

Rethinking Personalized Reward Modeling for LLMs under Preference Heterogeneity via Group-Debiased Federated Learning

Seongyeon Kim
Institute of Engineering Research,
Korea University
Seoul, Republic of Korea
curisam@korea.ac.kr

Boryeong Cho
Kim Jaechul Graduate School of AI,
KAIST
Seoul, Republic of Korea
venntum@kaist.ac.kr

Jihwan Oh
Kim Jaechul Graduate School of AI,
KAIST
Seoul, Republic of Korea
erico929@kaist.ac.kr

Seokhyun Chung[†]
Industrial Management Engineering,
Korea University
Seoul, Republic of Korea
csh8901@korea.ac.kr

Se-Young Yun[†]
Kim Jaechul Graduate School of AI,
KAIST
Seoul, Republic of Korea
yunseyoung@kaist.ac.kr

ABSTRACT

Large language models are increasingly aligned to human preferences via reward modeling, but user preference data are sensitive and often cannot be centralized. Federated learning keeps such data local while learning a shared initial reward model, which is later personalized for each client through local fine-tuning. Because users often assign opposite labels to the same pair of responses, existing federated methods address preference heterogeneity by clustering similar clients and training one reward model per group, assuming that each group requires its own initialization. We show that this assumption is unnecessary. Under balanced preference groups, a single FedAvg model, despite starting at nearly random accuracy, surpasses reward models trained separately for each ground-truth group after only a few local optimization steps. We attribute this phenomenon to the flatness of the shared initialization: averaging across all clients learns richer shared representations that distinguish responses while canceling conflicting preference directions, leaving the model near a decision boundary that can be rapidly adapted. Group imbalance breaks this effect as the cancellation becomes asymmetric and leaves minority clients too far from the boundary to recover. Motivated by this observation, we propose **FedGD** (Federated Learning with Group Debiasing), which discovers latent preference groups during federated training and learns a single reward model using group-debiased client sampling. By counteracting the effect of group imbalance, **FedGD** learns an initialization that remains highly adaptable, enabling effective personalization without prior knowledge of the underlying groups.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in human-facing systems, where alignment with human preferences is crucial to user safety and satisfaction [1–3]. A standard alignment pipeline typically trains a *reward model* to predict human preferences and then optimizes the LLM through RLHF methods [32, 39]. However, preference data are highly sensitive and often cannot be centralized due to data-protection regulations [17, 36] and cross-jurisdictional

sharing constraints [23]. Federated learning (FL) [20, 25, 29] offers a practical alternative: data remain local while a coordinating server aggregates model updates from decentralized clients.

Existing FL-based preference alignment methods [12, 49] struggle with *preference heterogeneity* because they ultimately produce a single global LLM. For instance, FedRLHF [12] shows clear performance degradation under severe heterogeneity. Even FedBiscuit [49], which uses multiple server-side reward models to label unlabeled response pairs, aggregates these signals into a single consensus to train one LLM via DPO [34]. Consequently, existing methods fail to represent individual user preferences, underscoring the need for *personalized reward models* tailored to each client.

Recent personalized FL (PFL) methods [10, 21, 22, 30] widely adopt a two-stage strategy: pre-training a single shared model globally and fine-tuning it locally per client. While effective under standard data heterogeneity, where clients share label consensus despite input distribution shifts, this approach struggles under preference heterogeneity, where different clients may assign opposing labels to the same response pair. This fundamental conflict raises a question about whether a single shared initialization can still provide an effective starting point for personalized reward modeling.

Existing federated methods that explicitly address conflicting preferences or tasks instead avoid this question by clustering similar clients and training one single model per group. For example, FedBiscuit [49] groups clients according to preference similarity, while FedLEASE [48] clusters clients across related tasks, so that each client is initialized from a model that never averages over conflicting objectives. However, whether these group-specific initializations actually lead to better personalized reward models than a single shared initialization remains unknown. **We therefore ask: what makes a good initialization for personalized reward modeling under preference heterogeneity?**

We find that a group-specific initialization is not the answer. Under balanced preference groups, a single global model trained with FedAvg [29] performs at the level of random guessing before adaptation, yet within a few local steps for personalization, it quickly surpasses both centralized training (CL) and a separate centralized model per ground-truth preference group (Multi-CL).

[†]Corresponding authors.

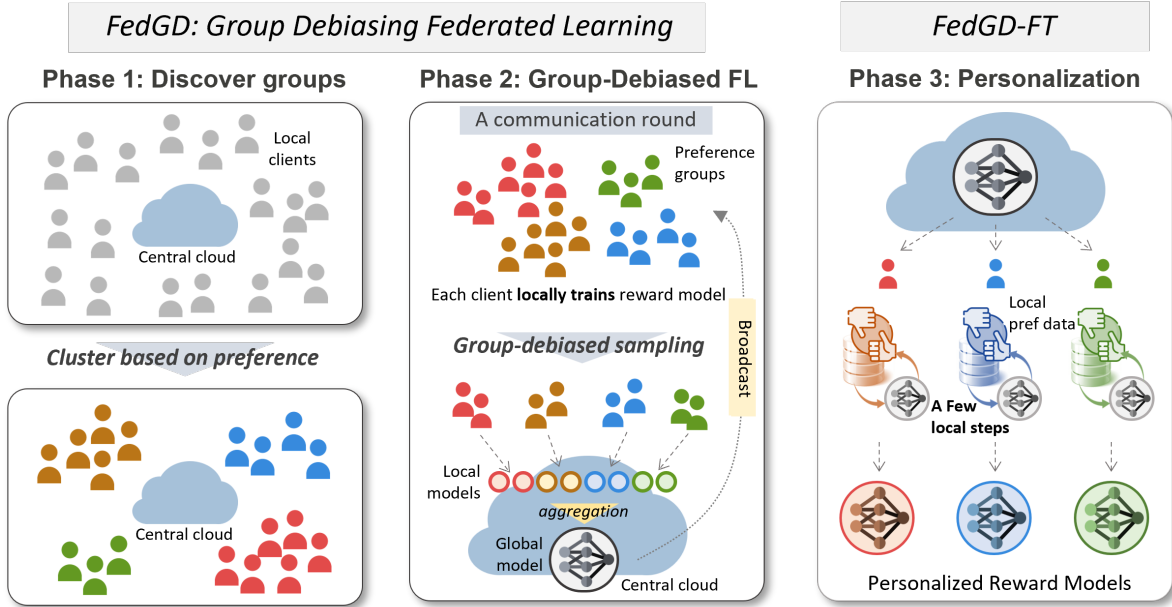


Figure 1: Overview of FedGD. Phase 1 discovers the preference groups, which can differ in size. Phase 2 trains a single reward model over them with group-debiased client sampling, which compensates for group-size imbalance during client selection. The resulting model serves as a shared initialization for personalization. Phase 3 personalizes the initial model for each client through a few local optimization steps, yielding FedGD-FT.

Our key hypothesis is that preference learning contains two distinct components. The first is *shared knowledge*: identifying the latent feature that distinguishes the two responses. All clients agree on this distinction, so collaborative FL reinforces a common representation of it. The second is *client-specific knowledge*: deciding which side of the latent feature should be preferred. Since this preference differs across clients, no single decision boundary can be optimal for everyone, explaining the poor pre-adaptation accuracy of the global model trained by FL. Consequently, the goal of federated learning should not be to produce the final personalized model, but rather an initialization that captures the shared knowledge while remaining readily adaptable to each client’s preference.

To diagnose such an initialization, we adopt the *Gradient Quotient* (GQ) [7], which measures how rapidly the local gradient changes. A small GQ indicates that local optimization remains stable, allowing successive personalization steps to reinforce one another. Such an initialization is therefore well suited for rapidly adapting to each client’s local preferences. In our experiments, FedAvg converges to initializations with substantially lower GQ than CL or Multi-CL, which explains why it serves as a better starting point for personalization despite its low start accuracy.

However, when preference groups are imbalanced, the cancellation becomes asymmetric, causing the shared model to become biased toward the majority preferences and reducing its ability to serve as a good initialization for minority clients. We find that balancing client sampling across preference groups during federated training restores the personalization capability of the resulting initialization. By this key observation, we propose **FedGD** (Federated

Learning with Group Debiasing), which first discovers the preference groups during FL and then trains a single reward model under group-debiased sampling over the discovered partition (Figure 1).

Our contributions are as follows:

- We introduce an optimization-based perspective on what makes a good initialization for efficient personalization in reward modeling: an initialization adapts well when the loss around it is flat, so that successive local steps reinforce one another. We measure this flatness with GQ , which tracks how much the local gradient changes after a single step.
- We construct a synthetic benchmark with configurable preference groups and group-size imbalance, enabling controlled analysis for personalized reward modeling under federated settings.
- We show that, under balanced preference groups, a single FedAvg model provides a better initialization for personalization than both CL and Multi-CL despite its poor pre-adaptation accuracy, and identify group imbalance as the condition under which this advantage weakens.
- We propose **FedGD**, which discovers latent preference groups during federated learning and performs group-debiased client sampling to recover high-quality initializations without requiring prior knowledge of the true preference groups.

2 PRELIMINARIES

We study federated personalized reward modeling, where clients collaboratively train initial global reward models on decentralized pairwise preferences and then fine-tune them locally. We first formalize this workflow (Section 2.1) and then propose two key metrics

to evaluate the personalization potential of initial global models (Section 2.2). Finally, Section 2.3 details the experimental setup.

2.1 Federated Reward Modeling and Personalization

We study FL [29] over C clients with pairwise preference data. Each client c holds a local dataset $D_{\text{train}}^c = \{(x_i^c, y_i^{c,+}, y_i^{c,-})\}_{i=1}^{N_c}$, where x_i^c is a prompt, $y_i^{c,+}$ the preferred response, and $y_i^{c,-}$ the less-preferred one. Let $r_\theta(x, y)$ denote the reward the model assigns to response y for prompt x . For a preference (x, y^+, y^-) , the *reward margin* $m_\theta(x, y^+, y^-) = r_\theta(x, y^+) - r_\theta(x, y^-)$ is the signed gap between the two responses, so the prediction is correct when $m_\theta > 0$ and incorrect when $m_\theta \leq 0$. Clients minimize the pairwise preference loss $\ell_\theta = -\log \sigma(m_\theta)$, which drives the margin positive, giving the full-batch training loss $\mathcal{L}_c(\theta) = \frac{1}{N_c} \sum_i \ell_\theta(x_i^c, y_i^{c,+}, y_i^{c,-})$ of client c .

FL proceeds over R communication rounds. At the beginning of round r , the server broadcasts the current global parameters $\theta^{(r-1)}$ to a sampled subset of clients $S_r \subset [C]$. Each selected client $c \in S_r$ performs τ local training iterations on D_{train}^c with batch size B , and returns updated parameters $\theta_c^{(r)}$. The server aggregates these local updates via weighted averaging [26, 49] to obtain the new global parameters $\theta^{(r)}$. The description above corresponds to a single-global design in which the server maintains a single parameter vector $\theta^{(r)}$. Some of the designs we compare instead maintain K global models $\{\theta_k^{(r)}\}_{k=1}^K$ on the server and apply the same broadcast-update-aggregate procedure independently to each model over its associated subset of clients.

After R training rounds in the *global* phase, we run a separate *personalization* phase. Each client c receives a single initialization θ_c^{init} , fine-tunes it on its own preference data D_{train}^c without further communication, and obtains a personalized reward model θ_c^{PFL} . Since local adaptation runs on the client, where compute is limited, the number of local steps an initialization requires is itself a cost. We therefore report accuracy after a small number of local steps alongside the accuracy eventually reached. How θ_c^{init} is constructed is the design choice we study, and we specify it for each method in Sections 3 and 5. Throughout the paper, we refer to the models that serve as initializations for personalization as *global* models and to θ_c^{PFL} as the resulting *personalized* reward model for client c .

2.2 Quantifying Personalization Capability

Given a reward model θ as the *initialization* for personalization, we characterize its personalization capability by estimating how effectively a small number of local updates can improve it. For this, we introduce two diagnostics: (i) *Gradient Quotient* (GQ) [7], a model-level measure of *flatness* that assesses whether consecutive local steps reinforce one another, and (ii) *Headroom* (H), a per-sample measure that we propose to assess whether a local update moves an individual sample toward its correct preference. Both quantities are computed using each client’s local preference data. **Gradient Quotient (GQ): flatness of the initialization.** A good initialization should be one from which local updates consistently move toward the client’s optimum rather than changing gradient dramatically after one update. We measure this property

using the GQ [7], which quantifies how much the gradient changes after a single optimization step.

Let $g^c(\theta) = \nabla \mathcal{L}_c(\theta)$ be the gradient of client c ’s full-batch training loss and $g^{c,l}$ its restriction to the LoRA parameters θ^l of layer l . The layer-wise GQ is

$$GQ(c, l) = \frac{1}{\#\theta^l} \left\| \frac{g^{c,l}(\theta - \eta g^c(\theta))}{g^{c,l}(\theta)} - \mathbf{1} \right\|_1 \approx \frac{\eta}{\#\theta^l} \left\| \frac{[\nabla^2 \mathcal{L}_c(\theta) g^c(\theta)]^l}{g^{c,l}(\theta)} \right\|_1,$$

where $\#\theta^l$ is the number of LoRA parameters in layer l . The first-order approximation shows that GQ is determined by the Hessian–gradient product, which captures the local curvature along the update direction. Consequently, GQ increases with the curvature encountered by gradient descent. A small GQ indicates that the gradient remains nearly unchanged in both direction and magnitude, allowing successive local updates to remain aligned and accumulate toward the client’s optimum. In contrast, a large GQ indicates that the gradient is rapidly distorted after a single step, causing subsequent steps to deviate from the original descent direction and partially cancel earlier progress, thereby reducing the effectiveness of local personalization.

We obtain a layer-wise statistic by averaging over clients, $GQ(l) = \frac{1}{C} \sum_c GQ(c, l)$ over the 24 LoRA-adapted layers of Qwen2-0.5B [50], our base model (Section 5). Studies of few-step adaptation commonly separate a shared representation body from task-specific upper layers, both in meta-learning [31] and in personalized FL [30], so we report a Body average $GQ_{\mathcal{B}}$ over the front 16 layers and a Head average $GQ_{\mathcal{H}}$ over the back 8.

Headroom (H): one-step correctability of a sample. GQ measures whether local updates are stable, not whether they lead to the right direction. We define *Headroom* to answer the latter for a single sample. Starting from the initialization under evaluation, $\theta_0 = \theta_c^{\text{init}}$, we take one full-batch *training* step on client c ’s data, $\theta_1 = \theta_0 - \eta g^c(\theta_0)$, and define the Headroom as the first-order effect of this step on the margin of a held-out *test* sample (x, y^+, y^-) of the same client:

$$H(x, y^+, y^-) = -\eta \langle \nabla_\theta m_\theta|_{\theta_0}, g^c(\theta_0) \rangle \approx \Delta m,$$

where $\Delta m = m_{\theta_1} - m_{\theta_0}$ is the true margin change. A positive H means one training step moves the held-out sample toward its correct preference, and a larger H indicates more room to move it.

We compute H for every sample in the client’s held-out test split. Since correctability depends on whether a sample is already correct, we partition the test set at θ_0 into the *retention* set $\mathcal{R} = \{m_{\theta_0} > 0\}$ and the *correction* set $\mathcal{W} = \{m_{\theta_0} \leq 0\}$, and report for each set the mean Headroom H^0 and the fraction of samples with positive Headroom H^+ , averaged over clients.

2.3 Datasets and Preference Groups

We evaluate on two datasets: a real-world set with natural annotator preferences, and a synthetic set with controlled style heterogeneity. For each client, we reserve 50 samples each for validation and testing, and use the rest for training.

Real-world dataset. We use the Reddit TL;DR summarization dataset [43, 46], which contains human preference annotations along with client IDs. We filter 144,502 samples for 34 clients who participate in all three train/valid1/valid2 splits with at least 100

Table 1: Diagnostics and personalized accuracy under balanced settings. $GQ(l)$: mean $\pm$ std over the 24 layers, with Body/Head averages GQ_B, GQ_H . R/W : test samples initially correct / wrong at θ_0 . m^0 : mean reward margin at θ_0 . H^+ : fraction with positive Headroom. P^{10} : fraction correct after 10 local steps. acc^0 in parentheses. Highlighting the best acc^{10} and bold the best per column.

method	Flatness ($\downarrow$)			Margin (m^0)		Retention (R , %, $\uparrow$)		Correction (W , %, $\uparrow$)		Accuracy (%, $\uparrow$)
	$GQ(l)$	GQ_B	GQ_H	m_R^0	m_W^0	H_R^+	P_R^{10}	H_W^+	P_W^{10}	acc^{10} (acc^0)
CL	1.14 ± 0.24	1.29	0.84	+16.00	-15.77	34.18	82.32	62.02	21.49	52.15 (50.45)
Multi CL	0.86 ± 0.23	1.00	0.56	+22.88	-12.39	13.01	97.28	59.51	25.77	91.45 (92.20)
FL	0.10 ± 0.01	0.11	0.09	+2.38	-2.34	56.94	94.01	96.00	91.99	93.45 (50.40)

samples each. Each client corresponds to an actual human annotator, and the dataset inherently exhibits natural data imbalance and genuine preference heterogeneity, with the underlying preference groups being *unknown*.

Synthetic dataset. We gather prompts by combining UltraFeed-back [6] and p-Soups [19], resulting in 52K unique prompts. Using GPT-4o-mini,¹ we generate two candidate responses per prompt along two style axes—(1) *Elementary* vs. *PhD-level* and (2) *Humorous* vs. *Non-humorous*—allocating 26K prompts per axis. We then simulate 40 clients (650 samples per axis), grouped into the four preference groups formed by the two axes: (G1) Elementary/Humorous, (G2) Elementary/Non-humorous, (G3) PhD/Humorous, and (G4) PhD/Non-humorous. Clients are indexed by group, starting with G1 and progressing through G4 in sequence.

Group-size balance. Using the synthetic dataset, we evaluate how effectively the initial shared model adapts to individual clients under varying group-size distributions. We consider two group-size configurations across G1, G2, G3, and G4:

- **Balanced (10/10/10/10):** 10 clients per group, maintaining an equal balance across both preference axes.
- **Imbalanced (15/15/5/5):** the Elementary-dominant groups (G1, G2) are overrepresented while the PhD-dominant groups (G3, G4) are underrepresented, naturally biasing the shared model toward the majority preference.

3 A SINGLE FEDERATED MODEL SUFFICES, UNTIL GROUP IMBALANCE

In this section, we compare three initializations in terms of the personalization capability of the resulting models. Interestingly, our finding indicates that FL provides a stronger initialization when preference groups are well balanced. Results show that this advantage arises because the local training loss is relatively flat around the initialization, while most test samples retain positive Headroom toward the correct preference. However, this benefit weakens under group imbalance, where FL becomes increasingly biased toward majority preferences and adaptation slows sharply. These findings motivate **FedGD**, our proposed approach.

Compared initializations. We compare 3 ways of producing the reward model that each client fine-tunes locally, trained for 200 communication rounds with the same number of updates.

- **CL:** pools all preference data on one server and trains a single centralized model, so conflicting preferences are mixed *within every batch*.

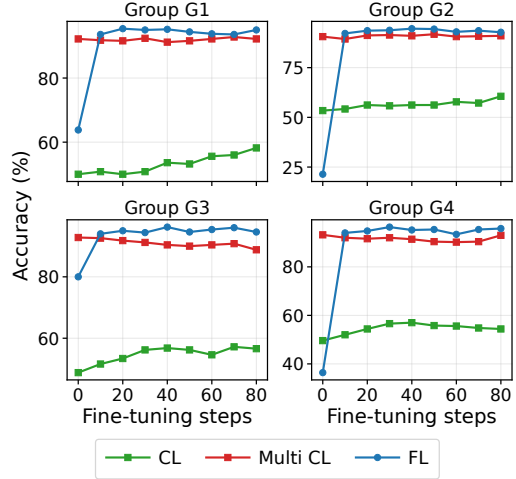


Figure 2: Group-wise personalized accuracy under the balanced setting, evaluated every 10 steps up to 80 local fine-tuning steps.

- **FL:** trains a single global model with FedAvg [29], so the conflict is concentrated *at aggregation*.
- **Multi-CL:** trains one centralized model per ground-truth preference group, so *no conflict arises within a model*. Each client is initialized from its own group’s model. Since the true groups are unknown in practice, Multi-CL is a reference point rather than a deployable method.

3.1 FL Suffices Under Balanced Groups

Under the balanced setting, FL shows the best initialization for personalization. Let acc^0 denote the accuracy of an initialization before any local update and acc^{10} the accuracy after 10 local fine-tuning steps, both averaged over clients. Before fine-tuning, Multi-CL reaches $\text{acc}^0 = 92.20$, while CL and FL reach only 50.45 and 50.40, the accuracy of random guessing on binary preference pairs (Table 1). Figure 2 shows that this average hides a wide spread for FL, whose initial accuracy ranges from about 20% on G2 to about 80% on G3. However, just 10 local steps change the picture: FL reaches $\text{acc}^{10} = 93.45$ and overtakes Multi-CL, which ends at 91.45, slightly below where it started, while CL improves only to 52.15. The same pattern appears in every group: FL gains almost all of its improvement within 10 steps, whereas CL improves only gradually over 80 steps and Multi-CL changes little or slightly declines.

FL gains the most from a few local steps despite its low initial accuracy, and Table 1 attributes this to a loss that is flat across all

¹<https://platform.openai.com/docs/models/gpt-4o-mini>

Table 2: Diagnostics and personalized accuracy under the imbalanced setting (15/15/5/5); the first row repeats balanced FL for reference. $GQ(l)$: mean $\pm$ std over the 24 LoRA layers, with Body/Head averages GQ_B, GQ_H . R/W : test samples initially correct / wrong at θ_0 . m^0 : mean reward margin at θ_0 . H^+ : fraction with positive headroom. P^{10} : fraction correct after 10 local steps. Highlighting marks the best acc¹⁰; bold the best per column among the imbalanced rows.

method	Flatness ($\downarrow$)			Margin (m^0)		Retention (R , %, $\uparrow$)		Correction (W , %, $\uparrow$)		Accuracy (%, $\uparrow$)
	$GQ(l)$	GQ_B	GQ_H	m_R^0	m_W^0	H_R^+	P_R^{10}	H_W^+	P_W^{10}	acc ¹⁰ (acc ⁰)
FL (balanced)	0.10 $\pm$ 0.01	0.11	0.09	+2.38	-2.34	56.94	94.01	96.00	91.99	93.45 (50.40)
CL	1.09 $\pm$ 0.28	1.25	0.76	+20.41	-19.42	25.72	86.50	69.61	20.14	60.50 (59.85)
Multi CL	0.95 $\pm$ 0.25	1.11	0.63	+23.96	-11.79	10.44	97.60	58.42	26.14	91.50 (92.10)
FL	0.72 $\pm$ 0.25	0.88	0.39	+27.90	-27.02	3.08	99.11	96.74	2.09	61.75 (61.80)
FL_target	0.27 $\pm$ 0.02	0.28	0.24	+3.59	-3.65	75.78	93.42	93.44	90.62	92.35 (54.95)

layers around the FL model. The flatness follows from how far each initialization has already committed. Every client must detect what distinguishes the two responses—here the style axis—to fit its own labels, and what differs is only which side it prefers, so averaging the locally trained models reinforces the shared detection while the opposing side choices offset one another. The resulting model therefore sits close to the decision boundary, with initial margins of only $m_R^0 = +2.38$ and $m_W^0 = -2.34$, where the loss $-\log \sigma(m)$ is far from saturation and its gradient changes slowly. CL cannot separate detecting the axis from choosing a side, since conflicting preferences enter the same batch, and Multi-CL removes the conflict entirely by training each model on a quarter of the clients that share one direction; both commit to a side and reach $m_R^0 = +16.00$ and $+22.88$, deep in the saturated regime where a single step distorts the gradient. This is reflected in the diagnostics: FL is an order of magnitude flatter than both baselines at every layer ($GQ_B = 0.11$ vs. 1.29 and 1.00; $GQ_H = 0.09$ vs. 0.84 and 0.56), and the gap is largest in the Body, the layers that detect the axis.

On the flat FL initialization, fine-tuning corrects wrong predictions without sacrificing the correct ones. For FL, 96.00% of the wrong set W carries positive headroom (H_W^+), and after 10 steps 91.99% of W is answered correctly (P_W^{10}), while 94.01% of the right set R stays correct (P_R^{10}). CL and Multi-CL start with less headroom on W (62.02% and 59.51%) and correct only 21.49% and 25.77% of it, even though Multi-CL keeps slightly more of R than FL (97.28%).

3.2 Group-Debiased Sampling Restores FL Under Imbalance

Under group imbalance, FL adapts more slowly. Although FL begins with a substantially higher accuracy than in the balanced case, 10 local steps leave it essentially unchanged. FL starts in acc⁰ = 61.80, compared with 50.40 under balanced groups, yet reaches only acc¹⁰ = 61.75 (Table 2). Figure 3 shows that the initial accuracy again varies widely across groups, from near 10% on G4 to 98% on G1, but the gains from fine-tuning are smaller than under balanced groups, where every group rose above 90% within 10 steps. G2, G3, and G4 instead climb slowly and stay below G1 even after 80 steps.

To test the hypothesis that the degradation in personalization is caused by group imbalance rather than the FL procedure itself, we consider a group-debiased sampling strategy that prevents any preference group from being underrepresented in a communication round. Specifically, we modify FL only in the client sampling step while leaving the rest of the training procedure unchanged, denoted by FL_target. In our experiments, each round samples $|S_r| = 5$

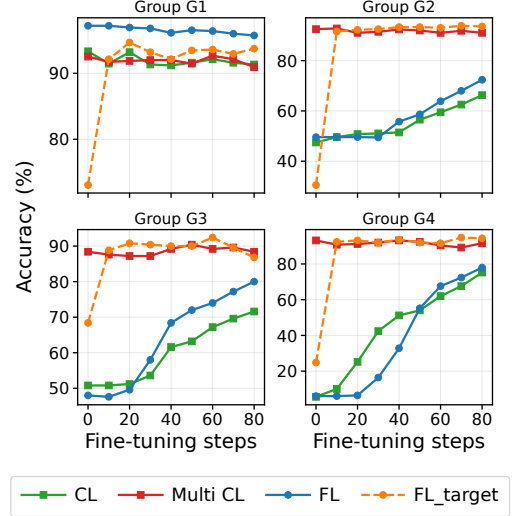


Figure 3: Group-wise PFL accuracy under the imbalanced setting for the four methods (CL, Multi CL, FL, and FL_target), evaluated every 10 steps up to 80 fine-tuning steps.

clients: one from each of the four groups and one additional client chosen uniformly at random from all groups. This simple modification lifts accuracy from acc⁰ = 54.95 to acc¹⁰ = 92.35, a level standard FL does not reach within the same number of local steps (acc⁰ = 61.80 to acc¹⁰ = 61.75) (Table 2). Figure 3 shows FL_target exceeding 90% on every group within 10 local steps, a behavior FL showed only in the balanced setting. On this dataset, FedGD with $K = 4$ recovers the four ground-truth groups exactly, so FL_target coincides with FedGD (Section 4).

Group-debiased sampling works by keeping the loss uniformly flat across layers, unlike uniform sampling. Under FL, the Body degrades far more than the Head, with GQ_B rising from 0.11 in the balanced setting to 0.88 and GQ_H from 0.09 to 0.39 (Table 2). Correction suffers as well: 96.74% of the wrong set W still carries positive headroom, yet only 2.09% of W is answered correctly after 10 steps, against 91.99% in the balanced setting. The direction of the update is therefore right but its reach is not: imbalanced FL commits to the majority side and starts at $m_W^0 = -27.02$, an order of magnitude beyond the balanced case (-2.34), so ten steps cannot close the gap. The right set R changes little instead, with 99.11% staying correct but only 3.08% carrying positive headroom.

FL_target starts lower than FL at $\text{acc}^0 = 54.95$, yet keeps the Body and the Head close ($GQ_B = 0.28$, $GQ_H = 0.24$) and stays near the decision boundary ($m_{\mathcal{W}}^0 = -3.65$ against -27.02 for FL), raises $H_{\mathcal{R}}^+$ from 3.08% to 75.78%, and recovers correction to $P_{\mathcal{W}}^{10} = 90.62\%$. Group-debiased sampling is therefore what lets FL adapt within a few local steps under group imbalance. However, FL_target assumes access to the true preference groups, which are unknown in practice. This naturally raises the following question: “Can group debiasing be achieved without knowing the true preference groups?”

4 FEDERATED LEARNING WITH GROUP DEBIASING

In this section, we propose **FedGD**, which applies group debiasing when the true groups are unknown. **FedGD** proceeds in two phases: (1) discovering the groups during federated learning, and (2) training a single reward model with group-debiased sampling over the discovered groups. Each client then fine-tunes the resulting model following the personalization phase of Section 2.1, yielding the personalized reward models we denote by **FedGD-FT**. The full algorithm is given in Appendix B.1.

4.1 Phase 1: Discovering the Groups

We discover the groups with clustered FL [14, 38], which partitions clients by training several models and letting each client join the one that fits its data best. Over the first $R/2$ rounds, the server maintains K expert models $\{\theta_k\}_{k=1}^K$ together with a reference model θ_g , and each client is assigned to one expert, forming the groups $\{\mathcal{A}_k\}_{k=1}^K$. Every T rounds the clients are reassigned, and in the rounds between two reassignments the experts are trained on the clients currently assigned to them. After round $R/2$ the clients are reassigned once more, and the resulting groups $\{\mathcal{A}_k^*\}_{k=1}^K$ are frozen; empty clusters are discarded, so K denotes the number of discovered groups. Neither the experts nor θ_g is used as an initialization for personalization; they serve only to discover the groups.

Reassigning clients ($\pi(c)$). Clients start from a uniform random assignment. Every T rounds each client evaluates all K experts on its own validation split and is reassigned to the expert with the lowest validation loss [49],

$$\pi(c) \leftarrow \arg \min_k \text{ValLoss}(\theta_k, D_{\text{val}}^c), \quad \mathcal{A}_k = \{c : \pi(c) = k\},$$

where $\pi(c)$ denotes the expert assigned to client c . Each reassignment also resets every expert to the current reference model, $\theta_k \leftarrow \theta_g$, so that the experts do not inherit what they learned from the clients they had before.

Updating the reference model (θ_g). Each round draws clients with group-debiased sampling over the current groups $\{\mathcal{A}_k\}_{k=1}^K$. Specifically, clients are sampled from each group with probability inversely proportional to the group’s cardinality, so that the resulting client set has a uniform group composition regardless of group size. We denote the resulting participating set G_r , the group-debiased counterpart of the sampled set S_r of Section 2.1. The reference model averages the returned models uniformly,

$$\theta_g^{(r)} = \frac{1}{|G_r|} \sum_{c \in G_r} \theta_{\pi(c), c}^{(r)}.$$

Here θ_g serves as a neutral point to which the experts are reset rather than as a loss minimizer, and every client takes the same τ local steps, so weighting by dataset size would bias the discovered partition toward data volume rather than preference.

Training the experts (θ_k). Every selected client $c \in G_r$ receives the expert $\theta_{\pi(c)}^{(r-1)}$ of its own group, trains it on its local data, and returns the updated model $\theta_{\pi(c), c}^{(r)}$ to the server. To estimate each expert from more than the clients of a single round, the server accumulates the returned models and takes their cumulative average,

$$s_k = \frac{1}{n_k} \sum_{r'=r_0}^r \sum_{c \in G_{r'} \cap \mathcal{A}_k} \theta_{k, c}^{(r')}, \quad n_k = \sum_{r'=r_0}^r |G_{r'} \cap \mathcal{A}_k|,$$

where r_0 is the round of the last reassignment. The expert is then updated as a moving average of its previous value and s_k ,

$$\theta_k^{(r)} = (1 - w) \theta_k^{(r-1)} + w s_k.$$

4.2 Phase 2: Training the Reward Model

The remaining $R/2$ rounds train a single reward model ϕ over the frozen groups $\{\mathcal{A}_k^*\}_{k=1}^K$, starting from a randomly initialized $\phi^{(R/2)}$. Each round draws clients with group-debiased sampling and each selected client trains $\phi^{(r)}$ on its local data, returning $\phi_c^{(r+1)}$. Write $G_{r, k}^*$ for the clients of group k that participate in round r .

Aggregation is *hierarchical*: size-weighted within a group, uniform across groups:

$$\bar{\phi}_k^{(r+1)} = \sum_{c \in G_{r, k}^*} \frac{N_c}{\sum_{c' \in G_{r, k}^*} N_{c'}} \phi_c^{(r+1)}, \quad \phi^{(r+1)} = \frac{1}{|\mathcal{K}_r|} \sum_{k \in \mathcal{K}_r} \bar{\phi}_k^{(r+1)},$$

where $\mathcal{K}_r$ indexes the groups reached in round r . Each group contributes one model regardless of its size, so the debiasing applied at sampling is preserved at aggregation.

The resulting $\phi^{(R)}$ is the initial global model from which every client personalizes using its own local preference data.

5 EXPERIMENTS AND RESULTS

We first evaluate the overall personalization performance of **FedGD** on synthetic and real-world datasets. After we verify the proposed mechanism through the diagnostic metrics and examine the robustness of **FedGD** with respect to the number of clusters K . Additional ablation studies and optimization sensitivity analyses are provided in Appendix C.

Implementation Details. Unless explicitly mentioned, all experiments follow the configuration. We run 400 communication rounds of FL, sampling five clients per round ($|S_r| = 5$, $|G_r| = 5$ for our methods), and each selected client performs exactly 30 local training iterations per round with a batch size of 16, using AdamW [28] with $(\beta_1, \beta_2) = (0.9, 0.95)$ and a constant learning rate of 1×10^{-5} . In configurations with multiple global models and dynamic client-model assignment, the assignments are refreshed every $T = 20$ rounds, and FedGD uses a mixing coefficient of $w = 0.6$. Methods that maintain multiple server-side models use $K = 4$ unless stated otherwise. Qwen2-0.5B [50] is used as the base model, and LoRA-based parameter-efficient tuning [15, 16] is applied with rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate 0.05. Following the standard Transformer architecture [45], LoRA is injected into the attention projection layers (q_proj, k_proj, v_proj, o_proj)

Table 3: PFL accuracy comparison over clients from a single seed. acc^t is the accuracy after t local fine-tuning steps, reported as $\text{mean}_{\pm\text{std}}$; acc^0 is given as the mean only. Multi CL requires known groups and is not deployable on real annotators (“-”). Bold marks the best value in each post-adaptation column.

Algorithm	Balanced (Synthetic)			Imbalanced (Synthetic)			Real-world		
	acc^0	acc^{10}	acc^{80}	acc^0	acc^{10}	acc^{80}	acc^0	acc^{80}	acc^{240}
Local	49.20	49.95 ± 7.26	58.05 ± 7.69	50.25	51.30 ± 7.44	58.70 ± 7.07	51.24	52.12 ± 6.65	54.59 ± 7.66
CL	51.80	53.70 ± 7.15	55.40 ± 5.99	56.00	56.90 ± 10.89	62.60 ± 7.93	58.06	57.06 ± 9.08	61.00 ± 6.69
Multi CL	92.10	92.15 ± 3.88	91.60 ± 4.29	92.30	91.70 ± 3.63	91.60 ± 3.67	—	—	—
FL [29]	50.35	92.45 ± 3.76	94.70± 2.92	55.60	90.55 ± 5.20	93.80± 3.40	60.82	61.53 ± 7.70	63.00 ± 9.10
FedBiscuit [49]	68.80	66.90 ± 6.18	71.45 ± 10.35	78.45	77.85 ± 9.85	80.80 ± 8.46	59.12	58.53 ± 6.76	57.06 ± 5.77
Soft-FL	51.70	62.40 ± 7.51	71.30 ± 6.09	56.50	75.95 ± 9.81	86.20 ± 7.41	60.88	58.71 ± 7.45	57.00 ± 9.07
FedGD (ours)	49.15	94.55± 3.58	94.30 ± 3.86	54.95	92.35± 4.02	93.75 ± 3.89	60.59	61.71± 5.98	64.41± 6.79

and the MLP projection layers (gate_proj, up_proj) of all 24 Transformer blocks, yielding 24 LoRA-adapted layers over which we compute our layer-wise diagnostics. All experiments are implemented on top of the FedBiscuit [49] codebase.

Baselines. Beyond CL, FL, and Multi-CL of Section 3, we compare three deployable methods that do not assume known groups.

- **Local:** fine-tunes the frozen backbone with LoRA on each client, without a global phase.
- **FedBiscuit** [49]: maintains K expert models on the server. At each communication round, every client receives its assigned expert model, performs local training, and uploads the updated model only to that expert for expert-specific aggregation. Every T communication rounds, clients are re-assigned to the expert with the lowest validation loss.
- **Soft-FL:** replaces the hard client-to-expert assignment in FedBiscuit with validation-based soft weights over the K experts. At each communication round, every client receives a weighted fusion of the K expert models for local training, and the resulting local model contributes to every expert-specific aggregation according to the client’s validation-based weights. See Appendix B for details.

5.1 Main Results

Table 3 compares the deployable methods after 400 communication rounds. **FedGD** attains the highest post-adaptation accuracy in every setting: $\text{acc}^{10} = 94.55$ under balanced groups, 92.35 under imbalance, and $\text{acc}^{240} = 64.41$ in the real-world setting. **FedGD** does so from the weakest starting point among all federated methods—49.15 and 54.95 on the synthetic datasets, close to random guessing. The federated phase therefore learns an initialization optimized for rapid personalization rather than high pre-adaptation accuracy.

Group-specific initializations do not necessarily improve personalization and can even degrade real-world performance. Even with ground-truth preference groups, Multi-CL gains only marginally (92.10% $\rightarrow$ 92.15% at step 10) and eventually degrades (91.60% after 80 steps). Similarly, in the balanced synthetic setting, FedBiscuit achieves an accuracy of only 66.90% after 10 adaptation steps, far behind standard FedAvg (92.45%). On the real-world dataset, both FedBiscuit (59.12% $\rightarrow$ 57.06%) and Soft-FL (60.88% $\rightarrow$ 57.00%) experience performance degradation during personalization, ultimately

Table 4: Real-world results across different models. acc^{240} is reported as $\text{mean}_{\pm\text{std}}$ over clients, marked with Bold for the best one, and acc^0 as the mean before personalization.

Method	Qwen-1.5B		Gemma-2B	
	acc^0	acc^{240}	acc^0	acc^{240}
Local	49.59	61.12 ± 8.25	58.53	67.71 ± 7.69
CL	63.71	64.53 ± 7.44	64.65	66.41 ± 7.14
FL	67.88	68.53 ± 7.86	71.29	71.47 ± 6.98
FedGD (ours)	68.65	70.29± 8.35	70.29	72.35± 7.44

finishing below single-model CL (61.00%). These results demonstrate that learning separate initializations is ineffective when preference groups are unknown or imperfectly defined.

Longer federated training substantially reduces the effect of group imbalance. Under the 200-round budget of Section 3, FL reaches only $\text{acc}^{10} = 61.75$ under the imbalanced setting (Table 2), whereas extending training to 400 rounds improves it to 90.55, indicating that group imbalance slows rather than prevents federated optimization. Nevertheless, **FedGD** still achieves the highest accuracy after only $R/2 = 200$ rounds of federated training. On the synthetic datasets, FL eventually catches up after sufficient personalization (93.75 vs. 93.80 under imbalance and 94.30 vs. 94.70 under balanced groups), indicating that the benefit of the debiased initialization lies in faster adaptation rather than a better final optimum.

FedGD outperforms the baselines across larger models (Table 4). On both Qwen 1.5B [50] and Gemma-2B [44], **FedGD** achieves the best post-adaptation accuracy, and the gain comes from personalization rather than from a stronger starting point. On Gemma-2B, **FedGD** improves from 70.29 to 72.35 while starting below FL, which gains only 0.18 points (71.29 $\rightarrow$ 71.47). On Qwen-1.5B, **FedGD** improves by 1.64 points against FL’s 0.65.

5.2 Robustness to the Choice of K

FedGD does not require identifying the true preference groups. Section 3 showed that the degradation under imbalance originates from

Table 5: Effect of the number of clusters K on FedGD under the imbalanced setting. $GQ(l)$ is the mean $\pm$ std over the 24 LoRA layers. m^0 denotes the mean reward margin at θ_0 . P^{10} is the fraction of samples correctly classified after 10 local steps. Accuracy is reported with the initial accuracy in parentheses.

Method	Flatness ($\downarrow$)			Init. margin (θ_0)		Retention ($\%, \uparrow$)	Correction ($\%, \uparrow$)	Accuracy ($\%, \uparrow$)
	$GQ(l)$	GQ_B	GQ_H	$m_{\mathcal{R}}^0$	$m_{\mathcal{W}}^0$	$P_{\mathcal{R}}^{10}$	$P_{\mathcal{W}}^{10}$	acc ¹⁰ (acc ⁰)
FedGD ($K = 2$)	0.18 ± 0.02	0.19	0.16	+3.76	-2.94	94.70	85.42	91.00 (60.90)
FedGD ($K = 3$)	0.29 ± 0.03	0.31	0.26	+4.23	-4.18	95.20	89.72	92.80 (55.80)
FedGD ($K = 4$)	0.27 ± 0.02	0.28	0.24	+3.59	-3.65	93.42	90.62	92.35 (54.95)

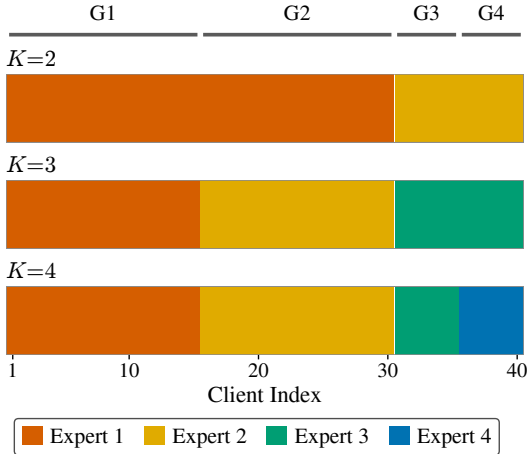


Figure 4: Phase 1 clustering results under the synthetic imbalance setting for different numbers of experts (K). Clients are colored by their assigned expert (lowest validation loss).

biased client sampling rather than an incorrect partition. Accordingly, **FedGD** only requires a partition that removes the sampling bias, rather than exactly recovering the preference groups.

Figure 4 illustrates this mechanism. With ground-truth groups G1 (Elementary & Humorous), G2 (Elementary & Non-Humorous), G3 (PhD & Humorous), and G4 (PhD & Non-Humorous), **FedGD** forms the partitions $\{G1, G2\}$ and $\{G3, G4\}$ for $K = 2$, preserving the shared education attribute while balancing the humor attribute within each partition. Increasing K to three further separates G1 and G2, resulting in the partitions $\{G1\}$, $\{G2\}$, and $\{G3, G4\}$. This preserves both education and humor information for the Elementary clients, while the PhD partition continues to average over the humor attribute. Only $K = 4$ produces the partitions $\{G1\}$, $\{G2\}$, $\{G3\}$, and $\{G4\}$, preserving both attributes for every group.

Despite these different partitions, Table 5 demonstrates that the results retain robust optimization. Across all three values of K , the learned models maintain low curvature ($GQ(l) \leq 0.29$), remain near the decision boundary with narrow initial margins, and achieve high correction rates just 10 local steps. As a result, the accuracy varies marginally, within a range of 91.0% to 92.8%.

These results indicate that recovering the exact preference groups is not necessary. Once the discovered partition sufficiently removes the biased client sampling, the resulting initialization retains the

flatness and under-confident margins enabling rapid local adaptation, making **FedGD** largely insensitive to the choice of K .

6 RELATED WORK

Reward modeling under preference heterogeneity. Reward models trained on preference data are central to alignment pipelines [32, 43], alongside alternatives such as DPO [34] and RRHF [54]. To accommodate diverse users, recent methods absorb the variation into the structure of one centralized model, including group-wise robust objectives [35], multi-objective reward heads [47], routing over latent subgroups [42], latent user variables [33], and post-hoc merging of objective-specific policies [19], so that a single deployed model can be steered per user. All of these approaches assume centralized access to preference data, which privacy regulation often precludes [17, 23, 36]. Federated preference alignment removes this assumption by aggregating locally computed alignment updates [12, 49, 51], but returning a single consensus policy limits its ability to serve heterogeneous clients, underscoring the need to leverage federated learning for local personalization.

Multiple global models under heterogeneity. When clients disagree in preference signals, a common method is to train multiple global models, one per group of similar clients. FedBiscuit [49] clusters clients by preference similarity and trains one reward model per cluster, the design our experiments compare against. FedLEASE [48], IFCA [14], CFL [38], and FeSEM [27] likewise partition clients by preference or domain similarity, while soft membership [37] and mixture-of-experts approaches [52] offer more flexible combinations. These designs rest on the premise that a client is better served by a model that never averaged over conflicting objectives. A complementary line of personalized federated learning instead keeps one global model. It defers personalization to local adaptation, either by training the global model as an initialization for a few local gradient steps [9, 11, 30] or by sharing a common representation while keeping heads or lightweight adapters client-specific [5, 24, 40, 53], and biased client participation in this setting can be corrected through clustered or importance-based sampling [4, 13]. However, prior work evaluates models primarily at the global level, so a deeper analysis is needed to examine performance trends after local fine-tuning, thereby clarifying the true role of client grouping.

7 CONCLUSION

We examined what constitutes a good initialization for personalized reward modeling under conflicting preferences. The prevailing response to such conflict is to partition clients and train one model

per group, on the premise that a client is better served by a model that never averaged over opposing labels. Our results do not support this premise. A single federated model performs at chance before adaptation, yet a few local steps carry it past a centralized model trained on the client’s own ground-truth group. Averaging cancels the opposing label choices while retaining the distinction clients agree on, leaving the initialization in a flat region from which successive local steps remain aligned. When the preference groups differ in size the cancellation becomes asymmetric: the shared model commits to the majority side and leaves minority clients too far from the decision boundary for a few local steps to recover. **FedGD** restores the symmetry by discovering the groups during federated training and debiasing the client sampling over them.

REFERENCES

- [1] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A General Language Assistant as a Laboratory for Alignment. *arXiv:2112.00861 [cs.CL]* <https://arxiv.org/abs/2112.00861>
- [2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Anna Chen, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv:2212.08073 [cs.CL]* <https://arxiv.org/abs/2212.08073>
- [3] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, J  r  my Schreuer, Javier Rando, Rachel Freedman, Tomek Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphael Segerie, Micah Carroll, Andi Peng, Phillip J.K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J Michaud, Jacob Pfau, Dmitrii Krashenninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *Transactions on Machine Learning Research* (2023). <https://openreview.net/forum?id=bx24KpJ4Eb> Survey Certification, Featured Certification.
- [4] Wenlin Chen, Samuel Horv  th, and Peter R  cht  rik. 2022. Optimal Client Sampling for Federated Learning. *Transactions on Machine Learning Research* (2022). <https://openreview.net/forum?id=8GvRCWKHIL>
- [5] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 2089–2099. <https://proceedings.mlr.press/v139/collins21a.html>
- [6] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. ULTRAfeedback: boosting language models with scaled AI feedback. In *Proceedings of the 41st International Conference on Machine Learning (Vienna, Austria) (ICML ’24)*. JMLR.org, Article 384, 23 pages.
- [7] Yann N Dauphin and Samuel Schoenholz. 2019. MetaInit: Initializing learning by learning to initialize. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alch  -Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/876e8108f87eb61877c6263228b67256-Paper.pdf
- [8] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* 39 (1977), 1–38. <http://web.mit.edu/6.435/www/Dempster77.pdf>
- [9] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. 2020. Personalized federated learning with moreau envelopes. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS ’20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1796, 12 pages.
- [10] Xin Dong, Sai Qian Zhang, Ang Li, and H.T. Kung. 2022. SphereFed: Hyper-spherical Federated Learning. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 165–184. https://doi.org/10.1007/978-3-031-19809-0_10
- [11] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS ’20)*. Curran Associates Inc., Red Hook, NY, USA, Article 300, 12 pages.
- [12] Flint Xiaofeng Fan, Cheston Tan, Yew-Soon Ong, Roger Wattenhofer, and Wei-Tsang Ooi. 2025. FedRLHF: A Convergence-Guaranteed Federated Framework for Privacy-Preserving and Personalized RLHF. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (Detroit, MI, USA) (AAMAS ’25)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 713–721.
- [13] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2021. Clustered Sampling: Low-Variance and Improved Representativity for Clients Selection in Federated Learning. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 3407–3416. <https://proceedings.mlr.press/v139/fraboni21a.html>
- [14] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS ’20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1643, 12 pages.
- [15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2790–2799. <https://proceedings.mlr.press/v97/houlsby19a.html>
- [16] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=nZeVKeeFY9>
- [17] Erin Illman and Paul Temple. 2019. California consumer privacy act. *The Business Lawyer* 75, 1 (2019), 1637–1646.
- [18] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3, 1 (1991), 79–87. <https://doi.org/10.1162/neco.1991.3.1.79>
- [19] Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2024. Personalized Soups: Personalized Large Language Model Alignment via Post-hoc Parameter Merging. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*. <https://openreview.net/forum?id=EMrnoPRvxe>
- [20] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aur  lien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends   in machine learning* 14, 1–2 (2021), 1–210.
- [21] Seongyeon Kim, Minchan Jeong, Sungyun Kim, Sungwoo Cho, Sumyeong Ahn, and Se-Young Yun. 2025. FedDr+: Stabilizing Dot-regression with Global Feature Distillation for Federated Learning. *Transactions on Machine Learning Research* (2025). <https://openreview.net/forum?id=a6WthNFhL2>
- [22] Seongyeon Kim, Gihun Lee, Jaehoon Oh, and Se-Young Yun. 2023. FedFN: Feature Normalization for Alleviating Data Heterogeneity Problem in Federated Learning. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*. <https://openreview.net/forum?id=4apX9Kcxe>
- [23] Andreas K  pf, Yannic Kilcher, Dimitri Von R  tte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richard Nagyfi, et al. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in neural information processing systems* 36 (2023), 47669–47681.
- [24] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6357–6368. <https://proceedings.mlr.press/v139/li21h.html>
- [25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papaliopoulos, and V. Sze (Eds.), Vol. 2. 429–450. https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396be6477d9475ba0c-Paper.pdf
- [26] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HJxNANvtdS>
- [27] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500.
- [28] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. <https://openreview.net/>

- forum?id=Bkg6RiCqY7
- [29] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
 - [30] Jaehoon Oh, SangMook Kim, and Se-Young Yun. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HuaYQfggn5u>
 - [31] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. 2021. {BOIL}: Towards Representation Change for Few-shot Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=umIdUL8rMH>
 - [32] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.
 - [33] Sriyash Poddar, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. 2024. Personalizing reinforcement learning from human feedback with variational preference learning. *Advances in Neural Information Processing Systems* 37 (2024), 52516–52544.
 - [34] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems* 36 (2023), 53728–53741.
 - [35] Shyam Sundhar Ramesh, Yifan Hu, Iason Chailamas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. 2024. Group robust preference optimization in reward-free rlhf. *Advances in Neural Information Processing Systems* 37 (2024), 37100–37137.
 - [36] Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. *Regulation (eu)* 679, 2016 (2016), 10–13.
 - [37] Yichen Ruan and Carlee Joe-Wong. 2022. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 8124–8131.
 - [38] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722. <https://doi.org/10.1109/TNNLS.2020.3015958>
 - [39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) [cs.LG] <https://arxiv.org/abs/1707.06347>
 - [40] Jonathan Scott, Hossein Zakerinia, and Christoph H Lampert. 2024. PeLL: Personalized Federated Learning by Learning to Learn. In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=MrYiWIDRQO>
 - [41] Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=B1ckMDqlg>
 - [42] Jingyan Shen, Jiarui Yao, Rui Yang, Yifan Sun, Feng Luo, Rui Pan, Tong Zhang, and Han Zhao. 2025. MiCRO: Mixture Modeling and Context-aware Routing for Personalized Preference Learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 17447–17463. <https://doi.org/10.18653/v1/2025.emnlp-main.882>
 - [43] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 253, 14 pages.
 - [44] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Stone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikula, Mateo Wirth, Michael Sharrman, Nikolai Chirayev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open Models Based on Gemini Research and Technology. [arXiv:2403.08295](https://arxiv.org/abs/2403.08295) [cs.CL] <https://arxiv.org/abs/2403.08295>
 - [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.
 - [46] Michael Völkske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL;DR: Mining Reddit to Learn Automatic Summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (Eds.). Association for Computational Linguistics, Copenhagen, Denmark, 59–63. <https://doi.org/10.18653/v1/W17-4508>
 - [47] Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. 2024. Arithmetic Control of LLMs for Diverse User Preferences: Directional Preference Alignment with Multi-Objective Rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 8642–8655. <https://doi.org/10.18653/v1/2024.acl-long.468>
 - [48] Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. 2025. Adaptive LoRA Experts Allocation and Selection for Federated Fine-Tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=es4TTVGJ9x>
 - [49] Feijie Wu, Xiaozhe Liu, Haoyu Wang, Xingchen Wang, Lu Su, and Jing Gao. 2025. Towards Federated RLHF with Aggregated Client Preference for LLMs. In *The Thirtieth International Conference on Learning Representations*. <https://openreview.net/forum?id=mqNKIEB6pd>
 - [50] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Quanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuyang Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. [arXiv:2407.10671](https://arxiv.org/abs/2407.10671) [cs.CL] <https://arxiv.org/abs/2407.10671>
 - [51] Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. 2024. OpenFedLLM: Training Large Language Models on Decentralized Private Data via Federated Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) (KDD '24). Association for Computing Machinery, New York, NY, USA, 6137–6147. <https://doi.org/10.1145/3637528.3671582>
 - [52] Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Qinghua Hu. 2026. pFedMoE: Data-Level Personalization With Mixture of Experts in Model-Heterogeneous Personalized Federated Learning. *IEEE Transactions on Knowledge and Data Engineering* 38, 3 (2026), 1905–1918. <https://doi.org/10.1109/TKDE.2026.3656194>
 - [53] Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Xiaoxiao Li. 2024. pFed-LoRA: Model-Heterogeneous Personalized Federated Learning with LoRA Tuning. [arXiv:2310.13283](https://arxiv.org/abs/2310.13283) [cs.LG] <https://arxiv.org/abs/2310.13283>
 - [54] Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. RRHF: Rank Responses to Align Language Models with Human Feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=EdIGMCHk4l>

Rethinking Personalized Reward Modeling for LLMs under Preference Heterogeneity via Group-Debiased Federated Learning

This appendix provides additional details omitted from the main paper. Appendix A describes the experimental setup, Appendix B presents the complete training procedures of **FedGD** and the compared methods, and Appendix C provides additional experimental analyses.

A EXPERIMENTAL SETUP

A.1 Computing Environment

Synthetic experiments are conducted on NVIDIA RTX 3090 and RTX A5000 GPUs using Python 3.9, PyTorch 2.1.0, and CUDA 12.1. Real-world experiments are performed on NVIDIA RTX 5090 GPUs using Python 3.12, PyTorch 2.12.0, and CUDA 13.0. Both environments use the same implementation with Transformers 4.49.0, PEFT 0.17.1, and Accelerate 1.10.1 under bfloat16 mixed precision.

A.2 Implementation Details

Unless explicitly mentioned otherwise, all experiments share the following implementation.

- **Training configuration.** We run 400 communication rounds of federated learning, sampling five clients per round ($|S_r| = 5$; $|G_r| = 5$ for group-debiased methods). Each selected client performs 30 local optimization steps with a batch size of 16 using AdamW [28] with $(\beta_1, \beta_2) = (0.9, 0.95)$ and a constant learning rate of 1×10^{-5} . Client assignments are refreshed every $T = 20$ communication rounds for methods with dynamic grouping. Unless otherwise specified, multi-global methods use $K = 4$, and **FedGD** uses a mixing coefficient of $w = 0.6$.
- **Backbone models and LoRA.** We use Qwen2-0.5B [50] for the synthetic experiments. For the real-world evaluation, we use the Hugging Face checkpoints Qwen/Qwen2.5-1.5B and google/gemma-2-2b-it. Across all backbones, LoRA fine-tuning [15, 16] is performed with rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate 0.05. LoRA adapters are inserted into `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, and `up_proj`, while all backbone parameters remain frozen.
- **Reward model implementation.** Our implementation is built upon the official FedBiscuit repository (<https://github.com/HarliWu/FedBiscuit>). We directly reuse its binary-selector reward model and training pipeline without architectural modification, interpreting the resulting two logits as the reward scores $r_\theta(x, y^+)$ and $r_\theta(x, y^-)$ throughout the paper.
- **Baseline implementations.**
 - **CL.** A single reward model is trained on the centralized dataset.
 - **FL.** Standard FedAvg [29] is used to optimize a single global reward model.
 - **Multi-CL.** Oracle multi-global training is performed using the ground-truth preference groups, where one centralized reward model is trained for each group.
 - **Local.** Each client independently fine-tunes a LoRA adapter without any global training or communication.
 - **FedBiscuit.** The server maintains multiple cluster-specific global models and periodically reassigns each client to the model with the lowest validation loss.
 - **Soft-FL.** Each client receives a validation-weighted fusion of all global models and contributes to every global model according to the same validation-based weights.

The complete training procedures of **FedGD**, FedBiscuit, and Soft-FL are provided in Appendix B.

B ALGORITHMIC DETAILS

This appendix presents the complete pseudocode for **FedGD** together with the FedBiscuit [49] and Soft-FL baselines used in our experiments. Section B.1 details the proposed method, while Sections B.2 and B.3 describe the FedBiscuit and Soft-FL implementations used for comparison.

B.1 FedGD

Algorithm 1: FedGD: Federated Learning with Group Debiasing and Final Personalization

Input: experts K ; total rounds R ; reassignment period T ; local steps τ ; sampling size $|G|$; mixing coefficient w ; client set $[C]$; dataset sizes $\{N_c\}_{c=1}^C$
Result: single reward model $\phi^{(R)}$ and personalized models $\{\theta_c^{\text{PFL}}\}_{c=1}^C$

```

1 Init:  $\theta_g^{(0)}, \{\theta_k^{(0)}\}_{k=1}^K \leftarrow$  LoRA default init;  $\pi(c) \leftarrow \text{Uniform}\{1, \dots, K\} \forall c$ ;  $\mathcal{A}_k \leftarrow \{c : \pi(c) = k\} \forall k$ ;
2 Phase 1: Discovering the Groups ( $r = 0, \dots, R/2 - 1$ );
3 for  $r = 0, 1, \dots, R/2 - 1$  do
4   if  $r > 0$  and  $r \bmod T \equiv 0$  then                                     // reassign clients and reset experts
5      $\pi(c) \leftarrow \arg \min_k \text{ValLoss}(\theta_k^{(r-1)}, D_{\text{val}}^c) \forall c$ ;
6      $\mathcal{A}_k \leftarrow \{c : \pi(c) = k\} \forall k$ ;  $W_k \leftarrow 0, n_k \leftarrow 0 \forall k$ ;
7      $\theta_k^{(r-1)} \leftarrow \theta_g^{(r-1)} \forall k$ ;                                     // reset experts to reference
8   end
9    $G_r \leftarrow \text{GroupDebiasedSample}(|G|, \{\mathcal{A}_k\})$ ;
10  foreach  $c \in G_r$  in parallel do
11     $\theta_{\pi(c),c}^{(r)} \leftarrow \text{LocalTrain}(\theta_{\pi(c)}^{(r-1)}; D_{\text{train}}^c)$ ;
12  end
13  // expert update: cumulative average + moving average
14  for  $k = 1, \dots, K$  do
15     $G_{r,k} \leftarrow G_r \cap \mathcal{A}_k$ ; if  $G_{r,k} = \emptyset$  then continue;
16     $\bar{\theta}_k^{(r)} \leftarrow \frac{1}{|G_{r,k}|} \sum_{c \in G_{r,k}} \theta_{k,c}^{(r)}$ ;
17     $W_k \leftarrow W_k + |G_{r,k}| \cdot \bar{\theta}_k^{(r)}$ ;  $n_k \leftarrow n_k + |G_{r,k}|$ ;  $s_k \leftarrow W_k/n_k$ ;
18     $\theta_k^{(r)} \leftarrow (1 - w) \theta_k^{(r-1)} + w s_k$ ;
19  end
20   $\theta_g^{(r)} \leftarrow \frac{1}{|G_r|} \sum_{c \in G_r} \theta_{\pi(c),c}^{(r)}$ ;                                     // reference: uniform average
21 end
22 // freeze the discovered groups at the end of Phase 1
23  $\pi(c) \leftarrow \arg \min_k \text{ValLoss}(\theta_k^{(R/2-1)}, D_{\text{val}}^c) \forall c$ ;
24  $\mathcal{A}_k^* \leftarrow \{c : \pi(c) = k\} \forall k$ ; discard empty clusters and reindex  $k = 1, \dots, K$ ;
25 Phase 2: Training the Reward Model ( $r = R/2, \dots, R - 1$ );
26 Initialize  $\phi^{(R/2)} \leftarrow$  LoRA default init;
27 for  $r = R/2, R/2 + 1, \dots, R - 1$  do
28    $G_r \leftarrow \text{GroupDebiasedSample}(|G|, \{\mathcal{A}_k^*\})$ ; broadcast  $\phi^{(r)}$  to  $G_r$ ;
29   foreach  $c \in G_r$  in parallel do
30      $\phi_c^{(r+1)} \leftarrow \text{LocalTrain}(\phi^{(r)}; D_{\text{train}}^c)$ ;
31   end
32   // hierarchical: size-weighted within a group, uniform across groups
33    $G_{r,k}^* \leftarrow G_r \cap \mathcal{A}_k^* \forall k$ ;  $\mathcal{K}_r \leftarrow \{k : G_{r,k}^* \neq \emptyset\}$ ;
34   for  $k \in \mathcal{K}_r$  do
35      $\bar{\phi}_k^{(r+1)} \leftarrow \sum_{c \in G_{r,k}^*} \frac{N_c}{\sum_{c' \in G_{r,k}^*} N_{c'}} \phi_c^{(r+1)}$ ;
36   end
37    $\phi^{(r+1)} \leftarrow \frac{1}{|\mathcal{K}_r|} \sum_{k \in \mathcal{K}_r} \bar{\phi}_k^{(r+1)}$ ;
38 end
39 Final Personalization;
40 Server broadcasts  $\phi^{(R)}$  to all clients;
41 foreach  $c \in [C]$  in parallel do
42    $\theta_c^{(\text{PFL},0)} \leftarrow \phi^{(R)}$ ; fine-tune on  $D_{\text{train}}^c$  to obtain  $\theta_c^{\text{PFL}}$ ;
43 end
44 return  $\phi^{(R)}, \{\theta_c^{\text{PFL}}\}_{c=1}^C$ ;

```

Description. Algorithm 1 implements **FedGD**, which applies group debiasing when the true preference groups are unknown. Unlike the multi-global designs of FedBiscuit [49], the K expert models are not the object of training: they exist only to *discover* a partition of the clients, and the model that clients ultimately personalize is a single reward model ϕ trained afterwards. Like the single-global design of Section 2.1, each client receives exactly one model per round, so the per-round communication cost is matched.

Phase 1 (rounds 0 to $R/2 - 1$). The server maintains K experts $\{\theta_k\}_{k=1}^K$ together with a reference model θ_g . Clients start from a uniform random assignment, and every T rounds each client evaluates all K experts on its own validation split and joins the one with the lowest validation loss [49], giving the groups $\{\mathcal{A}_k\}_{k=1}^K$. Each reassignment also resets every expert to the current reference model, $\theta_k \leftarrow \theta_g$, so that an expert does not carry over what it learned from the clients it held before the reassignment. In the rounds between two reassignments, the server draws the participating set G_r with group-biased sampling over the current groups, sends each selected client the expert of its own group, and receives the locally trained model. Because a single round contributes only $|G_r|/K$ clients to a given expert on average, the server accumulates the returned models since the last reassignment and takes their cumulative average s_k , then updates the expert as a moving average $\theta_k^{(r)} = (1 - w)\theta_k^{(r-1)} + w s_k$; this estimates each expert from more clients than a single round provides and damps the round-to-round variance of the estimate. The reference model is updated by a uniform average over G_r : it serves as a neutral point to which the experts are reset rather than as a loss minimizer, and since every client takes the same τ local steps, weighting by dataset size here would let the discovered partition follow data volume rather than preference. At the end of Phase 1 the clients are reassigned once more, empty clusters are discarded, and the resulting groups $\{\mathcal{A}_k^*\}$ are frozen.

Phase 2 (rounds $R/2$ to $R - 1$). The remaining half of the budget trains a single reward model ϕ from a fresh LoRA initialization, with group-biased sampling over the frozen groups and hierarchical aggregation: the returned models are averaged in proportion to N_c within each group, and the resulting group models are then averaged uniformly. Debiasing the sampling equalizes how often a group is heard, and aggregating hierarchically keeps that equality at the server, where weighting every client by N_c would otherwise let a group with larger clients dominate a round it shares with a smaller one. Starting from a fresh initialization rather than from θ_g isolates ϕ from the intermediate, unstable partitions that Phase 1 passes through; we compare the two choices empirically in Appendix C.3. Note that ϕ therefore receives only $R/2$ rounds of training, half the budget given to the single-global and multi-global baselines.

Final personalization. The server broadcasts $\phi^{(R)}$ to all clients, and each client fine-tunes it on D_{train}^c without further communication to obtain θ_c^{PFL} . In the notation of Section 2.1, **FedGD** is a single-global design with $\theta_c^{\text{init}} = \phi^{(R)}$ for every client; the K experts and the reference model are discarded after Phase 1. When the discovered partition coincides with the true preference groups, this reduces exactly to the FL_target oracle of Section 3.2.

B.2 FedBiscuit [49]

Description. Algorithm 2 is the federated counterpart of Multi-CL: it trains one global model per preference group while discovering the groups from decentralized data instead of assuming them. We follow FedBiscuit [49] and reinterpret it in an EM-style manner [8] for our personalized reward modeling setting. The server maintains K cluster-specific global models but sends only one of them to each client per round, so the per-round communication cost matches the single-global design of Section 2.1. Cluster assignments are refreshed every T rounds: the *E-step* assigns each client to the model with the lowest validation loss on D_{val}^c , and the *M-step* performs T rounds of clustered federated learning, where each cluster updates only its assigned model.

Phase 1 warms up each model independently with FedAvg for T rounds so that the first E-step does not operate on K identical models. Without this warm-up, every client would observe identical validation losses for all cluster-specific models, making the initial assignment arbitrary. Phase 2 then alternates between the E-step and M-step. Since assignments based solely on validation loss may produce highly imbalanced clusters, each reassignment is followed by the original size-balancing heuristic, which iteratively moves clients from the largest cluster to the smallest until the cluster sizes differ by at most one while minimizing the increase in validation loss.

Aggregation follows the original FedBiscuit rule [49],

$$\theta_k^{(r)} = \left(1 - \sum_{c \in S_{r,k}} p_c\right) \theta_k^{(r-1)} + \sum_{c \in S_{r,k}} p_c \theta_{k,c}^{(r)}, \quad p_c = \frac{N_c}{\sum_{j \in [C]} N_j}, \quad (1)$$

where $S_{r,k}$ denotes the sampled clients assigned to model k at round r . Since clients are sampled uniformly from $[C]$ rather than from each cluster, the total weight $\sum_{c \in S_{r,k}} p_c$ varies substantially across rounds. Directly averaging only the participating clients would therefore produce unstable updates when few clients from a cluster are sampled. The residual term preserves a $(1 - \sum_{c \in S_{r,k}} p_c)$ fraction of the previous model, stabilizing the optimization under sparse cluster participation.

In Phase 3, each client selects the model among $\{\theta_k^{(R)}\}_{k=1}^K$ with the lowest validation loss on its validation split and fine-tunes it locally, yielding $\theta_c^{\text{init}} = \theta_{\pi_c^*}^{(R)}$ in the notation of Section 2.1. Unlike **FedGD**, which discards the auxiliary experts after Phase 1 and deploys a single reward model, FedBiscuit retains all K cluster-specific models at deployment, requiring each client to evaluate all K models before personalization.

Algorithm 2: FedBiscuit: Federated Training with Size-Balanced Hard-Clustered Global Models and Final Personalization

Input: number of global models K ; total rounds R ; reassignment period T ; local steps τ ; client set $[C]$; dataset sizes $\{N_c\}_{c=1}^C$; initial parameters $\{\theta_k^{(0)}\}_{k=1}^K$

Result: clustered global models $\{\theta_k^{(R)}\}_{k=1}^K$ and personalized models $\{\theta_c^{\text{PFL}}\}_{c=1}^C$

```

1   $p_c \leftarrow N_c / \sum_{j \in [C]} N_j \quad \forall c$ ;
2  Phase 1: Warm-up (independent FedAvg for each model);
3  for  $k = 1, 2, \dots, K$  do
4      for  $t = 1, 2, \dots, T$  do
5          Sample a client subset  $S_{t,k} \subseteq [C]$ ;
6          foreach  $c \in S_{t,k}$  in parallel do
7              Client  $c$  receives  $\theta_k^{(t-1)}$ ;
8              Run  $\tau$  local update steps on  $D_{\text{train}}^c$  and send  $\theta_{k,c}^{(t)}$  to the server;
9          end
10          $\theta_k^{(t)} \leftarrow \frac{1}{|S_{t,k}|} \sum_{c \in S_{t,k}} \theta_{k,c}^{(t)}$ ; // FedAvg update for model  $k$ 
11     end
12 end
13 Phase 2: Size-balanced hard assignment;
14 for  $r = T + 1, T + 2, \dots, R$  do
15     if  $r - 1 \equiv 0 \pmod{T}$  then
16         // (1) Client-driven reassignment by validation loss
17         Server broadcasts  $\{\theta_k^{(r-1)}\}_{k=1}^K$  to all clients;
18         foreach  $c \in [C]$  do
19              $\ell_{c,k} \leftarrow \text{ValLoss}(\theta_k^{(r-1)}, D_{\text{val}}^c) \quad \forall k$ ;  $\pi(c) \leftarrow \arg \min_k \ell_{c,k}$ ;
20         end
21          $\mathcal{A}_k \leftarrow \{c : \pi(c) = k\} \quad \forall k$ ;
22         // (2) Size balancing
23         while  $\max_k |\mathcal{A}_k| - \min_k |\mathcal{A}_k| > 1$  do
24              $k^+ \leftarrow \arg \max_k |\mathcal{A}_k|$ ;  $k^- \leftarrow \arg \min_k |\mathcal{A}_k|$ ;
25              $c^* \leftarrow \arg \min_{c \in \mathcal{A}_{k^+}} (\ell_{c,k^-} - \ell_{c,k^+})$ ; // least-cost move
26              $\mathcal{A}_{k^+} \leftarrow \mathcal{A}_{k^+} \setminus \{c^*\}$ ;  $\mathcal{A}_{k^-} \leftarrow \mathcal{A}_{k^-} \cup \{c^*\}$ ;  $\pi(c^*) \leftarrow k^-$ ;
27         end
28         // (3) Uniform sampling and cluster-wise training
29         Sample a subset of participating clients  $S_r \subseteq [C]$ ;
30         for  $k = 1, 2, \dots, K$  do
31              $S_{r,k} \leftarrow S_r \cap \mathcal{A}_k$ ; //  $\{S_{r,k}\}_{k=1}^K$  partitions  $S_r$ 
32             if  $S_{r,k} \neq \emptyset$  then
33                 Server sends  $\theta_k^{(r-1)}$  to all  $c \in S_{r,k}$ ;
34                 foreach  $c \in S_{r,k}$  in parallel do
35                     Client  $c$  runs  $\tau$  local update steps on  $D_{\text{train}}^c$  and sends  $\theta_{k,c}^{(r)}$  to the server;
36                 end
37                  $\theta_k^{(r)} \leftarrow (1 - \sum_{c \in S_{r,k}} p_c) \theta_k^{(r-1)} + \sum_{c \in S_{r,k}} p_c \theta_{k,c}^{(r)}$ ; // weighted aggregation with residual
38             end
39         end
40     end
41 Phase 3: Final personalization;
42 Server broadcasts  $\{\theta_k^{(R)}\}_{k=1}^K$  to all clients;
43 foreach  $c \in [C]$  do
44      $\pi_c^* \leftarrow \arg \min_k \text{ValLoss}(\theta_k^{(R)}, D_{\text{val}}^c)$ ;
45 end
46 foreach  $c \in [C]$  in parallel do
47      $\theta_c^{(\text{PFL},0)} \leftarrow \theta_{\pi_c^*}^{(R)}$ ; fine-tune on  $D_{\text{train}}^c$  to obtain  $\theta_c^{\text{PFL}}$ ;
48 end
49 return  $\{\theta_k^{(R)}\}_{k=1}^K, \{\theta_c^{\text{PFL}}\}_{c=1}^C$ ;

```

B.3 Soft-Clustered Multi-Global Models (Soft-FL)

Algorithm 3: Soft-FL: Federated Training with Soft-Clustered Global Models and Final Personalization

Input: number of global models K ; total rounds R ; refresh period T ; local steps τ ; client set $[C]$; initial global parameters $\{\theta_k^{(0)}\}_{k=1}^K$

Result: global models $\{\theta_k^{(R)}\}_{k=1}^K$ and personalized models $\{\theta_c^{\text{PFL}}\}_{c=1}^C$

```

1  $\{w_{k,c}^{(0)}\}_{k,c} \leftarrow \text{UpdateSoftWeights}(\{\theta_k^{(0)}\}_{k=1}^K);$  // initial soft assignments
2 Phase 1: Federated multi-global training;
3 for  $r = 1, 2, \dots, R$  do
4   Sample participating clients  $S_r \subseteq [C]$ ;
5   // (1) Server  $\rightarrow$  clients: fused initialization
6   foreach  $c \in S_r$  do
7      $\theta_c^{\text{fuse},(r-1)} \leftarrow \sum_{k=1}^K w_{k,c}^{(r-1)} \theta_k^{(r-1)}$ ; send  $\theta_c^{\text{fuse},(r-1)}$  to client  $c$ ;
8   end
9   // (2) Local training at clients
10  foreach  $c \in S_r$  in parallel do
11    Run  $\tau$  local update steps on  $D_{\text{train}}^c$  from  $\theta_c^{\text{fuse},(r-1)}$  and send  $\theta_c^{+, (r)}$  to the server;
12  end
13  // (3) Expert-wise aggregation at the server
14  for  $k = 1, 2, \dots, K$  do
15     $\theta_k^{(r)} \leftarrow \frac{1}{Z_k^{(r)}} \sum_{c \in S_r} w_{k,c}^{(r-1)} \theta_c^{+, (r)}$ ,  $Z_k^{(r)} = \sum_{c \in S_r} w_{k,c}^{(r-1)}$ ;
16  end
17  // (4) Refresh soft weights every  $T$  rounds and at  $r=R$ 
18  if  $(r \bmod T = 0)$  or  $r = R$  then
19     $\{w_{k,c}^{(r)}\}_{k,c} \leftarrow \text{UpdateSoftWeights}(\{\theta_k^{(r)}\}_{k=1}^K);$ 
20  end
21 end
22 Phase 2: Final personalization;
23 foreach  $c \in [C]$  do
24    $\theta_c^{\text{init}} \leftarrow \sum_{k=1}^K w_{k,c}^{(R)} \theta_k^{(R)}$ ; send  $\theta_c^{\text{init}}$  to client  $c$ ;
25 end
26 foreach  $c \in [C]$  in parallel do
27    $\theta_c^{(\text{PFL},0)} \leftarrow \theta_c^{\text{init}}$ ; fine-tune on  $D_{\text{train}}^c$  to obtain  $\theta_c^{\text{PFL}}$ ;
28 end
29 Function  $\text{UpdateSoftWeights}(\{\theta_k\}_{k=1}^K)$ :
30   Server broadcasts  $\{\theta_k\}_{k=1}^K$  to all clients;
31   foreach  $c \in [C]$  in parallel do
32      $a_{k,c} \leftarrow \text{ValAcc}(\theta_k, D_{\text{val}}^c) \quad \forall k$ ;
33      $w_{k,c} \leftarrow a_{k,c} / \sum_{j=1}^K a_{j,c} \quad \forall k$ ; send  $\{w_{k,c}\}_{k=1}^K$  to the server;
34   end
35   return  $\{w_{k,c}\}_{k \in [K], c \in [C]}$ ;
36 return  $\{\theta_k^{(R)}\}_{k=1}^K, \{\theta_c^{\text{PFL}}\}_{c=1}^C$ ;

```

Description. Algorithm 3 replaces the hard cluster assignment of FedBiscuit (Algorithm 2) with a soft one. Each client c holds a weight vector $\{w_{k,c}\}_{k=1}^K$ over the K global models, obtained by normalizing its validation accuracies, so every model is updated by every client in proportion to how well it fits that client. This is a mixture-of-experts style soft assignment [18, 41], but the communication pattern is unchanged: a client still receives exactly one model per round, namely the *fused* model $\theta_c^{\text{fuse},(r-1)} = \sum_k w_{k,c}^{(r-1)} \theta_k^{(r-1)}$ formed at the server.

After local training the client returns a single model $\theta_c^{+, (r)}$, from which all K experts must be updated. We assign it to each expert with that client's weight, which amounts to solving

$$\theta_k^{(r)} = \arg \min_{\theta} \sum_{c \in S_r} w_{k,c}^{(r-1)} \|\theta - \theta_c^{+, (r)}\|_2^2,$$

whose closed form is the weighted average in Algorithm 3. The weights are refreshed every T rounds by broadcasting all K models for validation, the soft counterpart of the E-step in FedBiscuit. In Phase 2, client c is initialized from its own fused model $\theta_c^{\text{init}} = \sum_k w_{k,c}^{(R)} \theta_k^{(R)}$ and fine-tunes it locally without further communication.

Adapter initialization. Soft-FL requires the K models to differ at $r = 0$, which the standard LoRA initialization does not provide. There, the down-projection A is random and the up-projection B is zero, so $BA = 0$ and every model is functionally identical to the frozen base model; all validation accuracies coincide, the weights $w_{k,c}$ are uniform in k , and both the fusion and the expert-wise aggregation collapse to the single-global design of Section 2.1. We cannot break this tie with a warm-up phase as FedBiscuit does, since warming up K models for T rounds each would cost TK additional rounds and break the matched communication budget.

We instead initialize the K adapters as small, mutually orthogonal perturbations of the base weights. For a LoRA module with output dimension d_{out} , input dimension d_{in} , and rank r , we draw Gaussian random matrices and take their QR decompositions to obtain orthonormal $U_{\text{big}} \in \mathbb{R}^{d_{\text{out}} \times Kr}$ and $V_{\text{big}} \in \mathbb{R}^{d_{\text{in}} \times Kr}$. Adapter k takes the k -th contiguous column block U_k, V_k and is set to

$$A_k \leftarrow V_k^T, \quad B_k \leftarrow \epsilon_{\text{layer}} (1 + \text{jitter} \cdot k) U_k,$$

so that the K updates $B_k A_k$ span mutually orthogonal subspaces. Here ϵ_{layer} scales with the Frobenius norm of the corresponding base weight (a fan-in heuristic when unavailable) and jitter adds a small per-adapter offset; we use $\epsilon_{\text{layer}} = 0.005$ and jitter = 0.01, and apply the same procedure to LoRA embedding modules when present. Each model is thus a slightly different perturbation of the base model, which makes the initial validation accuracies and hence the soft weights non-uniform while keeping all updates in a small neighborhood of the base weights.

Table 6: Effect of the number of clusters K on personalized accuracy, averaged over clients.

Method	Imbalanced (acc^{80})			Real-world (acc^{240})		
	2	3	4	2	3	4
FedBiscuit	73.50	76.95	80.80	58.94	57.52	57.06
Soft-FL	84.45	86.30	86.20	57.23	56.77	57.00
FedGD (ours)	93.50	93.65	93.75	64.47	64.77	64.41

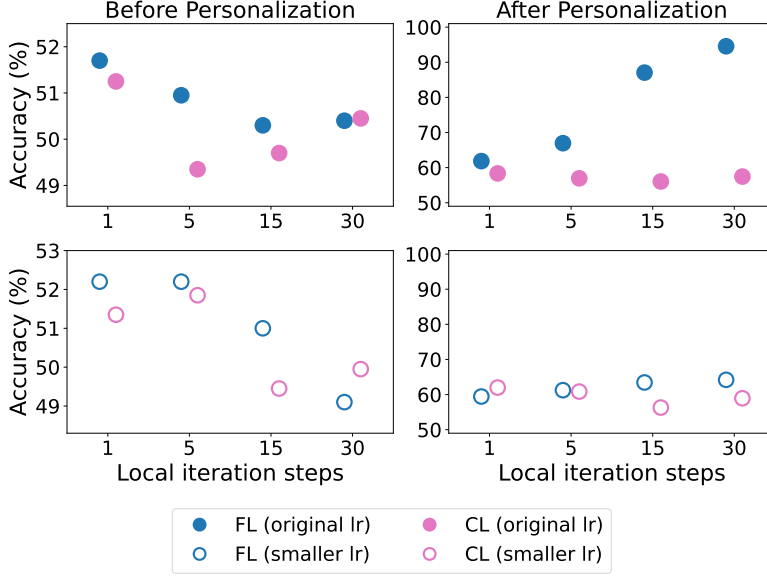


Figure 5: Sensitivity to the local optimization budget and learning rate on the balanced synthetic dataset. We compare FL and CL by varying the number of local optimization steps before aggregation ($\tau \in \{1, 5, 15, 30\}$) and using either the default learning rate or a learning rate that is $10\times$ smaller. The left and right columns report the global accuracy before and after personalization, respectively.

C ADDITIONAL EXPERIMENTAL RESULTS

This section provides additional experimental analyses that complement the main results. We first evaluate the sensitivity of **FedGD** to the number of experts. We then analyze the optimization behavior of FL under different local optimization budgets and learning rates. Next, we evaluate the effect of the Phase 2 initialization strategy.

C.1 Sensitivity to the Number of Experts

Table 6 reports the personalized performance under different numbers of experts ($K \in \{2, 3, 4\}$). Across both the imbalanced synthetic and real-world datasets, **FedGD** consistently maintains stable performance across different choices of K , indicating that its performance is not sensitive to the exact number of experts.

C.2 Optimization Sensitivity

Figure 5 further studies the effect of the local optimization budget and learning rate on FL and centralized learning (CL). The top row shows the results with the default learning rate, whereas the bottom row uses a learning rate that is $10\times$ smaller. The left and right columns report the global accuracy before personalization (acc^0) and the personalized accuracy after 10 local adaptation steps (acc^{10}), respectively. We vary the number of local optimization steps before aggregation ($\tau \in \{1, 5, 15, 30\}$).

With the default learning rate, FL and CL achieve similar global accuracies before personalization, whereas FL yields increasingly higher personalized accuracies after 10 local adaptation steps as the local optimization budget grows. In contrast, this trend is largely diminished with the smaller learning rate, suggesting that sufficient local adaptation before aggregation, rather than simply increasing the optimization budget, is essential for obtaining a better initialization for personalization.

C.3 Phase 2 Initialization

Table 7: Effect of the Phase 2 initialization strategy for the target reward model ϕ . *Continue* warm-starts ϕ from the Phase 1 global model θ_g , whereas *Random* initializes only ϕ from scratch while preserving the expert assignments obtained in Phase 1. Entries report personalized accuracies (%) averaged over clients, where acc^t denotes the accuracy after t local fine-tuning steps. Bold indicates the better result in each column.

Phase 2 init	Imbalanced			Balanced		
	acc^0	acc^{10}	acc^{80}	acc^0	acc^{10}	acc^{80}
Continue	50.20	91.00	92.45	49.00	87.95	90.10
Random (ours)	54.95	92.35	93.75	49.15	94.55	94.30

During Phase 1, the client-to-expert assignments gradually converge to the underlying preference groups, but they remain noisy for a substantial number of rounds before stabilizing. The reference model θ_g is updated throughout this period, so it may already encode optimization bias accumulated under unreliable partitions. To evaluate this effect, we compare two initialization strategies for the target reward model ϕ in Phase 2: *Continue*, which warm-starts ϕ from θ_g , and *Random*, which initializes only ϕ from scratch while preserving the expert assignments learned in Phase 1. As shown in Table 7, random initialization consistently achieves higher personalized accuracy after local adaptation.

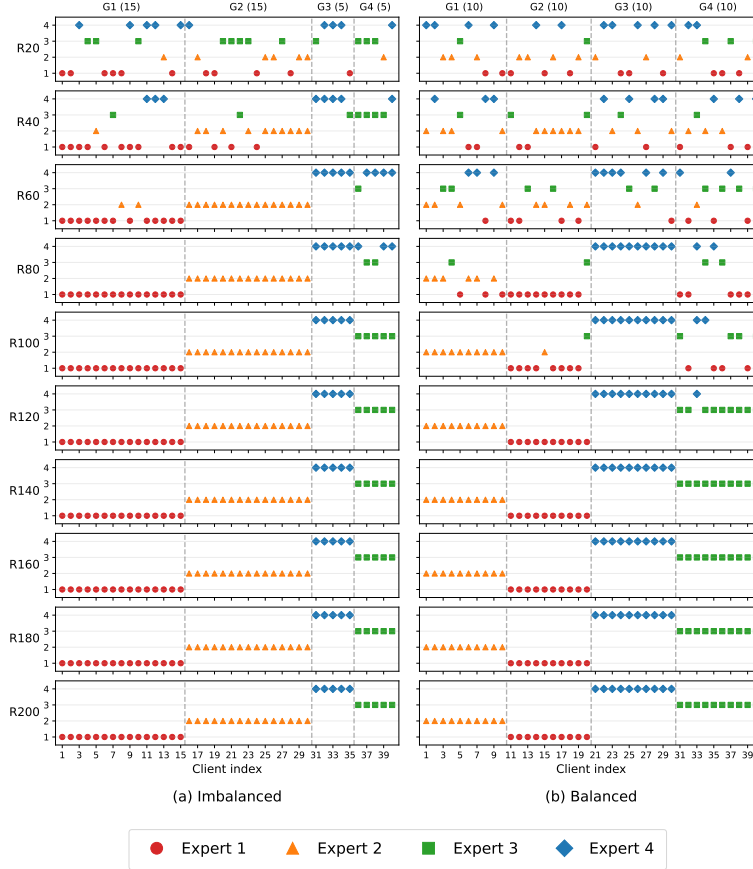


Figure 6: Evolution of client-to-expert assignments during Phase 1 on the synthetic dataset. Each marker denotes the expert assigned to a client at the corresponding communication round. The left and right columns show the imbalanced and balanced settings, respectively.

Figure 6 explains why this gap is wider under balanced groups. The assignments settle by round 100 in the imbalanced setting, whereas the balanced setting continues to reassign clients until round 140. θ_g is therefore updated under unreliable partitions for a longer portion of Phase 1 in the balanced setting, and warm-starting ϕ from it costs 6.60 points of acc^{10} (87.95 vs. 94.55) against 1.35 points under imbalance. Random initialization removes this inherited bias while retaining the converged grouping structure, and we adopt it for Phase 2 in all experiments.