypes cannot address under label skew. This design shifts computation off-device and enables selective transfer without requiring label-distribution statistics to be shared. A direct consequence of the server-centric architecture combined with class-wise attention is scalability: the client-side computation does not grow with the number of participating clients, and the attention mechanism learns instance-specific mixing rather than defaulting to population-wide averaging, pFLAIR maintains strong accuracy as the number of clients grows, a setup where most pFL methods deteriorate rapidly due to increasing fragmentation and conflicting updates (Section 4.4).

3 pFLAIR

3.1 Problem Statement

We consider a federated learning system with N clients. Client $i \in [N]$ owns a private labeled dataset $\mathcal{D}_i$ drawn from a client-specific distribution (typically non-IID). Our goal is personalized federated learning: learn client-specific models that perform well on each client's distribution while enabling beneficial cross-client transfer. A notation table is provided in Appendix A.

Client i's model is split into a feature extractor (base) and a classifier head (head). In practice the head corresponds to the final linear (classification) layer of the network (Oh et al., 2022; Chen & Chao, 2022). The feature extractor maps an input x to a representation $f_i = \phi(x; \theta_i^{base}) \in \mathbb{R}^d$, and the head maps this representation to logits $\hat{y}_i = \psi(f_i; \theta_i^{head}) \in \mathbb{R}^C$, C being the number of classes. In addition, each client maintains a local class embedding table $E_i^{pers} \in \mathbb{R}^{C \times d}$, where row $e_{i,c}^{pers} \in \mathbb{R}^d$ is a trainable embedding for class c on client i.

In pFLAIR , the server also produces an attention-informed class embedding table for each client, $E_i^{att} \in \mathbb{R}^{C \times d}$, obtained by recombining other clients' embeddings using the same learned mixing weights used to recombine model parameters. This table is communicated back to client i and used as an auxiliary target to guide representation learning. The personalized objective on client i is the expected local loss,

$$F_i = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\left[L_i\big(x, y; \theta_i^{base}, \theta_i^{head}, E_i^{pers}, E_i^{att}\big)\right],$$

and the overall learning goal can be written as minimizing an aggregate objective across clients, e.g., $\sum_{i=1}^{N} n_i F_i$ with $n_i \propto |\mathcal{D}_i|$. Unlike standard FL, pFLAIR additionally learns server-side aggregation parameters Θ_S that produce an attention matrix; these parameters are optimized using gradient feedback from clients, enabling end-to-end learning of the aggregation rule. This attention is an asymmetric mechanism, giving it the ability to let a client attend to others without forcing reciprocity.

3.2 Methodology

Overview. pFLAIR alternates between local client optimization and server-side construction of personalized mixed models (Figure 1). Locally, each client learns a standard classifier and a set of local class embeddings that structure its representation space. On the server, we compress each client's head parameters into a descriptor, fuse this descriptor with the client's class embeddings using FiLM (Perez et al., 2018) to build class-conditioned tokens, and run class-wise attention across clients to produce a mixing matrix R. The server then mixes full client state dictionaries according to R, sends the mixed models back for evaluation, and updates the server aggregation modules end-to-end from the local-loss gradients returned by clients. The server optimizes only its own parameters (projector, FiLM layer, and attention) and clients continue to train with standard local SGD, augmented with the two lightweight embedding losses (see Appendix B for algorithm).

Client training. On client i, the base encoder produces features $f_i = \phi(x; \theta_i^{base})$ and the head outputs logits $\hat{y} = \psi(f_i; \theta_i^{head})$. The task loss is standard cross-entropy $L_i^{task} = \ell(\hat{y}, y)$, which updates both θ_i^{base} and θ_i^{head}.

Client: Local class embedding learning. In parallel, client i trains a personalized class embedding table $E_i^{pers} \in \mathbb{R}^{C \times d}$ as of a set of learnable class prototypes in the same space as the feature representations. With cosine similarity $\text{cosim}(f_i, e) = \frac{f_i^\top e}{\|f_i\|_2 \|e\|_2}$, we define the embedding-guidance loss as a cross-entropy over cosine logits:

$$L_i^{emb} = -\log \frac{\exp\big(\text{cosim}(f_i, e_{i,y}^{pers})\big)}{\sum_{c=1}^{C} \exp\big(\text{cosim}(f_i, e_{i,c}^{pers})\big)}. \tag{1}$$

This loss pulls f_i toward $e_{i,y}^{pers}$ and pushes it away from other class embeddings, making E_i^{pers} discriminative and shaping the geometry of the local feature space, without containing any raw data information. This is illustrated in Appendix F. Both E_i^{pers} and the backbone parameters θ_i^{base} are updated jointly through this loss, so the embeddings and representations co-adapt during training.

Client: Attention-informed embedding guidance. At the end of each round, the server returns an attention-informed class embedding table $E_i^{att} \in \mathbb{R}^{C \times d}$ constructed by recombining other clients' embeddings using the same mixing weights used for model recombination. While E_i^{pers} describes only a client's local class geometry, E_i^{att} is composed of cross-client knowledge about each class, weighted by the learned compatibility between clients. Client i uses E_i^{att} as an auxiliary target to guide representation learning via $L_i^{mse} = \big\|f_i - e_{i,y}^{att}\big\|_2^2$, encouraging the representation of class y to align with the cross-client, attention-weighted embedding. We use MSE rather than cosine distance here because the goal is full alignment in representation space rather than angular similarity alone. This loss acts as a soft regularizer that steers the local feature extractor toward representations compatible with related clients, without requiring raw data exchange.

Client: Local objective and communicated summaries. Finally, on each client i, we optimize

$$L_i = L_i^{task} + L_i^{emb} + L_i^{mse},$$

over variables $(\theta_i^{base}, \theta_i^{head}, E_i^{pers})$. After each round of local training, client i uploads: the full backbone state θ_i^{base}, a flattened vector of head parameters $p_i = \text{vec}(\theta_i^{head})$, and the personalized class embedding table E_i^{pers} (depicted by ① in Figure 1). The server then uses $\{E_i^{pers}\}_{i=1}^N$ to construct $\{E_i^{att}\}_{i=1}^N$ and returns them to clients for the next local round.

Server: client representation conditioning. The server builds a compact, comparable representation of each client from its classifier head. It maps the head signatures $\{p_i\}_{i=1}^N$ into a shared descriptor space $Z = \{Z_i\}_{i=1}^N \subset \mathbb{R}^d$ using a learnable autoencoder, whose decoder reconstructs the input and is supervised by a reconstruction loss $L_{recon} = \|P - \hat{P}\|_2^2$, with $P = [p_1; \ldots; p_N]$. This serves two purposes: compressing high-dimensional head parameters into a stable low-dimensional summary of each client's decision geometry, and encouraging a compact representation in which client descriptors can be compared.

These descriptors are then fused with local class information to produce class-aware tokens. For each client i and class c, we generate FiLM parameters $(\gamma_{i,c}, \beta_{i,c})$ (Perez et al., 2018) from the local class embedding $e_{i,c}$, and apply an affine modulation to the descriptor:

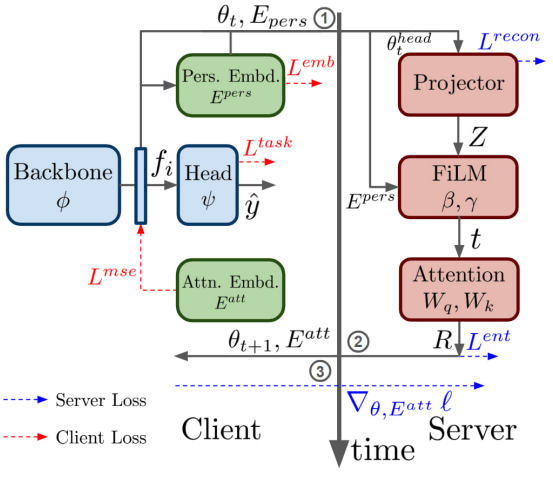


Figure 1: Overview of pFLAIR workflow: client training and server-side attention-guided recombination

$$t_{i,c} = \sigma((1 + \gamma_{i,c}) \odot Z_i + \beta_{i,c}), \tag{2}$$

where σ is the ReLU activation function (Agarap, 2018), and $\odot$ the Hadamard product. This fusion is key: class embeddings alone encode intra-client geometry and can be weakly aligned across clients under label skew, while descriptors Z_i contains model-level information about the client's current classifier. FiLM modulation yields tokens that are simultaneously comparable across clients (through the shared descriptor space) and class-conditional, substantially improving cross-client relationship identifiability over either signal alone (Section 5).

Server: class-wise multi-head attention for model mixing. We apply multi-head attention (Vaswani et al., 2017) across clients on the class-aware tokens, with one head per class. This design reflects the intuition that cross-client transfer can be class-dependent: two clients may be compatible for some classes but not others. For each class c, we compute scaled dot-product attention over clients:

$$A_{ij}^{(c)} = \mathrm{softmax}_j\left(\frac{\langle q_{i,c}, k_{j,c}\rangle}{\sqrt{d}}\right), \quad q_{i,c} = W_Q^{(c)} t_{i,c},\ k_{i,c} = W_K^{(c)} t_{i,c},\ W_Q^{(c)}, W_K^{(c)} \in \mathbb{R}^{d\times d} \tag{3}$$

where $A_{ij}^{(c)}$ quantifies how much client i should attend to client j for class c. The per-class heads are then averaged into a single mixing matrix:

$$R \;=\; \frac{1}{C}\sum_{c=1}^{C} A^{(c)} \in \mathbb{R}^{N\times N}. \tag{4}$$

To control the concentration of the mixing weights and stabilize optimization, we add an entropy regularizer $L^{ent} = -\mathrm{mean}(R \odot \log(R))$. This term helps early training, where attention is roughly uniform, by accelerating convergence toward selective clustering before the class-conditioned tokens become fully informative. Its effect is illustrated in Appendix D.

Server: mixing in parameter space and embedding space. Given R, the server synthesizes a personalized model for each client by applying R directly in parameter space. The mixed parameters and the attention-informed embeddings are then sent back to clients (② in Figure 1).

$$\theta_{i,t+1} = \sum_{j=1}^{N} R_{ij}\,\theta_{j,t}, \qquad E_i^{att} = \sum_{j=1}^{N} R_{ij}\,E_j^{pers}. \tag{5}$$

Server training and end-to-end gradients. After receiving the mixed model, each client evaluates it on its local data, and the server learns its aggregation modules by minimizing a sample-weighted objective:

$$L^{serv} \;=\; \sum_{i=1}^{N} n_i\,\ell_i \;+\; L^{recon} \;+\; L^{ent}, \qquad n_i = \frac{|\mathcal{D}_i|}{\sum_{j=1}^{N}|\mathcal{D}_j|}. \tag{6}$$

The key point is that ℓ_i depends on the server parameters Θ_S (projector, FiLM, and attention) only through the mixed model θ_i and embeddings E_i^{att} that the server produced, so the gradient factorizes exactly at the client/server boundary:

$$\frac{\partial \ell_i}{\partial \Theta_S} = \frac{\partial \ell_i}{\partial \theta_i}\,\frac{\partial \theta_i}{\partial \Theta_S} + \frac{\partial \ell_i}{\partial E_i^{att}}\,\frac{\partial E_i^{att}}{\partial \Theta_S}. \tag{7}$$

Each client therefore computes only the local factors $\partial\ell_i/\partial\theta_i$ and $\partial\ell_i/\partial E_i^{att}$ with a single backward pass on its own data (no raw data leaves the device) and returns them (③ in Figure 1). The server, which built R, completes the chain rule into Θ_S. This keeps the client on a standard supervised loop while making the aggregation rule end-to-end learnable for downstream client performance, and it is exactly equivalent to applying the chain rule to the server-side computation graph connecting Θ_S to the received θ_i and E_i^{att}, without approximating the server-side derivatives.

Communication and data exchange. Beyond the standard per-round model exchange, pFLAIR adds the two embedding tables and the returned gradient, which can be piggybacked on the next round's upload and needs no extra communication round; a full method analysis is given in Appendix C.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate PFLAIR across vision, language, and sensor modalities. For vision, we use Fashion-MNIST, CIFAR-10, and CIFAR-100 (Xiao et al., 2017; Krizhevsky, 2009) with a standard 4-layer CNN model(McMahan et al., 2017; Luo et al., 2021). For language, we use AG News (Zhang et al., 2015) and Amazon Reviews (Feng et al., 2021) with a FastText encoder (Joulin et al., 2017) followed by an MLP classifier. For sensor data, we use the HAR (Human Activity Recognition) dataset (Anguita et al., 2013) with the HARCNN architecture (Zhao et al., 2018). This combination of modalities and architectures tests whether PFLAIR 's server-side aggregation mechanism generalizes. Unless otherwise stated, we report the average test accuracy across clients as mean $\pm$ standard deviation over 10 independent runs, following the evaluation protocol of prior pFL work.

Data heterogeneity. To evaluate PFLAIR under diverse forms of statistical heterogeneity in real federated deployments, we construct three types of heterogeneous settings across our datasets.
Label skew is the most common form of heterogeneity, where clients observe different subsets or frequencies of classes. We consider two variants of increasing difficulty. In the **practical setting**, we sample data through the Dirichlet distribution (Lin et al., 2020), denoted $\mathrm{Dir}(\alpha)$: for each class c, we draw a client proportion vector $q_c \sim \mathrm{Dir}(\alpha)$ and allocate $q_{c,i}$ proportion of class c samples to client i. We also study the **pathological setting** (Huang et al., 2021b) case, a common comparison point for pFL methods where each client only receives a subset of classes.
Feature shift captures the scenario where clients share the same label space but differ in input characteristics. Following prior work (McAuley et al., 2015), we test this setting on Amazon Reviews, where each client corresponds to a distinct product domain, inducing natural covariate shift while preserving the same sentiment categories.
Real-world heterogeneity. On HAR (Anguita et al., 2013), sensor (IoT) data is naturally collected and stored across 30 users performing six different activities. This setting combines organic label imbalance, feature variation across devices, and unequal dataset sizes, reflecting deployment conditions without any artificial partitioning.

Baselines. Our goal is not an exhaustive comparison against all FL methods, but a controlled evaluation against representative approaches from the major pFL families described in Section 2: global aggregation (FedAvg (McMahan et al., 2017)), proximal regularization (FedProx (Li et al., 2020)), model decoupling (FedBABU (Oh et al., 2022), FedROD (Chen & Chao, 2022)), prototype-based (FedProto (Tan et al., 2022b), FedDBE (Zhang et al., 2023a)), adaptive aggregation (FedFOMO (Zhang et al., 2021), FedAMP (Huang et al., 2021b), FedALA (Zhang et al., 2023c)), representation-guided (GPFL (Zhang et al., 2023b)), and graph-based (pFedGAT (Zhou et al., 2025)). We also include a Local baseline where each client trains independently without any communication, to verify that collaboration provides benefit. All baselines use the hyperparameters reported in their respective original paper. They also use the same backbone per task and are run under identical conditions.

4.2 MAIN EXPERIMENT

Table 1 summarizes the comparison across datasets, modalities, and label-distribution regimes on a 20 clients setup. We evaluate on vision benchmarks (Fashion-MNIST, CIFAR-10, CIFAR-100) under both practical and pathological skew, and on AG News under practical skew only (It is composed of only 4 classes, pathological will reduce each client to a trivial task).

PFLAIR achieves the best or second-best accuracy across all settings, with the largest gains under pathological skew, the regime where class-indexed attention can most selectively promote or suppress transfer. The advantage is consistent across Fashion-MNIST, CIFAR-10, and AG News, confirming that the mechanism is not dataset dependent. On AG News, prototype-based methods struggle, suggesting that representation-space alignment is less effective for text encoders, whereas PFLAIR 's model-level descriptors remain robust. The one setting where PFLAIR does not lead is CIFAR-100 Because the mixing matrix averages class-wise heads (Equation 4), classes absent from a client contribute uninformative heads that dilute the aggregate when the label space is large. This can be adressed by masking or weighting heads with a per-client class-presence indicator; exactly the per-class-count signal that GPFL and prototype-based methods communicate to reach their best numbers (Zhang et al., 2023b; Tan et al., 2022b). We deliberately refuse this signal: class presence

Table 1: Performance comparison of different methods on Fashion MNIST, CIFAR-10, CIFAR-100, and AGNews datasets under different data heterogeneity types with 20 clients.

Method	Practical Skew — $Dir(0.5)$				Pathological Skew		
	FMNIST	**CIFAR-10**	**CIFAR-100**	**AGNews**	**FMNIST**	**CIFAR-10**	**CIFAR-100**
Local	88.30±0.24	68.95±0.22	34.21±0.17	71.99±0.18	97.20±0.12	82.15±0.20	46.12±0.18
FedAvg	86.44±0.18	43.26±0.21	29.26±0.21	68.42±0.21	96.10±0.07	37.97±0.19	22.49±0.24
FedProx	85.90±0.20	58.06±0.19	28.61±0.28	68.21±0.24	94.20±0.12	36.98±0.15	22.13±0.26
FedBABU	90.75±0.15	60.36±0.21	35.44±0.35	72.01±0.20	98.23±0.04	75.80±0.09	50.90±0.20
FedROD	90.82±0.14	63.28±0.17	33.21±0.36	72.23±0.17	99.04±0.05	80.71±0.10	48.61±0.22
FedProto	90.63±0.15	58.12±0.24	37.90±0.31	59.43±0.24	99.11±0.04	81.33±0.13	50.79±0.21
FedDBE	88.61±0.18	50.50±0.22	30.41±0.25	69.85±0.22	97.95±0.07	60.77±0.20	25.66±0.23
FedFOMO	89.61±0.17	68.43±0.19	27.80±0.39	72.03±0.19	98.88±0.05	83.69±0.21	37.45±0.30
FedAMP	87.53±0.23	69.00±0.25	26.17±0.44	71.99±0.25	98.81±0.05	85.79±0.18	38.63±0.29
FedALA	89.31±0.17	75.26±0.17	36.91±0.30	71.04±0.22	99.02±0.04	86.12±0.18	55.73±0.24
GPFL	90.95±0.15	76.85±0.18	**54.81±0.21**	72.19±0.18	99.01±0.04	87.95±0.10	59.81±0.19
pFedGAT	90.44±0.16	73.48±0.16	37.55±0.33	71.82±0.20	98.63±0.05	85.43±0.11	52.38±0.18
pFLAIR	**91.65±0.15**	**80.28±0.16**	53.87±0.28	**73.55±0.16**	**99.26±0.06**	**91.98±0.11**	**60.95±0.19**

and frequency are known to leak sensitive information about a client's label space, such as user attributes or institutional specialization (Gu & Bai, 2023), and avoiding it is central to pFLAIR 's premise. The high-class-count regime is itself atypical, as most federated deployments, from on-device tasks such as activity recognition to cross-silo applications such as disease classification or fraud detection, involve far fewer categories (Kairouz et al., 2021; Huang et al., 2022). The small accuracy cost on CIFAR-100 is minor relative to the client-scalability gains where pFLAIR excels (Section 4.4).

4.3 Effect of Heterogeneity Severity

We next vary the severity of label skew using Dirichlet partitions on CIFAR-10. Table 2 reports accuracy for $Dir(\alpha)$ with $\alpha \in \{0.1, 0.3, 0.5\}$ and 20 clients, where smaller α means that per-client label distributions become more concentrated and cross-client overlap diminishes. $Dir(0.5)$ and $Dir(0.3)$ are generally considered the most realistic proxies for real-world heterogeneity (Domini et al., 2025; Hsieh et al., 2020), while $Dir(0.1)$ represents a more extreme regime. IID is included as a standard reference point. Across all non-IID skew levels, pFLAIR achieves the best performance, with the advantage growing as heterogeneity intensifies, confirming that the attention-guided recombination adapts its mixing to the available cross-client overlap: when clients have little in common, the learned attention concentrates on a few compatible partners rather than diffusing over the population. Under IID, pFLAIR ranks second behind GPFL, an expected result as selective client-specific transfer provides limited benefit when all clients share the same distribution. This setting is largely academic: identically distributed data across clients removes the motivation for personalization.

Table 2: The accuracy (%) on CIFAR-10 for different data heterogeneity levels using 20 clients setup.

Setting	IID	$Dir(0.1)$	$Dir(0.3)$	$Dir(0.5)$
Local	56.41±0.32	78.45±0.19	73.21±0.22	68.95±0.22
FedAvg	60.67±0.18	21.38±0.11	28.32±0.15	43.26±0.21
FedProx	60.68±0.20	22.45±0.14	30.11±0.17	58.06±0.19
FedBABU	55.88±0.41	71.25±0.17	63.19±0.20	60.36±0.21
FedROD	61.91±0.27	71.50±0.13	64.33±0.20	63.28±0.17
FedProto	42.04±0.28	70.08±0.17	61.17±0.17	58.12±0.24
FedDBE	61.07±0.22	55.35±0.18	52.23±0.18	50.50±0.22
FedFOMO	60.82±0.52	80.01±0.12	72.83±0.17	68.43±0.19
FedAMP	49.78±0.60	82.32±0.16	70.79±0.19	69.00±0.25
FedALA	66.10±0.28	82.85±0.18	78.42±0.19	75.26±0.17
GPFL	**73.71±0.28**	87.43±0.12	79.21±0.16	76.85±0.18
pFedGAT	64.32±0.24	83.27±0.19	75.19±0.18	73.48±0.16
pFLAIR	67.50±0.16	**90.80±0.13**	**81.11±0.14**	**80.28±0.16**

4.4 Scalability with Number of Clients

We finally stress-test scalability by increasing the number of clients on CIFAR-10 on $Dir(0.5)$ (Table 3) as the number of client grows from 20 to 300. As the client population increases, data become more fragmented and client overlap decreases, which exacerbates the fragility of global averaging under heterogeneity. This degradation is visible across most baselines: FedAvg and FedProx collapse rapidly, and even strong pFL methods that rely on broad aggregation deteriorate substantially as increasing the number of clients injects more conflicting signals into their aggregation.

pFLAIR degrades far more gracefully and remains the best method at every scale. We attribute this to a structural limitation shared by most pFL algorithms: their effective update averages contributions from a large portion of the client set. The key mechanism is that the learned attention matrix actively suppresses knowledge sharing between incompatible clients. As the number of clients grows and more dissimilar ones are added, the attention simply assigns them low mixing weights, rather than diluting every client's model with irrelevant contributions. This selective suppression appears to contribute to the observed robustness as the federation scales. Interestingly, pFedGAT also degrades more gently than most baselines, which we attribute to a similar idea: its graph-based architecture learns to cut edges between incompatible clients, providing a structural form of the same selectivity. The difference is that pFLAIR learns this selectivity through class-conditioned attention in a fully trainable, end-to-end manner, rather than through graph topology, yielding stronger absolute performance across all federation sizes.

Table 3: The accuracy (%) on CIFAR-10 for a varying number of clients for $Dir(0.5)$ scenario

Clients Nb.	20	50	100	300
Local	68.95±0.22	57.28±0.24	45.29±0.35	38.17±0.38
FedAvg	43.26±0.21	29.80±0.30	17.03±0.47	12.50±0.52
FedProx	58.06±0.19	28.11±0.31	20.10±0.45	15.20±0.50
FedBABU	60.36±0.21	47.80±0.28	36.61±0.36	28.82±0.42
FedROD	63.28±0.17	49.10±0.29	37.97±0.36	30.21±0.42
FedProto	58.12±0.24	45.80±0.28	36.81±0.33	30.45±0.38
FedDBE	50.50±0.22	39.00±0.31	30.21±0.35	24.14±0.40
FedFOMO	68.43±0.19	50.20±0.30	38.60±0.38	30.50±0.44
FedAMP	69.00±0.25	52.60±0.31	40.80±0.39	32.43±0.45
FedALA	75.26±0.17	50.10±0.30	40.23±0.37	33.50±0.43
GPFL	76.85±0.18	57.52±0.34	47.23±0.38	40.10±0.44
pFedGAT	73.48±0.16	59.40±0.26	51.10±0.34	45.56±0.40
pFLAIR	**80.28±0.16**	**67.61±0.29**	**64.56±0.36**	**53.82±0.42**

4.5 Feature shift and real-world scenario

Table 4 reports average accuracy and intra-client standard deviation across CIFAR-10, AG News, Amazon Reviews, and HAR. Those two last datasets are particularly relevant here because they test models under heterogeneity that stems primarily from feature distribution rather than label imbalance. On Amazon Reviews, each client corresponds to a distinct product domain inducing natural feature shift. On HAR, data is partitioned across 30 users performing the same six activities; heterogeneity arises naturally from individual differences in motion patterns, sensor placement, and device characteristics, reflecting a deployment-realistic source of feature shift.

Table 4: Average accuracy (%) and inter-client standard deviation [std] (x10e-2) across datasets.

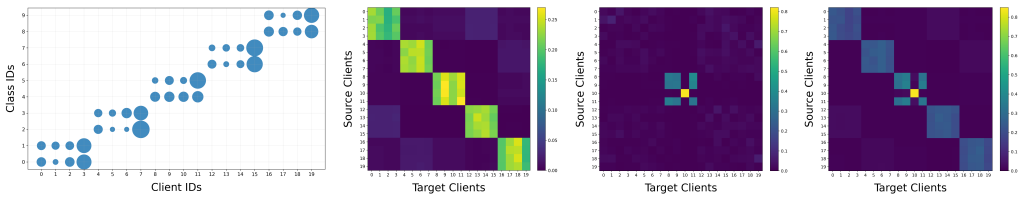
Method	**CIFAR-10**	**AG News**	**Amazon R.**	**HAR**
Local	68.95 [7.2]	71.99 [9.8]	87.41 [2.1]	91.54 **[4.8]**
FedAvg	43.26 [8.3]	68.42 [13.5]	87.33 [2.2]	87.31 [11.1]
FedProx	58.06 [8.5]	68.21 [15.5]	87.03 [1.9]	82.6 [11.2]
FedBABU	60.36 [6.7]	72.01 **[8.5]**	**88.60** [2.1]	92.05 [7.4]
FedROD	63.28 [7.5]	72.23 [13.2]	88.00 [2.1]	86.51 [9.8]
FedProto	58.12 [5.9]	59.43 [8.7]	87.27 [1.8]	93.06 [5.5]
FedDBE	50.50 [6.6]	69.85 [10.8]	87.52 [1.9]	84.88 [9.7]
FedFOMO	68.43 [8.2]	72.03 [9.7]	86.71 [2.0]	92.07 [5.8]
FedAMP	69.00 [7.8]	71.99 [9.8]	87.32 [1.8]	92.61 [5.7]
FedALA	75.26 [7.5]	71.04 [9.4]	87.38 [2.0]	91.10 [8.3]
GPFL	76.85 [5.0]	72.19 [10.1]	88.41 [2.3]	93.45 [6.1]
pFedGAT	73.48 [7.2]	71.82 [10.1]	87.41 [1.8]	91.07 [5.6]
pFLAIR	**80.28 [4.7]**	**73.55** [9.5]	88.52 **[1.7]**	**94.31** [5.6]

pFLAIR achieves strong accuracy while exhibiting among the lowest client variance across all settings, indicating that gains are distributed across the federation rather than concentrated on a few favored clients. This fairness property emerges naturally from the attention mechanism.

To illustrate this even more we have pushed it to the apex, by arbitrarily replacing one class with samples from a different category for a specific client, creating a targeted semantic mismatch at the class level. Figure 2 visualizes how pFLAIR reacts.

Figure 2 (b) shows the learned mixing matrix R in the standard (undisturbed) setting: the weights form a clearly structured block pattern, with clients sharing similar label support attending heavily to each other and low weights across incompatible groups, consistent with prior pFL works that visualize client-dependent mixing weights (Huang et al., 2021b; Zhou et al., 2025).

Figure 2 (c) and (d) present the model reaction when the class is changed (This is a different training than the one without replacement). The per-class attention concentrates on mutually compatible clients and isolates the tampered client, which attends essentially only to itself, while the other clients in the same cluster continue sharing representations with each other. The server never receives any indication that a client has been modified and this selective suppression emerges purely from disagreement in the learning phase of the attention.



(a) Data distribution (b) R (normal) (c) Specific class head (d) R (Class change)

Figure 2: Data distribution and attention-based mixing under label skew. (a) Pathological label distribution across N=20 clients for all (b), (c) and (d). (b) Learned client mixing matrix R in the standard setting. (c) Single class/head attention (class swap case) (d) Learned attention matrix R under the perturbation. Colorscale is different between figures.

5 Ablation Study

We evaluate simplified variants of pFLAIR under $Dir(0.5)$ with $N = 20$ clients on CIFAR-10 to quantify the contribution of each main component. All variants use the same backbone, training protocol, and hyperparameters.

Client-side ablations. We evaluate three client modifications: (w/o E^{att}) removes the attention embedding table and its MSE alignment loss L^{mse}; (w/o head mix) disables server-side head recombination by keeping θ^{head} unchanged across rounds; and (w/o FiLM+E^{pers}) removes FiLM conditioning on the server and drops the personalized prototype learning on clients (removing E^{pers} and L^{emb}), reducing local optimization to task-only learning.

Server-side ablations. We test three server variants: (w/o Proj.) bypasses the learned projector and computes the mixing weights directly from raw head signatures, (w/o Attn.) removes the attention mechanism by replacing learned query/key attention with a cosine-similarity based mixing rule applied to the projected descriptors Z, and finally (w/o Ent.) that removes the entropy loss from the server global loss. The ablations in Table 5 support the intended roles of the main components of pFLAIR . The largest degradation occurs when removing FiLM together with the personalized embeddings, highlighting that class-conditioned tokenization is critical for making server-side recombination effective under label skew. Disabling head mixing and removing the attention mechanism also lead to substantial drops, suggesting that (i) mixing the classifier head transfers useful decision-boundary geometry across clients and (ii) learned attention captures richer, class-dependent compatibility than simpler similarity-based alternatives. In contrast, removing the projector yields a smaller but consistent decrease, indicating that it primarily improves the conditioning and comparability of client representations. Finally, removing the entropy regularizer has only a minor impact on final accuracy, which is consistent with its role as a stabilizer that mainly improves convergence speed rather than driving the end performance.

Table 5: Ablation study of pFLAIR under $Dir(0.5)$ with N=20 clients on CIFAR-10.

Variant	**Accuracy** (%)
pFLAIR (full)	80.28±0.16
Client-side ablations	
w/o E^{att}	73.04±0.16
w/o head mix	68.09±0.17
w/o FiLM+E^{pers}	61.20±1.20
Server-side ablations	
w/o Proj.	76.08±0.30
w/o Attn.	72.37±0.18
w/o Ent.	80.12±0.20

6 Conclusion

We introduced pFLAIR , a server-centric personalized federated learning framework that achieves effective personalization across diverse heterogeneity regimes without burdening clients with heavy adaptation loops or requiring label-distribution statistics. pFLAIR shifts personalization to the server through class-conditioned representations and multi-head attention, keeping client-side training close to standard supervised learning. Across vision, language, and sensor benchmarks, pFLAIR consistently performs among the strongest pFL baselines and degrades far more gracefully as the federation scales, demonstrating the robustness essential for practical deployments.

7 AI USE STATEMENT

In this work, we used generative AI tools to assist with two aspects of the research. For the experimental code, their use was restricted to practical automation tasks, such as running batches of experiments, managing repeated iterations, and gathering and organizing results; they were not used to design or develop the PFLAIR algorithm or its underlying methodology. Second, we used them to improve the writing of the paper, including rephrasing, grammar and clarity edits, and polishing of exposition. We did not use generative AI tools for research ideation, the design of the method, the derivation of any analytical result, the design or execution of the experiments, or the interpretation of their results; these constitute the core intellectual contributions of the work and were carried out entirely by the authors.

We have reviewed all AI-assisted work. All AI-generated or AI-edited code was read, verified, and tested for correctness by the authors. All AI-assisted text was reviewed and edited by the authors to ensure it accurately reflects the methods, claims, and findings of the paper. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

We have taken several steps to ensure reproducibility of this work. The complete method is described in Section 3, with full pseudocode provided in Appendix B. All architectural choices, loss functions, and the end-to-end training procedure, including the gradient factorization that enables distributed execution (Eq. 7), are specified in the main body of the paper. The experimental setup (Section 4.1) details the datasets (publicly available), backbone architectures, data-partitioning strategies, baseline implementations, and evaluation protocol used throughout. Communication and computational costs are analyzed in Appendices C and D, respectively. The full source code, including PFLAIR 's client and server implementations and all baseline configurations, is provided in the supplementary material.

REFERENCES

Amr Abourayya, Jens Kleesiek, Bharat Rao, and Michael Kamp. Whom to trust? adaptive collaboration in personalized federated learning. *arXiv preprint arXiv:2507.00259*, 2025.

Abien Fred Agarap. Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375*, 2018. URL `http://arxiv.org/abs/1803.08375`.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks (ESANN)*, 2013.

Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations (ICLR)*, 2022.

Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. SPEAR: Exact gradient inversion of batches in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with Moreau envelopes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Davide Domini, Gianluca Aguzzi, and Mirko Viroli. Profed: a benchmark for proximity-based non-iid federated learning. 2025. doi: 10.48550/arXiv.2503.20618.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Haozhe Feng, Zhaoyang You, Minghao Chen, Tianye Zhang, Minfeng Zhu, Fei Wu, Chao Wu, and Wei Chen. KD3A: Unsupervised multi-source decentralized domain adaptation via knowledge distillation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 3274–3283. PMLR, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, chapter 15, pp. 519–559. MIT Press, 2016. URL http://www.deeplearningbook.org.

Yuhao Gu and Yuebin Bai. Ldia: Label distribution inference attack against federated learning in edge computing. *Journal of Information Security and Applications*, 74:103475, 2023.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The Non-IID data quagmire of decentralized machine learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR, 2020.

Ming Hu, Zhihao Yue, Xiaofei Xie, Cheng Chen, Yihao Huang, Xian Wei, Xiang Lian, Yang Liu, and Mingsong Chen. Is aggregation the only choice? federated learning via layer-wise model recombination. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2024.

Chao Huang, Jianwei Huang, and Xin Liu. Cross-silo federated learning: Challenges and opportunities. *arXiv preprint arXiv:2206.12949*, 2022.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on Non-IID data. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021b.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 427–431, 2017.

Peter Kairouz, H. Brendan McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019. URL http://arxiv.org/abs/1910.06378.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local SGD on identical and heterogeneous data. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Weishi Li, Yong Peng, Mengyao Du, Fuhui Sun, Xiaoyan Wang, and Li Shen. Hypernetwork aggregation for decentralized personalized federated learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015. URL https://arxiv.org/abs/1506.04757.

Connor McLaughlin and Lili Su. Personalized federated learning via feature distribution adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *International Conference on Learning Representations (ICLR)*, 2018.

Jaehoon Oh, SangMook Kim, and Se-Young Yun. FedBABU: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations (ICLR)*, 2022.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Mehdi Setayesh, Xiaoxiao Li, and Vincent W.S. Wong. Perfedmask: Personalized federated learning with optimized masking vectors. In *International Conference on Learning Representations (ICLR)*, 2023.

Jiarui Song, Yunheng Shen, Chengbin Hou, Pengyu Wang, Jinbao Wang, Ke Tang, and Hairong Lv. Fedaghn: Personalized federated learning with attentive graph hypernetworks. *Knowledge-Based Systems*, 329:114355, November 2025. ISSN 0950-7051. doi: 10.1016/j.knosys.2025.114355. URL http://dx.doi.org/10.1016/j.knosys.2025.114355.

Lili Su, Jiaming Xu, and Pengkun Yang. A non-parametric view of FedAvg and FedProx: Beyond stationary points. *Journal of Machine Learning Research*, 24(203):1–48, 2023.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 35(9):9587–9603, 2022a.

Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. FedProto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022b.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

Jiaqi Wang, Qi Li, Lingjuan Lyu, and Fenglong Ma. pFedClub: Controllable heterogeneous model aggregation for personalized federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. In *International Conference on Learning Representations (ICLR)*, 2023.

Xiyuan Yang, Wenke Huang, and Mang Ye. Dynamic personalized federated learning with adaptive differential privacy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Chen Zhang, Husheng Li, Xiang Liu, Linshan Jiang, and Danxin Wang. Hypernetworks for model-heterogeneous personalized federated learning, 2025. URL https://arxiv.org/abs/2507.22330.

Jianqing Zhang, Yang Hua, Jian Cao, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Eliminating domain bias for federated learning in representation space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Jian Cao, and Haibing Guan. GPFL: Simultaneously learning global and personalized feature information for personalized federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023b.

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. FedALA: Adaptive local aggregation for personalized federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023c.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

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.

Ziran Zhou, Guanyu Gao, Xiaohu Wu, and Yan Lyu. Personalized federated learning via learning dynamic graphs. *arXiv preprint arXiv:2503.05474*, 2025. URL https://arxiv.org/abs/2503.05474.

A NOTATION SUMMARY

Table 6: Notation summary for PFLAIR .

Symbol	Meaning
N	Number of clients; clients are indexed by $i \in [N] = \{1, \dots, N\}$.
$\mathcal{D}_i$	Client i's private local training set.
C	Number of classes.
d	Feature / embedding dimension; also used as attention scaling dimension.
x, y	Input sample and its label ($y \in \{1, \dots, C\}$).
$\phi(\cdot; \theta_i^{base})$	Client i's feature extractor (backbone / base) with parameters θ_i^{base}.
$\psi(\cdot; \theta_i^{head})$	Client i's classifier head with parameters θ_i^{head}.
θ_i	Full client model state (base and head), mixed on the server via R.
f_i	Feature vector: $f_i = \phi(x; \theta_i^{base}) \in \mathbb{R}^d$.
$\hat{y}$	Predicted label.
p_i	Head signature uploaded to the server: $p_i = \mathrm{vec}(\theta_i^{head})$.
$E_i^{pers} \in \mathbb{R}^{C \times d}$	Personalized (client-local) class embedding table; row $e_{i,c}^{pers} \in \mathbb{R}^d$.
$E_i^{att} \in \mathbb{R}^{C \times d}$	Attention-informed class embedding table returned by the server; row $e_{i,c}^{att} \in \mathbb{R}^d$.
$\mathrm{cosim}(\cdot, \cdot)$	Cosine similarity, $\frac{u^\top v}{\|u\|_2 \|v\|_2}$.
L_i^{task}	Client task loss (cross-entropy), $L_i^{\text{task}} = \ell(\hat{y}, y)$.
L_i^{emb}	Personalized embedding-guidance loss using E_i^{pers} (cosine-softmax over classes).
L_i^{mse}	Alignment loss pulling f_i toward the attention-informed embedding $e_{i,y}^{att}$.
L_i	Total client loss: $L_i = L_i^{\text{task}} + L_i^{\text{emb}} + L_i^{\text{mse}}$.
$Z_i \in \mathbb{R}^d$	Server-side client descriptor produced by the projector from p_i.
$P = [p_1; \dots; p_N]$	Stacked head signatures; $\hat{P}$ denotes the reconstruction from descriptors.
L_{recon}	Projector reconstruction loss, $\|P - \hat{P}\|_2^2$.
$\gamma_{i,c}, \beta_{i,c}$	FiLM parameters (scale/shift) produced from the class embedding of client i, class c.
$t_{i,c}$	Class-conditioned token (FiLM-modulated descriptor), $t_{i,c} = \sigma\big((1 + \gamma_{i,c}) \odot Z_i + \beta_{i,c}\big)$.
σ	FiLM activation function (ReLU).
$W_Q^{(c)}, W_K^{(c)}$	Attention projection matrices for class-head c.
$q_{i,c}, k_{i,c}$	Query/key vectors: $q_{i,c} = W_Q^{(c)} t_{i,c}$, $k_{i,c} = W_K^{(c)} t_{i,c}$.
$A^{(c)} \in \mathbb{R}^{N \times N}$	Per-class attention matrix; $A_{ij}^{(c)}$ is attention from target i to source j for class c.
$R \in \mathbb{R}^{N \times N}$	Mixing matrix, $R = \frac{1}{C} \sum_{c=1}^{C} A^{(c)}$.
Θ_S	Server-side parameters (projector, FiLM networks, attention projections).
L_{ent}	Entropy regularizer on R to control concentration/sharpness.
ℓ_i	Local loss of client i evaluated on its data after receiving the mixed model θ_i.
$\nabla_{\theta, E^{att}} \ell_i$	Local-loss gradients w.r.t. the received parameters (θ_i, E_i^{att}), returned to the server.
L_{serv}	Server training objective, $\sum_{i=1}^{N} n_i \ell_i + L_{\text{recon}} + L_{\text{ent}}$.

B ALGORITHM

Algorithm 1 summarizes one communication round of PFLAIR , organized in three phases. In the client phase, each client runs a standard supervised loop augmented with two embedding losses (L^{emb} and L^{mse}) and uploads its model state, and personalized embedding table. In the aggregation phase, the server handles all personalization: it projects head signatures into descriptors, conditions them per class with FiLM, computes class-wise attention to form the mixing matrix R, and recombines client models and embeddings directly in parameter space. Finally, in the update phase, each client evaluates the mixed model it receives and returns the local-loss gradient; the server backpropagates this signal through R to train its own modules Θ_S end-to-end (Eq. 7).

Algorithm 1 PFLAIR : one communication round

Require: Clients $\{1, \dots, N\}$; server modules $\Theta_S = \{\text{Proj}, \text{FiLM}, \text{Attn}\}$

—- Client local training —-

for each client i in parallel **do**
  for each local epoch over $\mathcal{D}_i$ **do**
    $f_i \leftarrow \phi(x; \theta_i^{base})$; $\hat{y} \leftarrow \psi(f_i; \theta_i^{head})$
    $L_i \leftarrow L_i^{\text{task}}(\hat{y}, y) + L_i^{\text{emb}}(f_i, E_i^{pers}) + L_i^{\text{mse}}(f_i, E_i^{att})$
    update $\theta_i^{base}, \theta_i^{head}, E_i^{pers}$ by SGD on L_i
  end for
  upload θ_i, and E_i^{pers} to the server ▷ ①
end for

—- Server aggregation —-

$Z, \hat{P} \leftarrow \text{Proj}(P)$, $P = [p_1; \dots; p_N]$ ▷ descriptors + reconstruction
$L_{\text{recon}} \leftarrow \|P - \hat{P}\|_2^2$
$t_{i,c} \leftarrow \sigma\big((1 + \gamma_{i,c}) \odot Z_i + \beta_{i,c}\big)$, $(\gamma_{i,c}, \beta_{i,c}) = \text{FiLM}(e_{i,c}^{pers})$ $\forall i, c$ ▷ class-aware tokens
$A_{ij}^{(c)} \leftarrow \text{softmax}_j\big(\langle W_Q^{(c)} t_{i,c}, W_K^{(c)} t_{j,c} \rangle / \sqrt{d}\big)$ $\forall c$ ▷ per-class attention
$R \leftarrow \frac{1}{C} \sum_{c=1}^{C} A^{(c)}$; $L_{\text{ent}} \leftarrow -\text{mean}(R \odot \log R)$
$\theta_i \leftarrow \sum_j R_{ij} \theta_j$, $E_i^{att} \leftarrow \sum_j R_{ij} E_j^{pers}$ $\forall i$ ▷ differentiable mixing
send mixed θ_i and E_i^{att} back to each client ▷ ②

—- Server update from client feedback —-

for each client i in parallel **do**
  evaluate ℓ_i on local data using the mixed θ_i ▷ ③
  send $\nabla_{\theta_i, E_i^{att}} \ell_i$, to the server.
end for
$L_{\text{serv}} \leftarrow \sum_i n_i \ell_i + L_{\text{recon}} + L_{\text{ent}}$
update Θ_S by descending $\nabla_{\Theta_S} L_{\text{serv}}$ via the chain rule (Eq. 7)

Table 7: Per-client per-round communication on CIFAR-10 (4-layer CNN). ϕ denotes the backbone, so $|\theta| = |\theta^{base}| + |\theta^{head}|$; M=5 for FedFOMO in standard setups. (Order from lighter to heavier)

Method	Upload	Download	Total (MB)	×FedAvg
FedProto	$\leq Cd{+}C$	Cd	0.04	0.006
FedBABU	$\lvert\theta^{base}\rvert$	$\lvert\theta^{base}\rvert$	6.99	0.99
FedAvg	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
FedProx	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
FedALA	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
FedROD	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
FedAMP	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
pFedGAT	$\lvert\theta\rvert$	$\lvert\theta\rvert$	7.03	1.00
FedDBE	$\lvert\theta\rvert{+}d$	$\lvert\theta\rvert{+}d$	7.03	1.00
PFLAIR (ours)	$2\lvert\theta\rvert{+}2Cd$	$\lvert\theta\rvert{+}Cd$	10.60	1.51
GPFL	$\lvert\phi\rvert{+}\lvert\text{CoV}\rvert{+}Cd{+}C$	$\lvert\phi\rvert{+}\lvert\text{CoV}\rvert{+}Cd$	11.25	1.60
FedFOMO	$\lvert\theta\rvert$	$M\lvert\theta\rvert$	21.08	3.00

C Communication and Data Transmission

This appendix characterizes the per-round communication of PFLAIR and compares it with representative baselines. All figures are instantiated on CIFAR-10 with the 4-layer CNN backbone ($|\theta| = 878{,}538$ parameters, of which $|\theta^{head}| = 5{,}130$ in the classifier head), feature dimension $d = 512$, and $C = 10$ classes. We assume 32-bit floats and no compression.

Each round, PFLAIR exchanges the client model in both directions (as in FedAvg), and additionally transmits the two class-embedding tables (E_i^{pers} up, E_i^{att} down, Cd floats each) and the local-loss gradient $\nabla_{\theta_i, E^{att}} \ell_i$ that trains the server modules (Section 3). This gradient is model-sized, but can be sent on the following round's upload and therefore introduces no additional communication round. Table 7 reports the resulting per-client per-round cost.

PFLAIR 's total is $3(|\theta| + Cd)$: the upload carries the model, E_i^{pers}, and the returned gradient, and the download carries the mixed model and E_i^{att}. This amounts to $1.51\times$ FedAvg, remaining below both GPFL and FedFOMO. Importantly, PFLAIR transmits no explicit class-conditioned statistics, unlike FedProto and GPFL.

D Convergence and Computation Time

We report convergence curves and wall-clock computation time across datasets. All experiments are conducted on an NVIDIA H100 GPU under identical conditions.

Convergence. On CIFAR-10 (Figure 3), we also include PFLAIR without entropy regularization (dotted line), confirming that the entropy term accelerates convergence without altering final accuracy, as discussed in Section 5. Across datasets, PFLAIR converges within the same number of rounds as standard baselines, so its advantage comes from the quality of the recombination rather than from longer training.

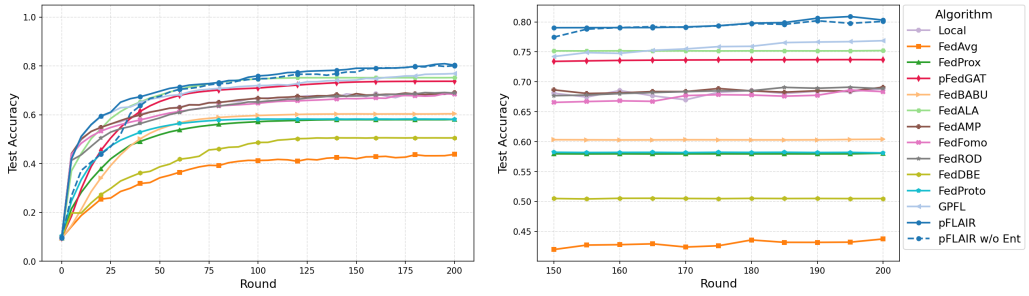
Client-side cost. The computation-time breakdown (Figures 3–6) shows that PFLAIR 's client-side cost is comparable to baselines and, crucially, independent of the federation size: each client runs the same local loop regardless of N, since all coordination is handled centrally. These measurements are on high-end hardware; in realistic deployments where clients are resource-constrained devices and not trained on high performing GPUs (phones, IoT sensors, edge nodes), absolute client-side time would be substantially longer, making methods with heavy client-side personalization loops prohibitively expensive. This is the practical motivation for PFLAIR 's server-centric design: the expensive computation is shifted to a well-provisioned central server rather than borne by every client, and because a synchronous round cannot complete until its slowest participant finishes, reducing per-client work directly shortens the round.

Server-side cost. The price of this design is a heavier server. Unlike averaging-based methods, PFLAIR recombines N models from N inputs and runs C attention heads over N clients, giving a per-round cost of $O(N^2|\theta|) + O(CN^2d)$. Table 8 places this in context: the quadratic term is shared by the adaptive-aggregation family (FedAMP, pFedGAT).

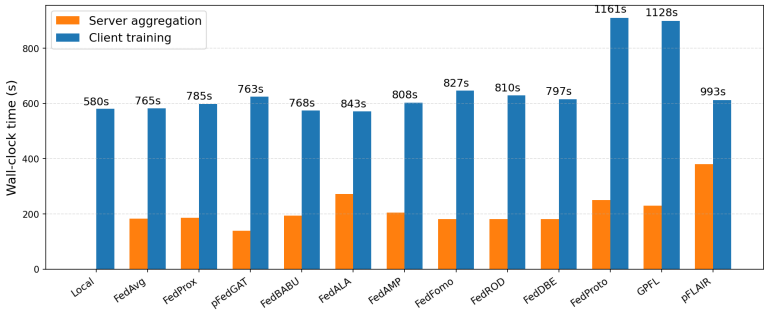
This cost remains modest in absolute terms. At N=300, the dominant mixing term is $N^2|\theta| \approx 10^{11}$ FLOPs per round, which is roughly three orders of magnitude below a single forward pass of a modern billion-parameter model on a short input, a workload routinely served on the same class of hardware. Since it is paid once per round on the server rather than on every device, it leaves the client footprint that motivates PFLAIR untouched, and it overlaps with the round's dominant cost, local client training. Where the quadratic term becomes limiting at very large N, the block-sparse structure of the learned R (Figure 2) could potentially be reduced through top-k sparsification reducing the mixing to $O(kN|\theta|)$.

Table 8: Theoretical server-side computation per round. (Order from lighter to heavier)

Method	Server cost / round
FedAvg	$O(N\|\theta\|)$
FedProx	$O(N\|\theta\|)$
FedALA	$O(N\|\theta\|)$
FedROD	$O(N\|\theta\|)$
FedBABU	$O(N\|\theta\|)$
FedDBE	$O(N\|\theta\| + Nd)$
FedProto	$O(NCd)$
GPFL	$O(N\|\theta\| + Nd^2)$
FedFOMO	$O(NM\|\theta\|)$
FedAMP	$O(N^2\|\theta\|)$
pFedGAT	$O(N^2\|\theta\|)$
PFLAIR	$O(N^2\|\theta\|) + O(CN^2d)$

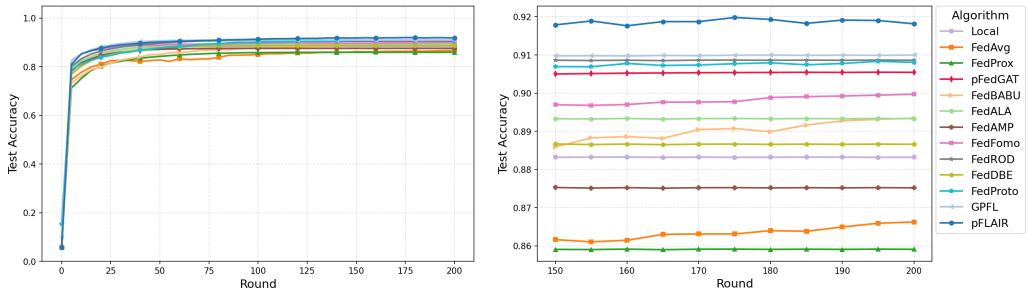


(a) Test accuracy (b) Last epochs test accuracy

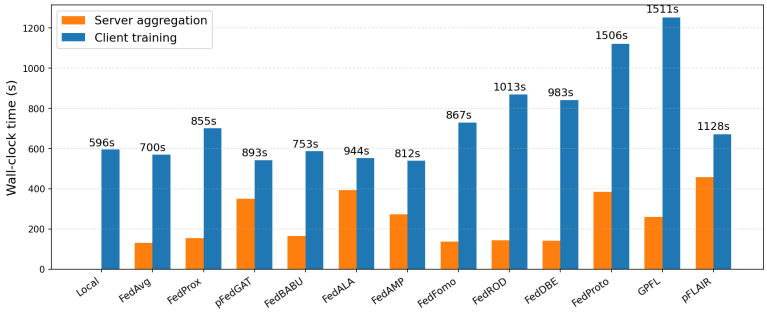


(c) Computation time

Figure 3: Accuracy convergence and computation time for the CIFAR10 dataset with 20 clients with a $Dir(0.5)$ setup scenario

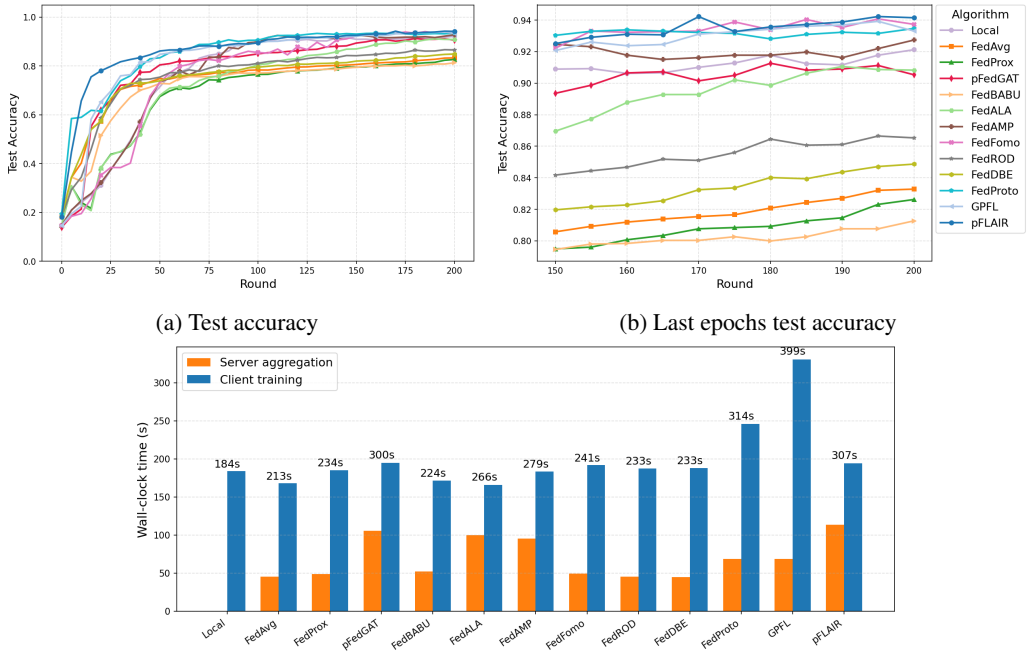


(a) Test accuracy (b) Last epochs test accuracy



(c) Computation time

Figure 4: Accuracy convergence and computation time for the Fashion MNIST dataset with 20 clients with a $Dir(0.5)$ setup scenario



(a) Test accuracy

(b) Last epochs test accuracy

(c) Computation time

Figure 5: Accuracy convergence and computation time for the HAR dataset (30 different IoT device and 6 classes)



(a) Test accuracy

(b) Last epochs test accuracy

(c) Computation time

Figure 6: Accuracy convergence and computation time for the Amazon Review dataset (4 different domains and 2 classes)

E Partial Participation

Our main experiments assume full participation, standard for the cross-silo regimes we target. To assess robustness under cross-device conditions, we evaluate a partial-participation setting on CIFAR-10 (Dir(0.5), N=20) where only a fraction $\rho \in \{0.2, 0.5, 0.8\}$ of clients is sampled each round. The server caches the descriptors and mixing rows of absent clients and refreshes them upon participation. Absent clients keep their previous personalized model for the round.

pFLAIR remains strong under partial participation, achieving the best accuracy at ρ=0.5 and ρ=0.8 and the second best at ρ=0.2, and it degrades gracefully as participation decreases. The only method competitive with it here is GPFL, which edges ahead at the most extreme rate (ρ=0.2, 64.0 vs 62.9) but is overtaken as soon as participation rises. This pattern is a direct consequence of the design: unlike methods with a fixed aggregation rule, pFLAIR learns how to recombine clients, and this learned mixing is only as informative as the set of clients observed each round. When only a fifth of the federation participates, the server sees fewer clients per round from which to estimate compatibilities, so the attention has fewer partners to combine and its modules receive a sparser training signal, naturally reducing effectiveness in the low-participation corner. The caching mechanism mitigates this (the server retains the descriptors and mixing rows of absent clients and refreshes them upon participation, preserving a coherent if partially stale view of the federation), which is why pFLAIR still stays within roughly one point of the best method at ρ=0.2 and recovers the lead as soon as participation increases.

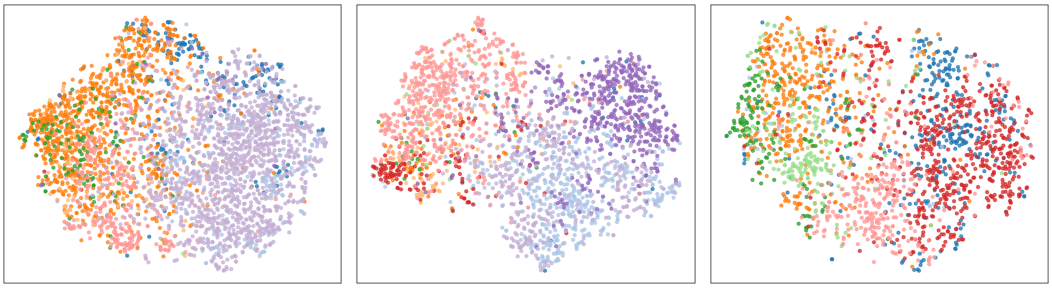
Table 9: Test accuracy (%) under partial participation on CIFAR-10 (Dir(0.5), N=20), for participation rates ρ.

Method	ρ=0.2	ρ=0.5	ρ=0.8
FedAvg	33.49±0.41	39.79±0.36	41.17±0.27
FedProx	47.64±0.25	49.40±0.21	54.36±0.19
FedBABU	61.65±0.31	67.00±0.23	66.79±0.24
FedROD	57.96±0.16	58.45±0.18	59.20±0.14
FedProto	57.43±0.32	57.99±0.35	57.90±0.29
FedDBE	40.69±0.36	45.40±0.29	47.59±0.23
FedFOMO	58.09±0.20	64.79±0.19	66.69±0.22
FedAMP	60.16±0.26	66.78±0.23	68.30±0.17
FedALA	59.26±0.45	66.70±0.38	72.74±0.31
GPFL	**64.00±0.35**	69.06±0.28	75.39±0.22
pFedGAT	61.35±0.31	68.37±0.20	71.80±0.16
pFLAIR	62.94±0.31	**72.67±0.27**	**78.35±0.21**

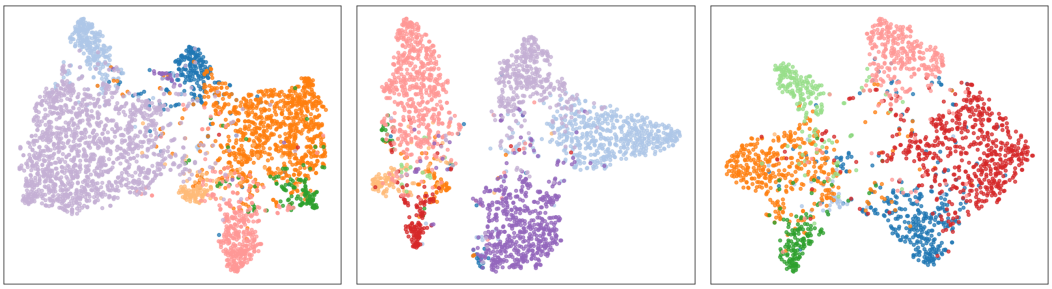
F Representation Analysis

To understand why pFLAIR improves personalized performance under label skew, we visualize the client-side feature space it induces. For three different clients, we extract the backbone features $f_i = \phi(x; \theta_i^{\text{base}})$ over their local data after convergence and project them to two dimensions with t-SNE (van der Maaten & Hinton, 2008), coloring each point by its class. Figure 7 compares pFLAIR with FedAvg and GPFL under identical conditions (same clients, perplexity, and seed).

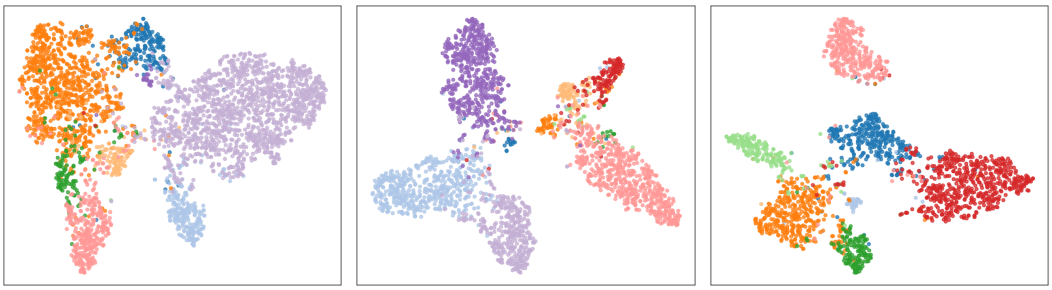
The three methods yield different local geometries, and the difference is consistent across all three clients. Under FedAvg, the single global objective forces each client toward a representation shaped by the entire federation; under label skew, where a client observes only a subset of the classes, this drags its features toward classes it barely sees and leaves the local space heavily entangled, with no clear class structure. GPFL recovers a visibly cleaner geometry: its global and personalized feature guidance imposes some organization, and distinct classes begin to occupy distinct regions. Yet the separation remains imperfect, neighboring classes bleed into one another and cluster boundaries stay diffuse, which reflects that its guidance operates through shared feature-level signals rather than a mechanism tailored to each client's own class structure. pFLAIR produces the most discriminative geometry by a clear margin: the class-conditioned recombination selectively imports decision-boundary information from compatible clients, while the embedding-guidance loss L^{emb} pulls each local feature toward its class prototype and away from the others. Together these two forces organize the space into compact, cleanly separated clusters, one per class. Crucially, this pattern holds across all three clients rather than in a single favorable case, suggesting that the observed representation structure is consistent across clients. The qualitative gap also mirrors the quantitative results of Section 4.2: a feature space in which classes are cleanly separated on each client is exactly the substrate that supports the strong personalized accuracy pFLAIR achieves under skew, and it offers a concrete visual explanation for that advantage.



(a) FedAvg



(b) GPFL



(c) pFLAIR (ours)

Figure 7: t-SNE of backbone features for three different clients after training (CIFAR-10, Dir(0.5), N=20), colored by class; each row shows one method across the three clients. Those clients are the same for the different methods.

G Discussion

Why class-wise attention helps. A natural question is why pFLAIR computes class-conditioned attention while ultimately applying a single mixing matrix to the full model parameters. A direct class-wise recombination of model parameters is not necessarily well-defined for modern deep networks. In CNNs and Transformers, parameters generally contribute to the representations of multiple classes simultaneously, particularly in early and intermediate layers that capture features shared across classes (Goodfellow et al., 2016). Assigning individual parameters to specific classes would therefore require an additional attribution mechanism, such as gradient-, activation-, or Fisher-based importance estimates, potentially computed for every client and every class. Such a decomposition would introduce substantial computational and memory overhead and would make the parameter partition dependent on an additional attribution criterion.

Moreover, completely separating model parameters by class would impose an artificial constraint on the network. Useful representations in deep models are often shared across multiple classes, and forcing class-specific parameter subsets could prevent beneficial transfer of common features. pFLAIR therefore uses class-conditioned attention to capture fine-grained differences in client compatibility across classes, while preserving a shared parameter space for model recombination. Specifically, the class-wise attention matrices $A^{(c)}$ capture potentially different compatibility patterns

across classes, but are aggregated into a single mixing matrix $R = \frac{1}{C}\sum_{c=1}^{C} A^{(c)}$ before being applied to the model parameters. Thus, class information is used to determine which clients should exchange knowledge, rather than to impose an explicit class decomposition of the network parameters.

This distinction also preserves the expressiveness of adaptive client mixing. If all class-wise attention matrices are identical, i.e., $A^{(1)} = \cdots = A^{(C)}$, then R reduces to this shared attention matrix, recovering the standard one-weight-per-client-pair formulation. The class-conditioned formulation therefore generalizes scalar client compatibility while avoiding an artificial decomposition of deep model parameters into class-specific components. This design is particularly appropriate under label skew, where client relationships may differ across classes, but the underlying feature representations can still benefit from shared parameters. This expressivity gap is not merely theoretical: the ablation replacing learned attention with a cosine-similarity scalar rule (Section 5, W/O Attn.) drops accuracy by nearly eight points, confirming that the finer-grained compatibility is used in practice.

Privacy posture. pFLAIR is an aggregation algorithm, not a privacy mechanism, and like every method in our comparison it inherits the baseline exposure of parameter-sharing federated learning: clients transmit model states, which the server never needs to invert to perform its task. Three points are worth making explicit.

First, pFLAIR communicates no explicit class-conditioned statistics: no per-class counts, proportions, or labeled prototypes, unlike prototype- and representation-guided methods (e.g., FedProto, GPFL) whose best performance relies on sharing such quantities, which are known to leak information about a client's label distribution (Gu & Bai, 2023). Whatever residual signal pFLAIR 's uploads carry about class structure is at most indirectly inferable, whereas these methods expose it by construction.

Second, the returned local-loss gradient is a poor target for reconstruction attacks. It is not a per-example gradient but a single gradient accumulated over the client's entire local set, and gradient-inversion attacks degrade sharply as the number of aggregated samples grows: even exact-recovery methods are limited to batches of a few dozen to at most a hundred samples on very simple architectures (Dimitrov et al., 2024), while optimization-based attacks additionally assume knowledge unavailable here, such as per-batch normalization statistics or ground-truth labels, and weaken substantially without them (Huang et al., 2021a). A gradient aggregated over the full local dataset is substantially different from the small-batch settings typically studied in existing gradient-inversion attacks, and does not introduce a fundamentally different communication primitive from the model-related information already exchanged in parameter-sharing FL.

Third, the standard privacy toolkit composes with pFLAIR without any architectural change: client-level differential privacy (clipping and noise on the uploaded quantities) applies to our summaries and returned gradient exactly as it applies to the model updates every federated method already transmits (McMahan et al., 2018).

On convergence guarantees. pFLAIR is evaluated empirically, and we do not provide a formal convergence proof. This reflects a genuine difficulty rather than an oversight: convergence results in federated learning typically require unrealistic assumptions: (strong) convexity, bounded gradients, and bounded gradient dissimilarity across clients. The optimization literature itself has criticized as stringent and, in the label-skew regime we target, largely unrealistic (Khaled et al., 2020; Su et al., 2023). pFLAIR compounds this difficulty in two ways: the aggregation rule is not fixed but learned through an outer objective, giving the overall procedure a bilevel structure, and the mixing is applied directly in parameter space. Characterizing the convergence of learned, data-dependent aggregation is an open problem for the entire adaptive-aggregation family, not specific to pFLAIR , and we regard it as an important direction for future work.

Further limitations. Two additional limitations bound the current scope. pFLAIR assumes a shared model architecture across clients, standard in pFL but restrictive when clients deploy heterogeneous backbones; extending the projector to map heterogeneous parameter spaces into a common descriptor space is a promising avenue. Second, because the mixing matrix averages class-wise heads, performance degrades when the number of classes is large and many are absent per client (Section 4.2), a regime we argue is atypical of real deployments but which a class-presence-aware weighting could mitigate his limitation, at the cost of explicitly introducing class-presence information into the aggregation process.