PriSM: Prior-guided Shared-basis Mixture Personalization for LLMs under Sparse User Histories

Hea Eun Lee **Hyungi Lee*** **Jangho Kim***
Kookmin University
{helee,lhk2708,jangho.kim}@kookmin.ac.kr

Abstract

Personalizing large language models is essential for user-centric applications, yet remains challenging under sparse user histories, privacy constraints, and limited on-device resources. Existing approaches either rely on prompting or retrieval without adapting model parameters, train and store separate adapters for each user, or use coarse group-level adapters that cannot fully capture individual variation. We propose **PriSM** (**Pri**or-guided **S**hared-basis **M**ixture Personalization), a lightweight framework that models personalization as a posterior-inspired update over mixtures of shared LoRA bases. PriSM first learns reusable basis adapters that capture group-level adaptation patterns, then uses a cluster-conditional Dirichlet prior and a user-conditioned hypernetwork to infer posterior-like mixture weights from sparse user profiles. The resulting posterior mean synthesizes a personalized LoRA update in a single forward pass, allowing the model to rely on group-level priors when user evidence is limited and to adapt toward user-specific behavior when sufficient evidence is available. This design enables scalable, privacy-preserving, and on-the-fly personalization without per-user training or per-user adapter storage. Source code is available at https://github.com/bloomer-in-making/PriSM.

1 Introduction

Large language models (LLMs) have achieved remarkable success across a wide range of natural language tasks, yet their default one-size-fits-all behavior remains insufficient for user-centric applications. In domains such as healthcare, education, personalized customer service, and recommendation systems, model utility depends not only on general language ability, but also on alignment with each user's preferences, history, and communication style [11, 24, 20, 17, 16, 5, 39]. Personalization is therefore a practical requirement for real-world deployment. However, growing privacy constraints increasingly favor on-device personalization, where user data must remain local and adaptation must be performed under limited memory and computation. These constraints make personalization especially challenging when user histories are sparse.

Existing methods address only part of the personalization problem. As illustrated in Figure 1, current LLM personalization approaches are largely trapped between two unsatisfactory extremes. (a) Per-user adapter learning, exemplified by OPPU [32], achieves strong user specificity but requires separate training and storage for every user, making it costly at scale and unreliable under sparse histories. (b) Group-adapter methods, exemplified by CARD [31], improve sharing efficiency through cluster-level adapters but remain too coarse, since additional per-user refinement is still needed after the group adapter is fixed. Other alternatives do not resolve this trade-off either. Hypernetwork-based methods such as P2P [33] avoid per-user optimization but must directly generate millions of adapter parameters from compact user representations. Decoding-time methods such as PAD [4] and embedding-level methods such as PPLUG [19] inject personalization at the input or output layer but

*Co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

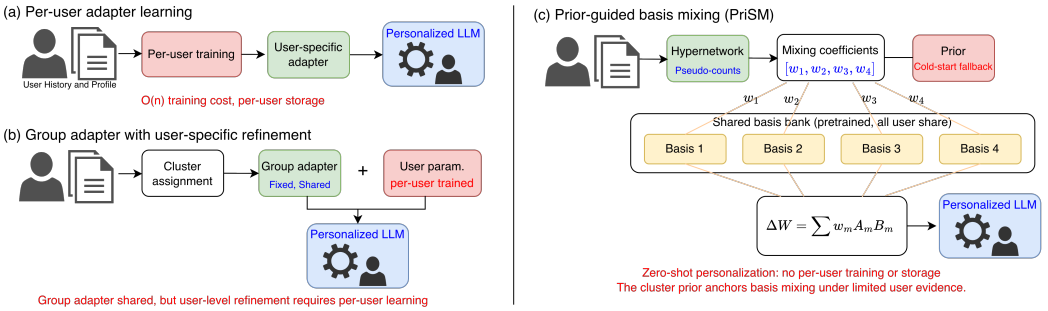


Figure 1: Three personalization paradigms for LLMs. (a) Per-user adapter learning trains and stores a separate adapter for each user, incurring $O(n)$ cost with no cross-user sharing. (b) Group-adapter methods share a cluster-level adapter, but still require per-user refinement for individual personalization. (c) **PriSM** personalizes by mixing shared LoRA bases: a lightweight hypernetwork infers mixing coefficients from the user profile, while a cluster-conditional prior stabilizes the mixture under sparse evidence. This enables personalized adaptation in a single forward pass without per-user training or storage.

leave the backbone's internal computation untouched. Retrieval-augmented approaches [15, 1] remain shallower by leaving model parameters unchanged. Overall, effective sparse-user personalization should exploit shared group-level behavioral patterns as informative priors while still capturing substantial within-group variation, without requiring per-user training or per-user adapter storage.

To address these limitations, we propose **PriSM** (**Pri**or-guided **S**hared-basis **M**ixture Personalization), a prior-guided basis-mixture framework for LLM personalization under sparse user histories. As illustrated in Figure 1(c), PriSM personalizes by composing a user-specific mixture over reusable shared Low-Rank Adaptation [LoRA; 8] bases, rather than selecting or estimating a single user adapter. Concretely, PriSM first warms up a small set of shared LoRA bases on cluster-structured data so that each basis captures a reusable group-level adaptation pattern. It then assigns a cluster-conditional Dirichlet prior over basis mixing weights and uses a lightweight hypernetwork to produce user-specific pseudo-evidence that updates the prior. The posterior mean defines the final personalized mixture over the shared bases, allowing the model to fall back to a reliable group prior when user evidence is limited and to shift toward more user-specific combinations when the profile is informative. As a result, PriSM synthesizes personalized LoRA updates in a single forward pass, without gradient-based per-user adaptation or per-user adapter storage at inference time.

By restricting user-specific adaptation to a shared basis space, PriSM avoids estimating high-capacity adapters from limited histories while retaining flexibility beyond fixed group-level adaptation. This design naturally supports efficient personalization of unseen users under sparse evidence and resource-constrained local adaptation.

In summary, this paper makes three contributions:

- We introduce **PriSM**, which reframes sparse-user personalization as posterior-inspired composition over mixtures of shared LoRA bases.
- We parameterize user-specific mixture weights through a cluster-conditional Dirichlet prior and a lightweight hypernetwork, enabling adaptive interpolation between group-level structure and user-specific evidence.
- PriSM synthesizes personalized LoRA updates in a single forward pass, enabling scalable personalization without per-user training or per-user adapter storage.

2 Preliminaries

2.1 Personalized LLM Adaptation

Let Θ_0 denote the parameters of a pretrained language model. In personalized LLM adaptation, each user u is associated with a history H_u, such as past interactions, profile information, or preference

descriptions, and optionally a small set of supervised examples $D_u = \{(x_i, y_i)\}_{i=1}^{n_u}$. The goal is to construct a user-specific predictor $p_{\Theta_0, \Delta W_u}(y \mid x)$, where ΔW_u is a user-dependent update applied to the frozen backbone. A standard approach estimates ΔW_u by optimizing a user-specific objective,

$$\Delta W_u^\star = \arg\min_{\Delta W} \sum_{(x_i, y_i) \in D_u} \ell(p_{\Theta_0, \Delta W}(y_i \mid x_i)) + \lambda \Omega(\Delta W),$$

where ℓ denotes the training objective (e.g., cross-entropy for next-token prediction) and Ω is a regularizer. While this formulation directly captures individual user behavior, it requires separate optimization and storage for each user. This becomes costly at scale and unreliable when n_u is small. In this work, we focus on the sparse-user setting, where personalization must be performed from limited user history without per-user gradient-based adaptation or per-user adapter storage.

2.2 LoRA-based Parameter-Efficient Adaptation

We consider parameter-efficient personalization using LoRA. Assume a fixed weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ in some frozen backbone, then LoRA represents an update as

$$W_u = W_0 + \Delta W_u, \qquad \Delta W_u = B_u A_u,$$

where $A_u \in \mathbb{R}^{r \times d_{\text{in}}}$, $B_u \in \mathbb{R}^{d_{\text{out}} \times r}$, and $r \ll \min(d_{\text{in}}, d_{\text{out}})$. Learning a dedicated LoRA update for each user is more efficient than full fine-tuning, but still requires user-specific training and storage. Our goal is therefore to construct ΔW_u efficiently from user information while avoiding a separately optimized adapter for every user.

2.3 Dirichlet Distribution and Conjugate Updating

PriSM uses simplex-valued mixture weights. A natural distribution over the probability simplex

$$\Delta^{M-1} = \left\{ w \in \mathbb{R}^M_{\geq 0} : \sum_{m=1}^{M} w_m = 1 \right\}$$

is the Dirichlet distribution with concentration parameter $\alpha = (\alpha_1, \ldots, \alpha_M)$ where $\forall i,\ \alpha_i > 0$:

$$\text{Dir}(w \mid \alpha) = \frac{\Gamma(\alpha_0)}{\prod_{m=1}^{M} \Gamma(\alpha_m)} \prod_{m=1}^{M} w_m^{\alpha_m - 1}, \qquad \alpha_0 = \sum_{m=1}^{M} \alpha_m,$$

where $\Gamma(t) = \int_0^\infty x^{t-1} e^{-x}\, dx$ denotes the Gamma function and $\alpha_m > 0$ can be interpreted as a prior pseudo-count for component m. The Dirichlet distribution is conjugate to the multinomial distribution. Suppose a count vector $n = (n_1, \ldots, n_M),\ \sum_{m=1}^{M} n_m = N$, is generated from a multinomial likelihood

$$p(n \mid w) = \frac{N!}{\prod_{m=1}^{M} n_m!} \prod_{m=1}^{M} w_m^{n_m}.$$

With a Dirichlet prior $p(w) = \text{Dir}(w \mid \alpha)$, the posterior is $p(w \mid n) = \text{Dir}(w \mid \alpha + n)$ and the posterior mean is

$$\mathbb{E}[w \mid n] = \frac{\alpha + n}{\mathbf{1}^\top (\alpha + n)}.$$

Thus, the posterior mean interpolates between prior knowledge α and observed evidence n: when evidence is scarce, the prior dominates, while stronger evidence shifts the estimate toward the empirical counts.

3 Prior-guided Shared-basis Mixture Personalization

3.1 Overview

In this section, we propose **Pri**or-guided **S**hared-basis **M**ixture Personalization (PriSM), which addresses sparse-user personalization by replacing per-user adapter learning with prior-guided composition over shared LoRA bases. The method has two stages. In **Stage I**, we learn a small set of

reusable basis LoRAs from cluster-structured user data and assign each cluster a fixed Dirichlet prior over basis mixing weights. In **Stage II**, we train a lightweight hypernetwork that maps a compact user profile to pseudo-evidence, which is combined with the cluster prior to form posterior-like Dirichlet parameters. The mean of the resulting distribution is used as the user-specific mixture over shared bases.

At inference, PriSM uses the user profile and assigned cluster to retrieve the cluster prior, compute the posterior-mean mixture in a single forward pass, and synthesize the personalized LoRA update from the shared bases, without per-user optimization or adapter storage. The full procedure is summarized in Appendix B.

3.2 Shared Foundations

Shared Basis LoRA Instantiation. PriSM constructs user-specific LoRA updates from a small set of shared basis adapters $\{\Delta W_m\}_{m=1}^{M}$, where each basis is parameterized as $\Delta W_m = B_m A_m$ with $A_m \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B_m \in \mathbb{R}^{d_{\text{out}} \times r}$. Given these shared bases, personalization reduces to estimating a user-specific mixing vector $w_u \in \Delta^{M-1}$, and the personalized LoRA update is

$$\Delta W_u = \sum_{m=1}^{M} w_{u,m} \Delta W_m, \qquad w_{u,m} \geq 0, \quad \sum_{m=1}^{M} w_{u,m} = 1.$$

We mix the completed basis updates ΔW_m directly, rather than mixing A_m and B_m separately, to avoid cross-basis terms such as $B_i A_j$ for $i \neq j$. In the current global-mixture setting, the same w_u is shared across all LoRA-injected modules; we compare this design with per-module mixtures in Appendix E.6.

User Profile Construction. To estimate w_u from a compact representation of user history, PriSM constructs a fixed-size representative-diverse buffer from the full history H_u and embeds each item with a frozen text encoder $\phi(\cdot)$. Let $e_{u,i} = \phi(h_{u,i}) \in \mathbb{R}^D$. We summarize the buffer by its empirical mean and element-wise variance,

$$\mu_u = \frac{1}{N_b} \sum_{i=1}^{N_b} e_{u,i}, \qquad v_u = \frac{1}{N_b} \sum_{i=1}^{N_b} (e_{u,i} - \mu_u)^2,$$

and form the raw profile $r_u = [\, \mu_u ; \log(v_u + \epsilon) \,]$. The final hypernetwork input is a normalized profile $\hat{r}_u$, obtained by separately standardizing the mean and log-variance components over the training-user population. Detailed buffer construction and normalization are referred to Appendix G.

3.3 Stage I: Prior Basis Formation

In **Stage I**, we define cluster-conditioned priors over LoRA basis mixtures and warm up each basis LoRA independently on the data assigned to its corresponding cluster. This aligns the prior preference for basis m with a basis that has been trained to capture the adaptation pattern of cluster m, while still allowing all bases to be recombined for user-specific personalization.

Cluster-Conditional Prior. Motivated by the cluster-level sharing strategy of CARD [31], PriSM uses user clustering to define a cluster-conditioned prior over LoRA basis mixing weights. Specifically, we run spherical KMeans [10, 22] on the training-user profile means $\{\mu_u\}_{u \in \mathcal{U}_{\text{train}}}$ with M clusters, matching the number of shared LoRA bases. We use μ_u for clustering to capture central preference tendency, while log-variance provides complementary within-history heterogeneity to the hypernetwork. Let $\{\kappa_m\}_{m=1}^{M}$ denote the resulting centroids. After constructing the centroids, each new user u is assigned to the nearest centroid $c(u)$ as follows:

$$c(u) = \operatorname*{argmin}_{m \in \{1,\ldots,M\}} \|\mu_u - \kappa_m\|_2^2,$$

where the user profile means and cluster centroids are L2-normalized; thus, this nearest-centroid assignment is equivalent to maximizing cosine similarity. The assigned cluster determines the Dirichlet prior over the basis mixing weights:

$$p(w_u \mid c(u)) = \text{Dir}(w_u \mid \alpha_{c(u)}),$$

where $\alpha_{c(u)} \in \mathbb{R}^M_{>0}$ is a predefined concentration vector. For each cluster $c \in \{1, \ldots, M\}$, we define the concentration vector $\alpha_c = (\alpha_{c,1}, \ldots, \alpha_{c,M})$ as

$$\alpha_{c,m} = \begin{cases} \alpha_{\text{on}}, & m = c, \\ \alpha_{\text{off}}, & \text{otherwise}, \end{cases} \qquad \alpha_{\text{on}} > \alpha_{\text{off}} > 0.$$

This weakly asymmetric prior encourages users in cluster c, but does not force them, to rely more on basis c, since all other bases retain nonzero prior mass. By conditioning the prior on cluster assignment rather than using a universal prior, PriSM incorporates both user-specific profile information and relational information induced by training-user clusters when estimating the mixture weights. For zero-history users, we use the population prior $\boldsymbol{\alpha}_{\text{pop}} = \sum_c \pi_c \boldsymbol{\alpha}_c$ as a non-personalized fallback, where π_c is the training cluster proportion. Prior and clustering sensitivity is analyzed in Appendix D.

Prior Basis Formation Objective. Since the cluster-conditional prior softly favors basis m for users in cluster m, we initialize each shared basis to encode the corresponding cluster-level adaptation pattern rather than starting from an entirely unstructured basis set. Let $\mathcal{U}_m = \{u \mid c(u) = m\}$ and $\mathcal{D}_m = \bigcup_{u \in \mathcal{U}_m} D_u$ denote the users and training data assigned to cluster m. In **Stage I**, we warm up each basis LoRA on its associated cluster data using the following objective:

$$\mathcal{L}_{\text{warmup}}(\Delta W_m; \mathcal{D}_m) = - \sum_{(x,y) \in \mathcal{D}_m} \log p_{\Theta_0, \Delta W_m}(y \mid x).$$

This stage aligns the semantics of basis m with the prior preference of cluster m, providing reusable group-level adaptation patterns that can later be recombined for user-specific personalization.

3.4 Stage II: Prior-guided Posterior Personalization

Posterior-Inspired Parameterization. In Stage I, PriSM defines a cluster-conditional Dirichlet prior over basis mixing weights. We now construct a user-dependent update rule that converts this prior into a posterior-like distribution using the user's history. Inspired by the Dirichlet–multinomial conjugate update in Section 2.3, PriSM treats the user profile as evidence that modulates the cluster prior, while using learned pseudo-evidence rather than observed multinomial counts. Specifically, given the profile $\hat{r}_u \in \mathbb{R}^{d_r}$ constructed in Section 3.2, two lightweight hypernetwork heads predict

$$\mathbf{M}_1(\hat{r}_u) = \text{softmax}(f_1(\hat{r}_u)) \in \Delta^{M-1}, \qquad \mathbf{M}_2(\hat{r}_u) = \text{softplus}(f_2(\hat{r}_u)) \in \mathbb{R}_{>0},$$

where $f_1 : \mathbb{R}^{d_r} \to \mathbb{R}^M$ and $f_2 : \mathbb{R}^{d_r} \to \mathbb{R}$ are both simple two-layer MLPs. Here, $\mathbf{M}_1(\hat{r}_u)$ determines how the user-specific evidence is allocated across the shared bases, while $\mathbf{M}_2(\hat{r}_u)$ controls the overall strength of that evidence relative to the prior. Separating these two roles allows PriSM to independently model which bases should be emphasized for a user and how strongly the user's history should influence the resulting mixture. Their product defines the profile-conditioned pseudo-evidence vector $\tilde{n}_u = \mathbf{M}_2(\hat{r}_u)\mathbf{M}_1(\hat{r}_u) \in \mathbb{R}^M_{>0}$. PriSM then updates the predefined cluster prior with this pseudo-evidence to obtain posterior-like Dirichlet parameters:

$$q_\theta(w_u \mid \hat{r}_u, c(u)) = \text{Dir}(w_u \mid \beta_u), \qquad \beta_u = \alpha_{c(u)} + \tilde{n}_u.$$

Thus, the fixed prior $\alpha_{c(u)}$ provides cluster-level information, while the hypernetwork predicts user-conditioned evidence that shifts the posterior-like distribution according to the user's profile.

Posterior-Mean Personalization Rule. Given the posterior-like Dirichlet distribution defined above, one could sample multiple mixture weights $w_u \sim q_\theta(w_u \mid \hat{r}_u, c(u))$, instantiate multiple personalized LoRA updates, and perform Bayesian Model Averaging [BMA; 36] over their predictions. However, BMA typically requires a separate forward pass for each sampled mixture and is computationally inefficient for on-the-fly personalization [38, 14]. PriSM therefore uses a simple deterministic rule based on the posterior mean:

$$\bar{w}_u = \frac{\alpha_{c(u),0}}{\alpha_{c(u),0} + \mathbf{M}_2(\hat{r}_u)} \frac{\alpha_{c(u)}}{\alpha_{c(u),0}} + \frac{\mathbf{M}_2(\hat{r}_u)}{\alpha_{c(u),0} + \mathbf{M}_2(\hat{r}_u)} \mathbf{M}_1(\hat{r}_u), \quad \alpha_{c(u),0} = \sum_{i=1}^{M} \alpha_{c(u),i},$$

showing that the final mixture adaptively interpolates between the cluster prior mean and user-specific profile evidence. When $\mathbf{M}_2(\hat{r}_u)$ is small, the mixture remains closer to the prior mean; as its value

increases, greater weight is assigned to the user-conditioned allocation $\mathbf{M}_1(\hat{r}_u)$. Then the resulting personalized LoRA update can be computed as:

$$\Delta W_u = \sum_{m=1}^{M} \bar{w}_{u,m} \Delta W_m.$$

In order to analyze the effectiveness of posterior-mean inference, we additionally evaluated inference quality using posterior sampling and predictive averaging; detailed results and uncertainty analyses are provided in Appendix E.5.

Training Objective. In **Stage II**, PriSM jointly trains the hypernetwork heads f_1, f_2 and further refines the **Stage I**-warmed basis LoRAs. The main objective is the standard next-token reconstruction loss under the posterior-mean personalized update:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{t} \log p_{\Theta_0, \Delta W_u}(y_t \mid x, y_{<t}),$$

which trains the hypernetwork to produce useful user-conditioned pseudo-evidence while allowing the basis LoRAs to adapt to mixtures encountered during personalization.

To prevent the shared bases from collapsing to redundant functions during joint training, we additionally use a functional diversity loss. Let $\mathcal{M}$ denote the set of LoRA-injected layers, and let $y_{\ell,m}$ be the module output induced by basis m at module ℓ. We normalize $\bar{y}_{\ell,m} = \mathrm{vec}(y_{\ell,m})/(\|\mathrm{vec}(y_{\ell,m})\|_2 + \epsilon)$ and penalize squared cosine similarity across basis outputs:

$$\mathcal{L}_{\mathrm{div}} = \frac{1}{|\mathcal{M}|} \sum_{\ell \in \mathcal{M}} \frac{1}{\binom{M}{2}} \sum_{i<j} \left(\bar{y}_{\ell,i}^{\top} \bar{y}_{\ell,j}\right)^2.$$

Thus, the full **Stage II** objective is $\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{div}} \mathcal{L}_{\mathrm{div}}$ where $\lambda_{\mathrm{div}} \geq 0$ is the hyperparameter.

4 Experiments

Benchmarks and metrics We evaluate PriSM on personalized classification and generation tasks from the LaMP [29] and LongLaMP [13] benchmarks. For classification, we use LaMP-1, LaMP-2, and LaMP-3 and report accuracy/F1 or MAE/RMSE depending on the task. For generation, we evaluate short-form LaMP tasks and long-form LongLaMP tasks, reporting ROUGE-1 and ROUGE-L [18]. Following CARD [31], we use a user-based split where training and evaluation users are disjoint, thereby testing personalization for unseen users. Refer to Appendix C.1 and C.2 to see the details of metrics and datasets, respectively. We report main results across all selected LaMP and LongLaMP benchmarks. Additional analyses use task subsets matched to their purpose: classification tasks for sparse-history evaluation, LaMP-4/5/7 for comparison with P2P on its user-disjoint Random and OOD splits, and representative generation tasks for mechanism analyses.

Baselines and implementation details We evaluate PriSM against representative baselines for personalized text classification and generation, covering different personalization paradigms: the *base model*, which uses only the input query without user history or personalization; retrieval- and prompt-augmented methods including **RAG** [28], **PAG** [26], **PAD** [4], and **PPLUG** [19]; a per-user PEFT baseline, **OPPU** [32]; and the hierarchical cluster-level adaptation framework **CARD** [31]. For classification tasks, we report **RAG** and **CARD** as representative baselines to compare retrieval-based personalization with cluster-level adaptation under short-label prediction settings. All methods use Qwen3-8B [37] as the frozen backbone LLM. PriSM retrieves the top-4 user-history examples using BM25 and employs a lightweight two-head hypernetwork to predict $M_1(\hat{r}_u)$ and $M_2(\hat{r}_u)$, which determine the personalized mixture over shared LoRA bases. Further implementation details and robustness evaluations, including cross-backbone and run-to-run analyses, are provided in Appendices C and F.

4.1 Classification Task

Table 1 evaluates PriSM on LaMP classification tasks using the same BGE-small encoder and $K = 4$ cluster-level LoRA configuration for both PriSM and CARD, without task-specific tuning. This controlled setting uses the same number of cluster-level LoRA components for both methods, providing a

Table 1: Performance and computational cost on LaMP-1/2/3 classification tasks.

Method	LaMP-1		LaMP-2		LaMP-3		Cost (LaMP-2)		
	Acc ↑	F1 ↑	Acc ↑	F1 ↑	MAE ↓	RMSE ↓	Train	Eval	Eval Memory
RAG	.564	.562	.392	.348	.339	.692	–	–	–
CARD	.511	.509	.367	.290	.312	.642	28 m	193 m	83.7 GB
PriSM (ours)	**.735**	**.734**	**.574**	**.527**	**.233**	**.541**	74 m	**3 m**	**17.7 GB**

Table 2: ROUGE performance comparison on LaMP and LongLaMP personalization benchmarks. We report ROUGE-1 (R-1) and ROUGE-L (R-L) on LaMP-4/5/7 and LongLaMP-1/2/3. PriSM uses the same BM25 top-4 user-history prompt construction as the RAG and PAG baselines.

Method	LaMP-4		LaMP-5		LaMP-7		LL-1		LL-2		LL-3	
	R-1	R-L	R-1	R-L	R-1	R-L	R-1	R-L	R-1	R-L	R-1	R-L
Base Model	.146	.128	.425	.342	.497	.439	.331	.184	.247	.119	.292	.130
RAG ($k = 4$)	.166	.148	.456	.372	.500	.431	.372	.203	.244	.118	.382	.152
PAG ($k = 4$)	.164	.146	.415	.352	.507	.435	.381	.201	.255	.125	.322	.141
PAD	.158	.139	.442	.360	.502	.437	.355	.194	.248	.121	.308	.136
PPLUG	.157	.138	.464	.386	.511	.433	.391	.217	.243	.122	.396	.149
OPPU	.152	.128	.426	.342	.498	.422	.382	.202	.245	.112	.295	.132
CARD	**.218**	**.195**	.459	.387	**.521**	.448	.411	**.316**	.252	.127	**.405**	.156
PriSM (ours)	.210	.192	**.516**	**.461**	.516	**.461**	**.457**	.306	**.269**	**.148**	.342	**.181**

capacity-matched comparison while keeping their distinct adaptation mechanisms unchanged. PriSM substantially outperforms CARD across all three tasks, showing that its weight-level personalization is effective beyond generation. CARD relies on token-level logit steering, whose effect is more naturally accumulated over generated sequences and is therefore less effective for short, fixed-label outputs. This limitation is particularly relevant to LaMP-1, where personalization must ultimately affect a binary label decision. In contrast, PriSM performs user-conditioned LoRA mixing at the weight level, modulating internal representations before prediction and thereby supporting both classification and generation.

Computational cost. PriSM's zero-shot user adaptation provides substantial efficiency advantages over CARD on classification tasks, as summarized in Table 1. This difference reflects their distinct adaptation paradigms: PriSM performs amortized prediction, whereas CARD requires per-user optimization at evaluation time. PriSM's evaluation consists of a single forward pass per user (3 minutes for the entire LaMP-2 dev set, 17.7 GB memory), as the hypernetwork directly produces user-specific LoRA mixing weights from the user's history embedding. CARD, in contrast, requires fitting a 128-dimensional preference vector λ_u for each dev user via Bradley–Terry pairwise loss. At evaluation time, CARD's per-user optimization takes 193 minutes (64× slower than PriSM) even with aggressive batch-size scaling that pushes memory usage to 83.7 GB. For both methods, training time in Table 1 measures only the personalization stage and assumes that the cluster-level LoRA modules are already trained, since this stage is shared across both pipelines.

4.2 Generation Task

Table 2 compares PriSM with personalization baselines across LaMP and LongLaMP generation tasks. PriSM achieves the best performance on a majority of the reported ROUGE metrics, with its largest gain on LaMP-5, where it improves over CARD from 0.459 to 0.516 R-1 and from 0.387 to 0.461 R-L. The gains also extend to long-form personalization: PriSM obtains the best R-1 on LongLaMP-1, both metrics on LongLaMP-2, and the best R-L on LongLaMP-3, while remaining competitive on tasks where CARD performs best. These results show that user-profile-conditioned LoRA mixing is effective across diverse generation settings without per-user training or adapter storage.

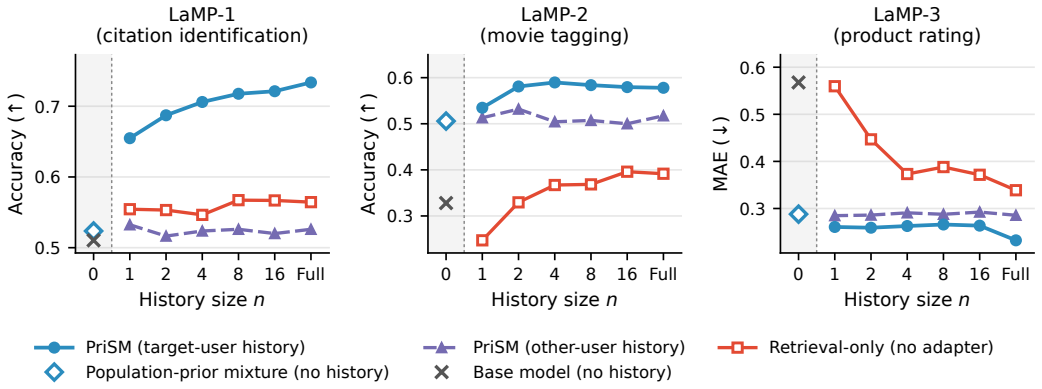


Figure 2: Performance under history-limited settings. For $n \geq 1$, PriSM uses either the target user's available history or an equally sized history subset from another user, while the retrieval-only baseline uses the target user's history without personalization adapters. PriSM consistently performs better with the target user's own history than with other-user history, while the latter still substantially outperforms retrieval-only inference, indicating contributions from both shared-basis adaptation and user-specific evidence. At $n = 0$, two non-personalized reference conditions are shown: a mixture of shared LoRA bases weighted by the population-prior mean and the base model without history.

Table 3: Comparison with P2P on user-disjoint Random and OOD splits. The 95% confidence intervals for R-1 differences are computed using 5,000 paired bootstrap resamples over the shared test instances.

		P2P		PriSM		Δ R-1 [95% CI]
Task	Split	R-1	R-L	R-1	R-L	
L4	Rand.	.186	.166	.190	.171	+.003 [-.005,.014]
	OOD	.225	.201	**.253**	**.232**	**+.028 [.001,.067]**
L5	Rand.	.534	.465	.526	.464	-.008 [-.034,.018]
	OOD	.511	.449	.513	.448	+.001 [-.022,.025]
L7	Rand.	.485	.422	**.520**	**.461**	**+.035 [.013,.057]**
	OOD	.495	.426	**.642**	**.595**	**+.147 [.114,.182]**

4.3 Generalization to Unseen Users

Sparse user history. A key motivation of PriSM is to personalize unseen users from limited history. All evaluation users are unseen during training, and we further examine this setting by limiting the available history to $n \in 1, 2, 4, 8, 16$ items. We compare PriSM using the target user's own history with two controls: retrieval-only inference without personalization adapters and the same PriSM model conditioned on an equally sized history subset from another user. As shown in Figure 2, PriSM consistently outperforms retrieval-only inference across all evaluated history sizes, including the single-item setting. Replacing the target user's history with another user's history preserves part of this gain, indicating that the shared LoRA bases themselves provide substantial task-level adaptation, but consistently degrades performance relative to using the correct target-user history across all three tasks. This additional gap indicates that PriSM also exploits user-specific evidence to adjust the composition of the shared bases. Overall, these results show that PriSM remains effective under limited history while retaining measurable sensitivity to the identity of the user's history.

Out-of-distribution users. We further compare PriSM with P2P [33], a closely related method that directly predicts personalized LoRA parameters for unseen users without per-user optimization. To ensure a split-matched comparison, we retrain PriSM on P2P's training split and evaluate both methods on the corresponding user-disjoint Random and OOD test sets using the same Qwen2.5-7B backbone. Here, Random contains randomly held-out users, whereas OOD contains users selected to be distributionally distinct from the training population. As shown in Table 3, PriSM performs

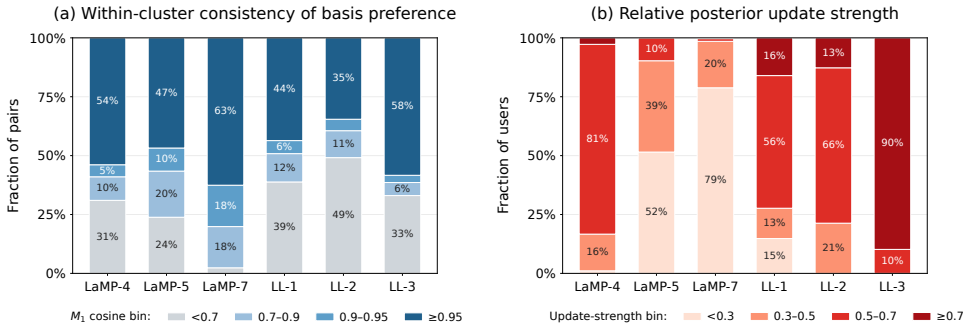
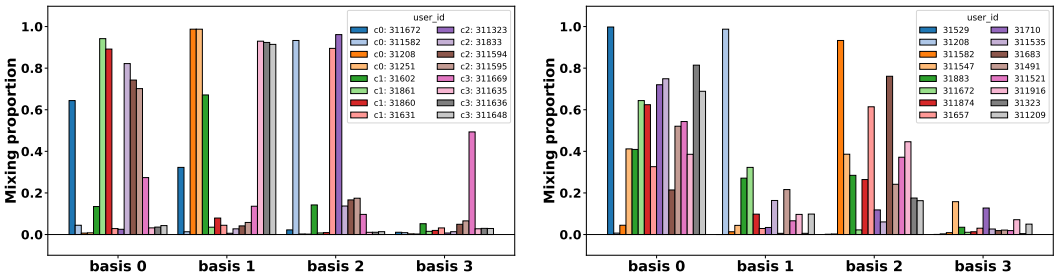


Figure 3: History-conditioned mixture inference across personalization tasks. (a) Within-cluster similarity of M_1, which determines how user-specific pseudo-evidence is allocated across the shared LoRA bases. (b) Relative evidence strength $M_2 / \sum_k \beta_{u,k}$, which measures how strongly user-specific pseudo-evidence contributes relative to the cluster-conditioned prior.



(a) Across clusters (b) Within cluster

Figure 4: User-specific basis allocation $M_1(\hat{r}_u)$ on LaMP-4. (a) Users from different clusters tend to favor different dominant bases. (b) Users within the same cluster retain distinct mixtures, showing user-specific composition beyond a fixed cluster-level adapter.

comparably to P2P on LaMP-5, while achieving significant gains on LaMP-4 OOD and on both LaMP-7 splits. The largest improvement occurs on LaMP-7 OOD, where many users exhibit highly repetitive histories, suggesting that PriSM's retrieval component can effectively exploit this structure. Overall, these results show that PriSM remains competitive on randomly held-out users and generalizes strongly to distribution-shifted users without per-user optimization.

4.4 Ablation and Mechanism Analysis

User-Level Personalization Analysis. PriSM combines a cluster-conditioned prior with history-conditioned pseudo-evidence as $\boldsymbol{\beta}_u = \boldsymbol{\alpha}_{c(u)} + M_2(\hat{r}_u)M_1(\hat{r}_u)$. Here, M_1 determines how the user evidence is allocated across the shared LoRA bases, whereas M_2 controls its strength relative to the cluster prior. Figure 3 shows that users within the same cluster often share similar allocation directions while retaining non-negligible variation, and that the relative contribution of user-specific evidence varies substantially across tasks. Thus, PriSM does not apply a fixed amount of user conditioning, but adapts both the allocation of evidence across shared bases and the strength with which that evidence influences the resulting mixture.

User-specific Basis Composition. To examine whether the history-conditioned allocation $M_1(\hat{r}_u)$ produces distinct mixtures across users, Figure 4 directly visualizes the predicted basis-allocation weights on LaMP-4. Users from different clusters tend to prefer different dominant bases, while users within the same cluster retain distinct mixtures. This variation corresponds to different compositions of the shared bases: the bases remain well separated after training (Appendix E.1), and the resulting user-specific LoRA updates exhibit both within- and cross-cluster variation (Appendix E.2). Thus, PriSM goes beyond fixed cluster-level adapters by composing distinct user-specific adapters from the same shared basis set.

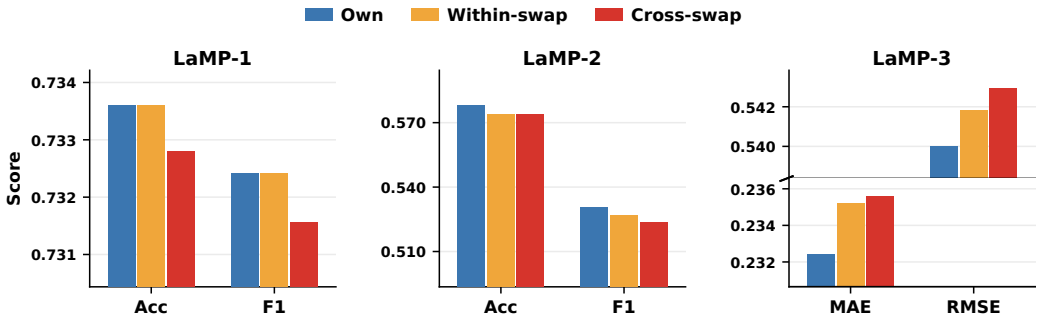


Figure 5: Functional effect of user-history conditioning on three LaMP tasks. OWN uses the target user's history statistics, while WITHIN-SWAP and CROSS-SWAP replace them with statistics from another user in the same cluster or from a globally sampled user, respectively. Higher is better for LaMP-1/2 (accuracy and macro-F1), whereas lower is better for LaMP-3 (MAE and RMSE).

History-Statistics Swap Test. Figure 5 replaces each target user's history statistics while keeping the RAG prompt fixed. Across LaMP-1/2/3, using the target user's own history yields the best or tied-best performance, while replacing it with another user's history leads to small performance degradation. Cross-cluster swaps generally produce larger degradation than within-cluster swaps. This suggests that the history-conditioned mixture captures user-specific information beyond the general capacity provided by the shared LoRA bases.

To further examine the mechanisms underlying PriSM, we analyze how Stage-I warm-up affects the learned mixture allocation (Appendix E.3) and when the cluster-conditioned prior provides additional benefit over direct history-conditioned routing (Appendix E.4).

5 Conclusion

We presented PriSM, a prior-guided shared-basis mixture framework for sparse-user LLM personalization. By combining reusable LoRA bases, a cluster-conditioned prior, and history-conditioned pseudo-evidence, PriSM synthesizes personalized LoRA updates in a single forward pass without per-user training or adapter storage. Our results show that PriSM provides effective personalization across classification and generation tasks, including unseen users under sparse history and distribution shift, while remaining sensitive to user-specific historical evidence. Importantly, PriSM allows unseen users to be personalized by composing shared adaptation patterns from their profiles, without any additional user-specific optimization, adapter fitting, or post-hoc refinement.

Limitation The current framework has limitations. Our ablations suggest that the current cluster-conditioned prior may derive much of its benefit from prior strength rather than cluster-specific basis preferences, motivating richer and more adaptive prior structures in future work. In addition, we focus on text-only personalization and do not study extension to broader domains such as multimodal personalization, where history extraction across heterogeneous modalities would need to be designed jointly with the prior.

6 Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25480825), the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Leading Generative AI Human Resources Development program (IITP-2024-RS-2024-00397085) funded by the Korea government (MSIT), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-02219317, AI Star Fellowship (Kookmin University)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-23524783), and the Global Research Support Program in the Digital Field program (IITP-2024-RS-2024-00417958) funded by the Korea government (MSIT) and supervised by the Institute of Information & Communications Technology Planning & Evaluation (IITP).

References

[1] Steven Au, Cameron Dimacali, Ojasmitha Pedirappagari, Namyong Park, Franck Dernoncourt, Yu Wang, Nikos Kanakaris, Hanieh Deilamsalehy, Ryan A Rossi, and Nesreen K Ahmed. Personalized graph-based retrieval for large language models. In *Proceedings of the 39th Pacific Asia Conference on Language, Information and Computation*, pages 678–698, 2025.

[2] Avinandan Bose, Zhihan Xiong, Yuejie Chi, Simon Shaolei Du, Lin Xiao, and Maryam Fazel. Lore: Personalizing llms via low-rank reward modeling. *arXiv preprint arXiv:2504.14439*, 2025.

[3] Eric L Buehler and Markus J Buehler. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2), 2024.

[4] Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. Pad: Personalized alignment of llms at decoding-time. *arXiv preprint arXiv:2410.04070*, 2024.

[5] Jingtong Gao, Bo Chen, Menghui Zhu, Xiangyu Zhao, Xiaopeng Li, Yuhao Wang, Yichao Wang, Huifeng Guo, and Ruiming Tang. Hierrec: Scenario-aware hierarchical modeling for multi-scenario recommendations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 653–662, 2024.

[6] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[7] Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. Hyperprompt: Prompt-based task-conditioning of transformers. In *International conference on machine learning*, pages 8678–8690. PMLR, 2022.

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[9] Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew E Peters. Hint: hypernetwork instruction tuning for efficient zero-and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11272–11288, 2023.

[10] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[11] Kevin B Johnson, Wei-Qi Wei, Dilhan Weeraratne, Mark E Frisse, Karl Misulis, Kyu Rhee, Juan Zhao, and Jane L Snowdon. Precision medicine, ai, and the future of personalized health care. *Clinical and translational science*, 14(1):86–93, 2021.

[12] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781, 2020.

[13] Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, et al. Longlamp: A benchmark for personalized long-form text generation. *arXiv preprint arXiv:2407.11016*, 2024.

[14] Hyungi Lee, Giung Nam, Edwin Fong, and Juho Lee. Enhancing transfer learning with flexible nonparametric posterior sampling. *arXiv preprint arXiv:2403.07282*, 2024.

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[16] Xiaopeng Li, Fan Yan, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. Hamur: Hyper adapter for multi-domain recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1268–1277, 2023.

[17] Xiaopeng Li, Pengyue Jia, Derong Xu, Yi Wen, Yingyi Zhang, Wenlin Zhang, Wanyu Wang, Yichao Wang, Zhaocheng Du, Xiangyang Li, et al. A survey of personalization: From rag to agent. *ACM Transactions on Information Systems*, 2025.

[18] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[19] Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. Llms+ persona-plug= personalized llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9373–9385, 2025.

[20] Zhengyi Ma, Zhicheng Dou, Yutao Zhu, Hanxun Zhong, and Ji-Rong Wen. One chatbot per person: Creating personalized chatbots based on implicit user profiles. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 555–564, 2021.

[21] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, 2021.

[22] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297, 1967.

[23] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025.

[24] Muh Putra Pratama, Rigel Sampelolo, and Hans Lura. Revolutionizing education: harnessing the power of artificial intelligence for personalized learning. *Klasikal: Journal of education, language teaching and science*, 5(2):350–357, 2023.

[25] Ruiyang Qin, Jun Xia, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Peipei Zhou, Jingtong Hu, and Yiyu Shi. Enabling on-device large language model personalization with self-supervised data selection and synthesis. In *Proceedings of the 61st ACM/IEEE design automation conference*, pages 1–6, 2024.

[26] Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081*, 2023.

[27] Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer, 1994.

[28] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 752–762, 2024.

[29] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392, 2024.

[30] Idan Shenfeld, Felix Faltings, Pulkit Agrawal, and Aldo Pacchiano. Language model personalization via reward factorization. *arXiv preprint arXiv:2503.06358*, 2025.

[31] Yutong Song, Jiang Wu, Weijia Zhang, Chengze Shen, Shaofan Yuan, Weitao Lu, Jian Wang, Amir Rahmani, Nikil Dutt, and Yu Wang. Card: Cluster-level adaptation with reward-guided decoding for personalized text generation. *arXiv preprint arXiv:2601.06352*, 2026.

[32] Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6476–6491, 2024.

[33] Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, et al. Instant personalized large language model adaptation via hypernetwork. *arXiv preprint arXiv:2510.16282*, 2025.

[34] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584, 2024.

[35] Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. Adaptive lora experts allocation and selection for federated fine-tuning. *Advances in Neural Information Processing Systems*, 38: 76018–76045, 2026.

[36] Larry Wasserman. Bayesian model selection and model averaging. *Journal of mathematical psychology*, 44(1):92–107, 2000.

[37] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[38] EungGu Yun, Hyungi Lee, Giung Nam, and Juho Lee. Traversing between modes in function space for fast ensembling. In *International Conference on Machine Learning*, pages 40555–40577. PMLR, 2023.

[39] Xiangyu Zhao, Yichao Wang, Bo Chen, Jingtong Gao, Yuhao Wang, Xiaopeng Li, Pengyue Jia, Qidong Liu, Huifeng Guo, and Ruiming Tang. Joint modeling in recommendations: A survey. *arXiv preprint arXiv:2502.21195*, 2025.

A Related Work

A.1 Hypernetwork-based Parameter Generation

Hypernetworks [6] are neural networks that generate parameters for another network, enabling conditional adaptation without explicit per-condition training. This idea has been extended to parameter-efficient transfer learning, where a hypernetwork generates lightweight modules conditioned on task descriptions or context [21, 7, 9]. Recent work applies hypernetworks to personalization, generating user-specific PEFT parameters from user profile representations [33]. PriSM adopts a different use of hypernetworks. Instead of directly generating user-specific LoRA parameters, PriSM uses a hypernetwork to predict mixing coefficients over a small set of shared basis LoRAs. This reduces the output dimensionality, eliminates the need to store separate user-specific adapters, and formulates personalization as a user-dependent combination of reusable adaptation patterns. PriSM further employs a Dirichlet-inspired prior-guided parameterization, in which a cluster-level prior anchors the mixture to group-level structure while user-profile evidence adjusts it toward individual preferences.

A.2 User History Representation

Effective personalization requires compact and informative user-history representations. Retrieval-based methods typically select query-relevant items using BM25 [27] or dense retrievers [12] and prepend them to the prompt as in-context demonstrations. While effective for query-level personalization, such retrieved items vary by query and may not capture a user's overall behavioral pattern. Recent personalization studies, therefore, condition model adaptation on user profiles or history representations [33, 16], and related work on data subset selection and on-device personalization highlights the importance of representative and diverse examples under limited storage or computation budgets [23, 25].

Motivated by these observations, PriSM constructs a fixed-size, query-independent history buffer using representative-diverse sampling. The buffer combines items close to the user's history centroid with diverse items selected by farthest-first traversal, providing a stable user representation for the hypernetwork. Details are provided in Appendix G.2.

For prompt-level retrieval, we follow the BM25 top-4 setting reported by CARD [31]. Prompt-level retrieval and buffer-level summarization therefore serve complementary roles: BM25 supplies query-relevant context to the prompt, whereas the history buffer provides a query-independent user representation for hypernetwork-based adaptation.

A.3 Factorized Personalization and Mixture-of-LoRA Methods

Factorized personalization represents user-specific preferences through combinations of shared components rather than learning an independent model for each user. PReF [30] and LoRe [2] factorize user-specific reward functions into shared components and user-specific coefficients. PriSM shares the general idea of reusable basis composition, but operates directly in the generator's parameter space: it infers a user-specific mixture from pre-existing interaction history and uses it to compose shared LoRA updates. This enables amortized personalization of unseen users without additional preference elicitation, target-user optimization, or per-user adapter storage.

Mixture-of-LoRA methods provide a complementary connection. X-LoRA [3] and HydraLoRA [34] combine multiple low-rank components through learned mixing or routing mechanisms. FedLEASE [35] further clusters heterogeneous federated clients to allocate and train shared LoRA experts, followed by adaptive expert selection. PriSM similarly composes reusable LoRA components, but differs in how the user-level mixture is inferred: it combines a cluster-conditioned prior with history-conditioned pseudo-evidence, separating its allocation across shared bases from its strength.

B Full Training and Inference Algorithm

Figure 6 provides an overview of the overall framework of our method and Algorithm 1 presents the full PriSM pipeline. In Stage I, PriSM constructs user profiles, assigns cluster labels, defines fixed cluster-conditional priors, and warms up each shared LoRA basis on its cluster-associated data. In Stage II, a lightweight hypernetwork maps each normalized user profile to user-conditioned pseudo-

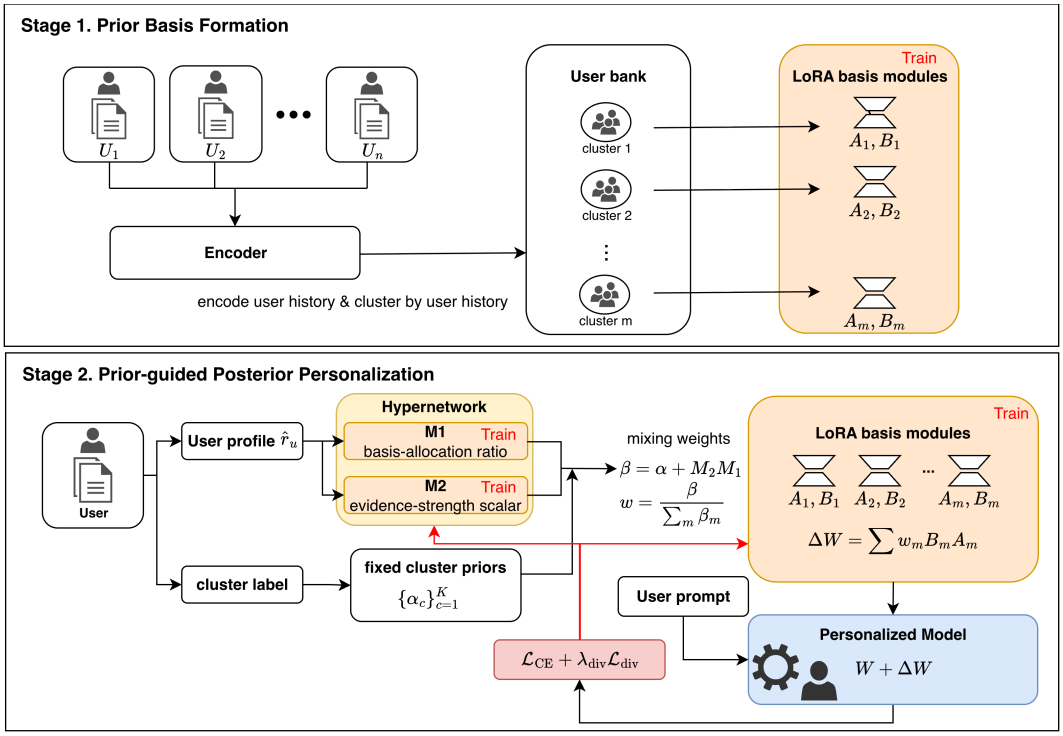


Figure 6: Overview of the proposed prior-guided shared-basis LoRA personalization framework. Stage 1 initializes LoRA basis modules from cluster-structured user data. Stage 2 maps a normalized user profile $\hat{r}_u$ to pseudo-evidence (M_1, M_2), which updates the fixed cluster prior and yields posterior-mean mixing weights over the shared bases. The personalized LoRA update is obtained by linearly combining the basis modules, enabling user-specific adaptation without per-user training or per-user adapter storage.

evidence, which updates the fixed cluster prior into posterior concentration parameters. The posterior mean defines the user-specific mixing weights over the shared bases, and the resulting personalized LoRA update is used to optimize the training objective. At inference time, the same posterior-mean rule synthesizes a personalized LoRA update in a single forward pass, without per-user training or per-user adapter storage.

In summary, PriSM is trained in two stages. **Stage I** warms up each shared basis LoRA on its associated cluster data, aligning the basis semantics with the cluster-conditioned prior. **Stage II** jointly trains the hypernetwork heads f_1, f_2 and refines the shared bases with the reconstruction and diversity losses. At inference time, no user-specific optimization or adapter storage is required. For a new user, PriSM constructs $\hat{r}_u$, assigns the user to a cluster $c(u)$, computes the posterior-mean mixture $\bar{w}_u$, and synthesizes the personalized update

$$\Delta W_u = \sum_{m=1}^{M} \bar{w}_{u,m} \Delta W_m.$$

Predictions are then generated from $p_{\Theta_0, \Delta W_u}(y \mid x)$.

C Experimental Details

All experiments are conducted on NVIDIA GeForce RTX 4090 and NVIDIA RTX PRO 6000 Black Edition GPUs. Experiments were conducted using Python 3.10 and PyTorch 2.10.0 with CUDA 12.8 support.

Base Model We use Qwen3-8B as the frozen backbone LLM for all experiments, with thinking mode disabled (`enable_thinking=False`). Chat templates are applied for prompt construction.

Algorithm 1 PriSM Training and Inference

Input: Training users $\mathcal{U}_{\text{train}}$, histories $\{H_u\}$, examples $\{D_u\}$, frozen backbone Θ_0, predefined prior template $(\alpha_{\text{on}}, \alpha_{\text{off}})$
Output: Basis LoRAs $\{\Delta W_m\}_{m=1}^{M}$, hypernetwork θ

Encode user histories, construct user profiles, and assign cluster labels $c(u)$
Define fixed cluster priors $\{\alpha_c\}_{c=1}^{K}$ from $(\alpha_{\text{on}}, \alpha_{\text{off}})$
Initialize basis LoRAs $\{\Delta W_m\}$ and hypernetwork θ
Training
Stage I: Prior Basis Formation
Warm up each basis ΔW_m on its cluster-associated subset
Stage II: Prior-guided Posterior Personalization
for each training step **do**
    for all sampled (u, x, y) **do**
        Compute $\hat{r}_u$, $M_1(\hat{r}_u)$, and $M_2(\hat{r}_u)$
        Set $\beta_u = \alpha_{c(u)} + M_2(\hat{r}_u)M_1(\hat{r}_u)$
        Set $w_u = \beta_u / (\mathbf{1}^\top \beta_u)$
        Form $\Delta W_u = \sum_{m=1}^{M} w_{u,m} \Delta W_m$
        Update θ and $\{\Delta W_m\}$ using $\mathcal{L}_{\text{CE}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}$
    end for
end for
Inference
Given a new user u, compute $\hat{r}_u$ and assign $c(u)$
Retrieve the predefined prior $\alpha_{c(u)}$
Compute $\beta_u = \alpha_{c(u)} + M_2(\hat{r}_u)M_1(\hat{r}_u)$ and $w_u = \beta_u / (\mathbf{1}^\top \beta_u)$
Form $\Delta W_u = \sum_{m=1}^{M} w_{u,m} \Delta W_m$ and decode with $p_{\Theta_0, \Delta W_u}(y \mid x)$
return $\hat{y}$

Implementation Details PriSM is trained in two stages. In Stage I, each basis LoRA is warmed up for one epoch on its cluster-associated subset using CE loss with BM25 top-4 prompts. In Stage II, the hypernetwork and basis LoRAs are jointly trained. Key hyperparameters are summarized in Table 4.

C.1 Evaluation Metrics

We evaluate classification and generation tasks using standard task-specific metrics. For classification tasks, we report accuracy and F1 score for categorical prediction tasks, and MAE/RMSE for rating or regression-style prediction tasks. For generation tasks, we report ROUGE-1 and ROUGE-L following the LaMP and LongLaMP evaluation protocols.

Accuracy. For a dataset of N examples, let y_i denote the ground-truth label and $\hat{y}_i$ denote the predicted label for the i-th example. Accuracy is defined as the fraction of correctly predicted examples:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{\hat{y}_i = y_i\}, \tag{1}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function.

F1 score. For a class c, let TP_c, FP_c, and FN_c denote the number of true positives, false positives, and false negatives, respectively. Precision and recall for class c are defined as

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \qquad \text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}. \tag{2}$$

The F1 score is the harmonic mean of precision and recall:

$$\text{F1}_c = \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \tag{3}$$

Table 4: Implementation details and hyperparameters used in PriSM.

Hyperparameter	**Value**
Backbone Model	
Base Model	Qwen/Qwen3-8B
Thinking Mode	Disabled (enable_thinking=False)
Stage I: Basis LoRA Warm-up	
Target Modules	gate_proj, up_proj, down_proj
LoRA Rank r / α	16 / 32
Per-basis Dropout	0.05
Learning Rate	5×10^{-5} (basis only)
Epochs	1
Prompt	BM25 top-4
Stage II: Joint Training	
Hypernet / Heads LR	5×10^{-4}
Basis LoRA LR	5×10^{-5} ($0.1\times$ hypernet)
Optimizer	AdamW (wd=0.01)
Effective Batch Size	8 (LaMP) / 4 (LongLaMP)
Functional Diversity λ_{div}	0.1
Mixing Method	Per-basis product
Base LM Precision	bf16
Basis LoRA Precision	fp32 master weights, bf16 forward
Hypernetwork Precision	fp32
Max Seq Length	4096 (Label masking active)
Epochs	1
Dirichlet Prior	
α_{on} / α_{off}	2.0 / 1.0
Number of Clusters K	4
Number of Bases M	4
Hypernetwork	
Trunk Architecture	2-layer FFN (independent $\times 2$)
Hidden Dimension	512
Output	$M+1$ scalars per user
Input Normalization	Dual standardization
M_2 Bias Init	0.0
User Profile	
Text Encoder	BAAI/bge-small-en-v1.5 (384d)
Profile Representation	$[\mu; \log v] \in \mathbb{R}^{768}$
Buffer Size N_b	32
Buffer Sampling	Representative-diverse (fixed)
Prompt Construction	
Retrieval	BM25 top-4
Max Tokens per Item	1000
Chat Template	Applied
Inference	
Mixing Weight	Posterior mean (no sampling)
Decoding Strategy	Greedy (do_sample=False)
Cluster Assignment	Nearest centroid (cosine)

For multi-class tasks, we report the benchmark F1 score computed by the official evaluation protocol. When macro-averaging is used, the score is

$$\text{Macro-F1} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{F1}_c, \tag{4}$$

where $\mathcal{C}$ is the set of classes.

Mean Absolute Error. For rating or regression-style tasks, let $s_i \in \mathbb{R}$ denote the ground-truth score and $\hat{s}_i \in \mathbb{R}$ denote the predicted score. Mean Absolute Error (MAE) measures the average

absolute prediction error:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{s}_i - s_i|. \tag{5}$$

Root Mean Squared Error. Root Mean Squared Error (RMSE) measures the square root of the average squared prediction error:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{s}_i - s_i)^2}. \tag{6}$$

Compared with MAE, RMSE assigns a larger penalty to large prediction errors.

ROUGE-1. For generation tasks, let $\hat{y}$ be the generated text and y be the reference text. ROUGE-1 measures unigram overlap between $\hat{y}$ and y. Let $\mathcal{G}_1(x)$ denote the multiset of unigrams in text x, and let $\mathrm{count}_x(g)$ denote the count of unigram g in x. The unigram overlap is

$$\mathrm{Overlap}_1(\hat{y}, y) = \sum_{g \in \mathcal{G}_1(\hat{y}) \cup \mathcal{G}_1(y)} \min\left(\mathrm{count}_{\hat{y}}(g), \mathrm{count}_y(g)\right). \tag{7}$$

The corresponding precision and recall are

$$P_1 = \frac{\mathrm{Overlap}_1(\hat{y}, y)}{|\mathcal{G}_1(\hat{y})|}, \qquad R_1 = \frac{\mathrm{Overlap}_1(\hat{y}, y)}{|\mathcal{G}_1(y)|}. \tag{8}$$

We report the ROUGE-1 F1 score:

$$\text{ROUGE-1} = \frac{2P_1R_1}{P_1 + R_1}. \tag{9}$$

ROUGE-L. ROUGE-L measures sequence-level overlap based on the longest common subsequence (LCS) between the generated text and the reference text. Let $\mathrm{LCS}(\hat{y}, y)$ denote the length of the longest common subsequence between $\hat{y}$ and y, after tokenization. The LCS-based precision and recall are

$$P_{\mathrm{L}} = \frac{\mathrm{LCS}(\hat{y}, y)}{|\hat{y}|}, \qquad R_{\mathrm{L}} = \frac{\mathrm{LCS}(\hat{y}, y)}{|y|}, \tag{10}$$

where $|\hat{y}|$ and $|y|$ denote the numbers of tokens in the generated and reference texts, respectively. We report the ROUGE-L F1 score:

$$\text{ROUGE-L} = \frac{2P_{\mathrm{L}}R_{\mathrm{L}}}{P_{\mathrm{L}} + R_{\mathrm{L}}}. \tag{11}$$

In all ROUGE computations, scores are averaged over evaluation examples.

C.2 Dataset Statistics and Task Details

We evaluate our method on LaMP and LongLaMP personalization benchmarks. Each instance is associated with a user, an input query, a target output, and a set of user history items. The model is required to generate the target output while leveraging user-specific information. Table 5 summarizes the datasets used in our experiments. LaMP-4, LaMP-5, and LaMP-7 evaluate short-form personalized generation, including news headline generation, academic paper title generation, and tweet paraphrasing. LongLaMP extends this setting to longer-form generation tasks, including abstract generation, topic writing, and review writing.

We use the user-based split of the LaMP and LongLaMP benchmarks. In this setting, evaluation users in the development set are disjoint from training users, so the model is evaluated on unseen users. This setting directly tests zero-shot personalization, since no user-specific parameters can be learned for evaluation users during training.

For prompt construction, we follow the BM25 top-4 retrieval setting used by retrieval-augmented baselines [27]. The retrieved history items are used in our method when constructing the input prompt, ensuring that differences in performance are not due to different prompt contexts. Separately, the hypernetwork receives a compact user profile constructed from a representative-diverse history buffer.

Table 5: Dataset statistics and task details for the personalization benchmarks used in our experiments.

Dataset	Task	#Train samples	#Eval samples	Metric
LaMP-4	News headline generation	12,527	1,925	R-1 / R-L
LaMP-5	Paper title generation	9,682	2,500	R-1 / R-L
LaMP-7	Tweet paraphrasing	10,437	1,500	R-1 / R-L
LongLaMP-1	Abstract generation	13,693	4,562	R-1 / R-L
LongLaMP-2	Topic writing	11,442	2,453	R-1 / R-L
LongLaMP-3	Product review	14,745	1,826	R-1 / R-L

C.3 Hypernetwork Architecture Details

The hypernetwork implements the two functions f_1 and f_2 used in the posterior parameter estimation. It takes the normalized user profile $\hat{r}_u \in \mathbb{R}^{2D}$ as input and produces two outputs: the basis-allocation ratio $M_1(\hat{r}_u)$ and the evidence-strength scalar $M_2(\hat{r}_u)$. In our setting, $D = 384$ because we use BGE-small embeddings, and therefore the input dimension is $2D = 768$.

We use two independent MLP branches:

$$z_{1,u} = f_1(\hat{r}_u), \qquad z_{2,u} = f_2(\hat{r}_u).$$

Each branch is a two-layer feed-forward network with ReLU activations and hidden dimension 512:

$$f_i(\hat{r}_u) = \mathrm{ReLU}\left(W_i^{(2)}\,\mathrm{ReLU}\left(W_i^{(1)}\hat{r}_u + b_i^{(1)}\right) + b_i^{(2)}\right), \qquad i \in \{1, 2\}.$$

The first head maps $z_{1,u}$ to the basis-allocation ratio:

$$M_1(\hat{r}_u) = \mathrm{softmax}\left(W_{M_1} z_{1,u} + b_{M_1}\right) \in \Delta^{M-1}.$$

The second head maps $z_{2,u}$ to the evidence-strength scalar:

$$M_2(\hat{r}_u) = \mathrm{softplus}\left(W_{M_2} z_{2,u} + b_{M_2}\right) \in \mathbb{R}_{>0}.$$

The two MLP branches are not shared because the two outputs represent different aspects of the pseudo-evidence update. M_1 determines the direction of the update over basis LoRAs, whereas M_2 determines its magnitude. Using separate branches reduces interference between these two learning signals.

We use a global-output design: for each user, the hypernetwork produces a single pair $(M_1(\hat{r}_u), M_2(\hat{r}_u))$, which is used to compute one user-level mixing weight shared across all LoRA-injected modules. Thus, the hypernetwork outputs only $M + 1$ scalars per user. For example, when $M = 4$, it outputs four allocation values and one scale value. This substantially reduces the output dimension compared with module-specific mixing and concentrates the learning signal on user-level basis selection.

Initialization. The M_1 head weights are initialized from $\mathcal{N}(0, 0.1^2)$ with zero bias, so that the initial softmax output is approximately uniform across bases. The M_2 head bias is initialized to zero, yielding $M_2 \approx \mathrm{softplus}(0) = \log 2 \approx 0.69$ at the start of training. This keeps the initial pseudo-count contribution relatively small compared with the cluster prior, so that the initial mixing weight remains close to the cluster prior mean. At the same time, $\mathrm{softplus}'(0) = 0.5$ provides a non-saturated gradient, allowing the evidence scale to be adjusted effectively during training.

D Sensitivity to Prior and Clustering Design

D.1 Prior Preference Toward Cluster-Matched Bases

Table 6 varies α_{on} with $\alpha_{\mathrm{off}} = 1$. PriSM is robust across moderate prior biases, with $\alpha_{\mathrm{on}} = 2$ providing the best overall trade-off. A stronger bias ($\alpha_{\mathrm{on}} = 3$) degrades LaMP-4 and LaMP-7, suggesting that excessive reliance on cluster-matched bases restricts user-specific composition. We therefore use a weakly asymmetric prior that anchors group structure without dominating user evidence. Additional analyses are provided in Appendices D.2 and D.3.

Table 6: Effect of cluster-matched prior bias.

α_{on}	α_{off}	LaMP-4		LaMP-5		LaMP-7	
		R-1	R-L	R-1	R-L	R-1	R-L
1.5	1.0	.2098	.1917	.5151	.4607	**.5162**	**.4618**
2.0	1.0	**.2101**	**.1924**	.5155	.4613	.5158	.4606
3.0	1.0	.2062	.1895	**.5156**	**.4616**	.5131	.4585

Table 7: Robustness analysis with different Dirichlet prior strengths on LaMP-4, LaMP-5, and LaMP-7. We vary α_{on} and α_{off} while keeping the prior mean fixed.

α_{on}	α_{off}	$\|\boldsymbol{\alpha}\|_1$	LaMP-4		LaMP-5		LaMP-7	
			R-1	R-L	R-1	R-L	R-1	R-L
1	0.5	2.5	.2066	.1896	.5161	.4617	.5163	.4619
2	1.0	5.0	.2101	.1924	.5158	.4606	.5155	.4613
4	2.0	10.0	.2090	.1920	.5152	.4611	.5159	.4614

D.2 Robustness to Dirichlet Prior Strength

We further examine the sensitivity of PriSM to the overall strength of the Dirichlet prior. In this analysis, we vary both α_{on} and α_{off} while keeping the prior mean fixed. Thus, the preferred cluster-matched basis remains the same, but the total prior concentration $\|\boldsymbol{\alpha}\|_1$ changes. A larger concentration makes the prior more influential relative to the user-specific pseudo-counts predicted by the hypernetwork, whereas a smaller concentration allows the posterior mixing weight to be more easily shifted by the hypernetwork output.

Table 7 shows that the final performance remains largely stable across different prior strengths. This indicates that PriSM is not overly sensitive to the absolute scale of the prior concentration. One possible explanation is that the hypernetwork and basis LoRAs are jointly trained under each prior setting, allowing the model to adapt its pseudo-count scale and basis usage to the given prior strength. Therefore, as long as the prior mean provides a reasonable cluster-level inductive bias, the exact prior concentration has only a limited effect on final generation performance.

D.3 Ablation on the Number of Clusters and Bases

We analyze the effect of the number of clusters K on PriSM in Table 8. In our design, each cluster is assigned one basis LoRA, and therefore increasing K also increases the number of basis LoRAs M. Thus, this ablation does not only vary the clustering granularity, but also changes the number of trainable basis adapters, storage cost, and warm-up cost.

We compare $K = M \in \{2, 4, 8\}$ across LaMP-4, LaMP-5, and LaMP-7. PriSM shows relatively stable performance across different choices of K, indicating that the proposed shared-basis mixture framework is reasonably robust to the number of clusters and basis LoRAs.

D.4 Clustering Sensitivity

We examine sensitivity to clustering initialization and algorithm using the same BGE-small profile embeddings and $K = 4$. We run K-means and GMM with 30 random seeds each and compare each solution with the deployed clustering. ARI measures agreement in individual user assignments, whereas marginal JSD measures differences in aggregate cluster proportions.

Individual assignments can vary substantially across clustering solutions, due to both initialization and algorithm choice. For example, on LaMP-4, ARI ranges from 0.5164 to 0.9986 across K-means initializations and from 0.4604 to 0.8997 across GMM initializations. In contrast, marginal JSD remains below 0.0200 across all evaluated solutions, indicating relatively stable aggregate cluster proportions. Importantly, small marginal JSD does not imply stable assignments: different user-level

Table 8: Ablation on the Number of Clusters and Bases.

$K = M$	Warm-up epochs	LaMP-4		LaMP-5		LaMP-7	
		R-1	R-L	R-1	R-L	R-1	R-L
2	1	.2121	.1941	.5164	.4639	.5233	.4679
4	1	.2101	.1924	.5155	.4613	.5158	.4606
8	2	.1994	.1823	.5099	.4563	.5170	.4610

Table 9: Clustering sensitivity over 30 seeds. Values are mean $\pm$ std.

Task	Method	ARI	Marginal JSD
LaMP-4	K-means	.8267±.1762	.0070±.0072
	GMM	.7524±.1542	.0080±.0079
LaMP-5	K-means	.7156±.0893	.0083±.0019
	GMM	.7094±.0962	.0055±.0033
LaMP-7	K-means	.9300±.0235	.0003±.0001
	GMM	.5084±.0073	.0151±.0003

partitions can preserve similar cluster proportions, as particularly illustrated by LaMP-7 GMM (ARI $= 0.5084 \pm 0.0073$, marginal JSD $= 0.0151 \pm 0.0003$).

Because cluster identity indexes a prior component in PriSM, these results indicate that aggregate population mass across prior components is relatively stable, while the component assigned to an individual user can depend on the clustering solution. This analysis does not assess downstream performance robustness.

E Analysis of the Learned Personalization Structure

E.1 Separation of Shared LoRA Bases

PriSM uses a single user-level mixture across all LoRA-injected modules, with each shared basis providing a corresponding update at every adapted module. We examine whether these basis updates remain distinct after training rather than collapsing to redundant parameter directions. For each adapted module, we consider the effective update $\Delta W_m = B_m A_m$ for each basis and measure pairwise absolute Frobenius cosine similarity. We also report the maximum-to-minimum ratio of the module-averaged basis Frobenius norms to assess whether the bases remain comparable in magnitude.

As shown in Table 10, the mean absolute cosine similarity is only $.0013$–$.0046$, with the maximum remaining below $.03$, indicating little directional overlap between the effective basis updates. The norm ratios are also close to one (1.02–1.06), indicating that the bases remain comparable in magnitude in addition to being directionally distinct. Together, these results show that the shared bases remain distinct parameter-space directions of comparable magnitude rather than collapsing to redundant updates.

E.2 Structure of user-specific LoRA Updates

Having established that the shared bases remain distinct, we next examine how their user-specific mixtures structure the final LoRA weight updates applied to individual test users, ΔW_u. For each task, we measure the mean pairwise Frobenius distance between these user-specific updates. Distances are computed exactly from the basis Gram matrix with the LoRA scaling factor included.

Across all three tasks, cross-cluster update distances are 3.3–$5.4\times$ larger than within-cluster distances (Table 11). At the same time, the nonzero within-cluster distances show that users assigned to the same cluster do not receive identical updates. Together with the basis-separation results, these findings show that PriSM does not simply assign a single adapter to each cluster. Instead, distinct shared bases are combined within a cluster-structured adaptation space, while retaining variation among users within each cluster.

Table 10: Separation of the shared LoRA bases across adapted modules. $|\cos|$ denotes the absolute Frobenius cosine similarity between effective updates $\Delta W = BA$. The norm ratio is the maximum-to-minimum ratio of the module-averaged basis Frobenius norms.

Task	Mean $\lvert\cos\rvert$	Max $\lvert\cos\rvert$	Norm ratio
LaMP-4	.0018	.0112	1.06
LaMP-5	.0013	.0063	1.02
LaMP-7	.0046	.0285	1.04

Table 11: Structure of user-specific LoRA updates across generation tasks. Within and Cross denote the mean pairwise Frobenius distance between users in the same and different clusters, respectively.

Task	Within	Cross	Cross/Within
LaMP-4	8.87	48.22	5.4×
LaMP-5	8.03	26.85	3.3×
LaMP-7	6.99	22.80	3.3×

E.3 Effect of Stage I Warm-up on Mixture Allocation

We examine how Stage-I warm-up affects the learned user-specific mixtures by measuring their distance from the corresponding cluster-prior mean. For each evaluation user u, we compute the total variation (TV) distance between the posterior-mean mixture $\bar{\mathbf{w}}_u$ and the cluster-prior mean $\boldsymbol{\pi}_{c(u)}$, where $\pi_{c(u),m} = \alpha_{c(u),m} / \sum_{m'} \alpha_{c(u),m'}$:

$$\mathrm{TV}\big(\bar{\mathbf{w}}_u, \boldsymbol{\pi}_{c(u)}\big) = \frac{1}{2} \sum_{m=1}^{M} \big|\bar{w}_{u,m} - \pi_{c(u),m}\big| .$$

TV ranges from 0 to 1, with 0 indicating that the inferred mixture matches the cluster-prior mean. We report the mean over evaluation users, with 95% confidence intervals obtained from 5,000 user-level bootstrap resamples.

For comparison, the variant without Stage-I warm-up starts from independently initialized shared bases and trains them through Stage II only. Both conditions use the same cluster assignments, fixed cluster priors, history buffers, and hypernetwork architecture, so the reference $\boldsymbol{\pi}_{c(u)}$ is identical. As shown in Figure 7, Stage-I warm-up increases the distance from the cluster-prior mean across all three generation tasks, with the largest difference on LaMP-4. These results show that Stage I affects the learned mixture allocation, with the magnitude of the effect varying across tasks.

E.4 Role of the Cluster-Conditioned Prior

We examine the role of the cluster-conditioned prior through controlled ablations on LaMP-2. Table 12 compares the full model with variants that remove Stage-I warm-up, replace the cluster-conditioned prior with a uniform prior, or directly predict mixture weights from user history. All three variants yield lower performance than the full model.

We further compare PriSM with direct history-conditioned routing under varying amounts of user history. As shown in Figure 8, PriSM outperforms direct routing on LaMP-2 at $n = 2$, 4, and 8, while the two approaches perform similarly with full histories. These results show that the cluster-conditioned design can provide an additional benefit over direct history-to-mixture routing on LaMP-2, particularly under limited user histories.

E.5 Posterior Mean versus Sampling

We further examine whether posterior sampling provides additional benefit over the posterior-mean inference used in PriSM. For each user, we draw S mixture weights from the posterior-like Dirichlet distribution and average the resulting predictive probabilities. A single sampled mixture is shared across all adapted modules. To isolate posterior sampling, all variants use identical fixed user buffers; the resulting posterior-mean values may therefore differ slightly from the main results, where buffer construction is stochastic.

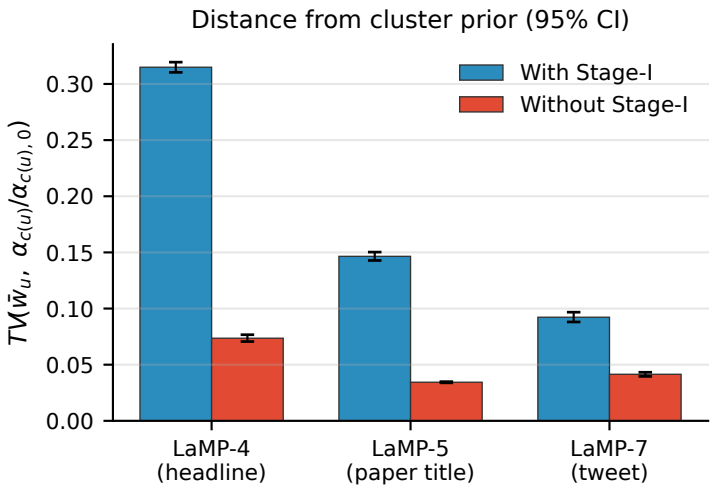


Figure 7: Effect of Stage I warm-up on the distance between inferred posterior-mean mixtures and their cluster-prior means. We report the TV distance between the inferred mixture $\bar{\mathbf{w}}_u$ and its corresponding cluster-prior mean. Error bars denote 95% user-bootstrap confidence intervals.

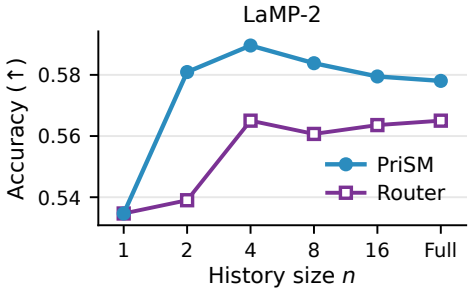


Figure 8: Comparison of PriSM and direct history-conditioned routing on LaMP-2 under varying amounts of user history.

Table 12: Ablation of the prior-guided design on LaMP-2.

Method	Acc.↑	F1↑
No Stage-I	.5318	.4940
Uniform prior	.5621	.5243
Direct Router	.5650	.5257
PriSM	**.5737**	**.5271**

As shown in Table 13, posterior sampling provides no statistically supported improvement over posterior-mean inference. Although BMA slightly improves the point estimate on LaMP-5 (up to $+0.0023$ R-1 at $S = 4$), the difference is not statistically significant (95% CI $[-0.0006, 0.0053]$), and the confidence intervals for all BMA comparisons include zero. Importantly, the lack of improvement does not simply reflect near-identical posterior samples: the mean pairwise total variation (TV) between sampled mixtures, where TV is zero for identical mixture weights and increases as the samples differ, ranges from 0.203 to 0.372 at $S = 8$. Thus, explicitly averaging multiple posterior samples provides little additional benefit over the efficient deterministic posterior-mean rule.

Posterior Mean vs. Bayesian Model Averaging To understand why posterior-mean inference performs comparably to BMA, consider the posterior-like Dirichlet distribution used by PriSM. For user u, the distribution over the basis mixture weights is

$$q_\theta(\mathbf{w}_u \mid \hat{r}_u, c(u)) = \mathrm{Dir}(\mathbf{w}_u \mid \boldsymbol{\beta}_u),$$

whose mean is

$$\bar{\mathbf{w}}_u = \mathbb{E}_{q_\theta}[\mathbf{w}_u] = \frac{\boldsymbol{\beta}_u}{\beta_{u,0}}, \qquad \beta_{u,0} = \sum_m \beta_{u,m}.$$

Since the personalized adapter is linear in the mixture weights,

$$\mathbb{E}_{q_\theta}\left[\sum_{m=1}^{M} w_{u,m}\Delta W_m\right] = \sum_{m=1}^{M} \bar{w}_{u,m}\Delta W_m = \Delta W_u. \tag{12}$$

Table 13: Posterior-mean inference and posterior sampling across tasks (LaMP-1/2: accuracy; LaMP-3: MAE; LaMP-4/5/7: ROUGE-1). BMA point estimates are not significantly different from the posterior mean under paired bootstrap tests (95% CIs include zero for all settings). TV denotes the mean pairwise total variation among $S = 8$ sampled mixture weights. All variants use the same fixed user buffers to isolate posterior sampling.

	LaMP-1 Acc.↑	LaMP-2 Acc.↑	LaMP-3 MAE↓	LaMP-4 R-1↑	LaMP-5 R-1↑	LaMP-7 R-1↑
Posterior mean	.7336	.5780	.2324	.2098	.5162	.5152
BMA ($S = 1$)	.7272	.5578	.2324	.2093	.5141	.5120
BMA ($S = 2$)	.7288	.5679	.2320	.2099	.5176	.5129
BMA ($S = 4$)	.7332	.5751	.2340	.2094	.5185	.5139
BMA ($S = 8$)	.7336	.5665	.2352	.2088	.5172	.5145
Mixture Pairwise TV ($S = 8$)	.3722	.3622	.3106	.2030	.3059	.3329

Table 14: Controlled comparison of per-module versus shared LoRA mixing. Δ = Per-module − Shared; 95% CIs are obtained by user-paired bootstrap.

Task	Δ R-1 [95% CI]	Δ R-L [95% CI]
LaMP-4	+.0008 [-.0026,+.0043]	+.0011 [-.0021,+.0042]
LaMP-5	+.0011 [-.0024,+.0045]	+.0014 [-.0020,+.0050]
LaMP-7	+.0006 [-.0024,+.0036]	-.0004 [-.0033,+.0025]

Thus, the posterior-mean rule used by PriSM exactly recovers the expected personalized adapter in parameter space.

This equivalence does not hold exactly for model predictions because the network is nonlinear in the adapter. However, for $g_{\mathbf{x},y}(\mathbf{w}) = p(y \mid \mathbf{x}, \Delta W(\mathbf{w}))$, a second-order Taylor expansion around $\bar{\mathbf{w}}_u$ gives

$$p_{\mathrm{BMA}} - p_{\mathrm{mean}} \approx \frac{1}{2}\mathrm{Tr}\left(\nabla^2_{\mathbf{w}} g_{\mathbf{x},y}(\bar{\mathbf{w}}_u)\,\mathrm{Cov}_{q_\theta}(\mathbf{w}_u)\right), \tag{13}$$

since the first-order term vanishes because $\mathbb{E}_{q_\theta}[\mathbf{w}_u - \bar{\mathbf{w}}_u] = 0$. For a Dirichlet distribution,

$$\mathrm{Cov}_{q_\theta}(\mathbf{w}_u) = \frac{\mathrm{Diag}(\bar{\mathbf{w}}_u) - \bar{\mathbf{w}}_u \bar{\mathbf{w}}_u^\top}{\beta_{u,0} + 1}. \tag{14}$$

Thus, the leading discrepancy between BMA and posterior-mean prediction is a second-order effect determined by the predictive curvature and posterior covariance. For the Dirichlet distribution, this covariance scales as $\mathcal{O}((\beta_{u,0} + 1)^{-1})$. Together with the empirical results above, this supports our use of efficient single-pass posterior-mean inference instead of Monte Carlo averaging.

E.6 Shared versus Per-Module Mixture Parameterization

PriSM uses a single user-level mixture shared across all LoRA-adapted modules. This choice provides a compact personalization space, but is less expressive than allowing each module to use its own mixture. We therefore conduct a controlled comparison with a per-module variant, where the hypernetwork predicts separate basis-allocation and evidence-strength outputs for each of the 108 adapted modules. The two variants use the same Stage-I bases, training configuration, fixed user buffers, and random seed, differing only in whether the mixture is shared across modules or predicted separately for each module.

Across all three tasks, per-module mixing yields only marginal differences from shared mixing, with all six 95% confidence intervals including zero (Table 14). Thus, the substantially greater flexibility of predicting separate mixtures for 108 adapted modules does not yield a statistically detectable improvement in this controlled comparison. We therefore use the more compact shared user-level mixture in PriSM.

Table 15: Cross-backbone evaluation on LaMP generation tasks. PriSM consistently improves the corresponding RAG baseline with Llama-3-8B-Instruct. Qwen3-8B results are included for reference.

Backbone	Method	LaMP-4		LaMP-5		LaMP-7	
		R-1	R-L	R-1	R-L	R-1	R-L
Llama-3-8B-Instruct	RAG	.1627	.1438	.4383	.3893	.3299	.2822
Llama-3-8B-Instruct	PriSM	**.2279**	**.2093**	**.5088**	**.4586**	**.5099**	**.4580**
Qwen3-8B	PriSM	.2101	.1924	.5155	.4613	.5158	.4606

Table 16: Three-run evaluation of CARD and PriSM on LaMP-1. Results are reported as Accuracy / F1.

Method	Run 1	Run 2	Run 3	Mean $\pm$ SD
CARD	.5440/.5436	.5333/.5305	.5107/.5092	.5293$\pm$.0170/.5278$\pm$.0174
PriSM	.7352/.7340	.7424/.7419	.7308/.7279	**.7361$\pm$.0059/.7346$\pm$.0070**

F Additional Robustness Evaluation

F.1 Cross-Backbone Evaluation

To examine whether PriSM's effectiveness is specific to the Qwen backbone used in the main experiment, we additionally evaluated PriSM with Llama-3-8B-Instruct on the LaMP generation tasks. We use the same personalization setting as in the main experiments and compare PriSM against the corresponding RAG baseline using the same Llama backbone.

As shown in Table 15, PriSM consistently improves over RAG across LaMP-4, LaMP-5, and LaMP-7. The improvements are particularly pronounced on LaMP-5 and LaMP-7, where PriSM improves R-1 from 0.4383 to 0.5088 and from 0.3299 to 0.5099, respectively. These results show that PriSM's gains over RAG transfer to a different backbone, beyond the Qwen3-8B model used in the main experiments.

F.2 Run-to-Run Variability

To assess run-to-run variability in the classification setting, we repeat PriSM three times on LaMP-1 under the same evaluation protocol. As shown in Table 16, PriSM exhibits limited variation across runs, with standard deviations of .0059 and .0070 for Accuracy and F1, respectively. This indicates that the reported LaMP-1 performance is stable across the evaluated runs.

G User-Profile Design

G.1 Dual Standardization

We use dual standardization to preserve cross-user differences in the user profile representation. The raw profile consists of the mean component μ_u and the log-variance component ℓ_u, which capture different aspects of the user history. While μ_u represents the central tendency of the user's historical items, ℓ_u captures within-user variation. Because these two components have different scales and distributions, they should be normalized separately.

A straightforward alternative is to apply per-sample normalization, such as LayerNorm, to the concatenated profile or to each component. Although this can reduce scale mismatch within a profile, it can also remove information about how each user deviates from the population-level distribution. Such cross-user deviation is important for personalization, because the hypernetwork must distinguish users based on relative differences in their history profiles.

In contrast, dual standardization uses statistics computed over the training-user population. This preserves the relative position of each user with respect to the population mean while still controlling the scale mismatch between μ_u and ℓ_u. Therefore, dual standardization provides a more suitable input normalization strategy for user-conditioned hypernetwork mixing.

G.2 Representative-Diverse Buffer Sampling

User histories can contain many items, and using all items is often inefficient. We therefore construct a fixed-size buffer $\mathcal{B}_u$ from the full user history $\mathcal{H}_u$. The goal is to summarize the user history while preserving both central tendency and within-user variation.

This design is particularly motivated by deployment scenarios where storing or processing the full user history is infeasible, such as on-device personalization with tight memory and compute budgets [25]. In such settings, only a small subset of user data can be retained, making the choice of which items to keep critical for personalization quality. A naive strategy of retaining recent or random items risks losing either the user's dominant preference or the diversity of their behavior. We therefore combine two complementary criteria, in line with prior findings that small but carefully curated subsets retain performance only when quality and diversity are jointly considered [23]: a centroid-based representative component and a farthest-first diverse component, rather than relying on either alone.

Beyond static representation, the buffer is designed to operate in a continual setting where user history grows over time. As new interactions are constantly added to a user's history, storing or re-processing the full history becomes increasingly expensive as users accumulate data. Rather than retaining all items, the buffer maintains a fixed-size summary that selectively preserves the most informative items — those that either reinforce the user's central preferences or expand coverage of their behavioral spread — while older, redundant, or low-signal items can be discarded. This makes the buffer a practical mechanism for long-term personalization under bounded storage, where a user's representation is continuously updated without unbounded growth in stored data.

Given item embeddings $\{e_{u,i}\}_{i=1}^{N}$, we first compute the normalized history centroid:

$$\bar{e}_u = \frac{\frac{1}{N}\sum_{i=1}^{N} e_{u,i}}{\left\| \frac{1}{N}\sum_{i=1}^{N} e_{u,i} \right\|_2}.$$

We then compute the cosine similarity between each item and the centroid:

$$s_i = e_{u,i}^{\top} \bar{e}_u.$$

The buffer is composed of two parts:

$$\mathcal{B}_u = \mathcal{B}_u^{\mathrm{rep}} \cup \mathcal{B}_u^{\mathrm{div}}.$$

The representative subset $\mathcal{B}_u^{\mathrm{rep}}$ is sampled from items with high centroid similarity, capturing the user's dominant preference or style direction. This reflects the quality criterion emphasized in prior data selection work [23], where samples close to the user's central distribution are treated as reliable indicators of preference. The diverse subset $\mathcal{B}_u^{\mathrm{div}}$ is selected from a filtered candidate pool to preserve the spread of the user's history, providing complementary coverage analogous to informativeness-aware selection in on-device personalization [25]. In our default setting, we use an equal split between representative and diverse items, i.e., $|\mathcal{B}_u^{\mathrm{rep}}| \approx |\mathcal{B}_u^{\mathrm{div}}|$. For diverse selection, we exclude both extreme outliers and near-centroid items using a quantile band over centroid similarity. Specifically, we use the middle similarity region between the lower and upper quantiles, q_{low} and q_{high}, as the candidate pool. In our experiments, we set $q_{\mathrm{low}} = 0.20$ and $q_{\mathrm{high}} = 0.90$. We then apply farthest-first selection within this pool to choose mutually diverse items.

The representative component captures the user's central preference direction, while the diverse component preserves within-user variation reflected in the log-variance profile. Together, they provide a compact and robust history summary without relying exclusively on either centroid-near items or potentially noisy outliers. We empirically validate this design in Table 17, comparing Rep-Div sampling against using all history items for computing $[\mu_u; \log v_u]$. Across LaMP-4, LaMP-5, and LaMP-7, Rep-Div sampling achieves performance comparable to the all-items baseline while operating on a fixed-size buffer. Notably, on LaMP-4, the 32-item Rep-Div buffer slightly outperforms full-history aggregation, showing that a compact history summary can preserve performance without processing all available history items. We additionally evaluate sensitivity to the stochastic buffer draw using the fixed LaMP-4 checkpoint and four independently seeded Rep-Div buffers. Performance varies only from .2093–.2106 in R-1 and .1915–.1929 in R-L, with the reported result (.2101/.1924) lying within these ranges, indicating low sensitivity to the particular buffer draw in this evaluation.

Table 17: Ablation on buffer construction strategies.

	LaMP-4		LaMP-5		LaMP-7	
Strategy	R-1	R-L	R-1	R-L	R-1	R-L
All items	.2059	.1886	.5176	.4648	.5151	.4630
Rep-Div (32 items)	.2101	.1924	.5155	.4613	.5158	.4606

G.3 Effect of History Encoder Size

Recent personalization methods such as CARD [31] employ large multilingual encoders, such as BGE-M3 with 568M parameters, to construct user-history representations. Although larger encoders may capture finer semantic and stylistic distinctions, the history encoder in our framework lies on the recurring path of personalization: it is used when new interactions arrive, when the Rep-Div buffer is refreshed, and when user profile statistics are updated. Thus, encoder size is not only an accuracy-related design choice but also a deployment-level cost factor.

To examine this trade-off, we compare our default encoder, BGE-small-en-v1.5 (33M parameters, 384-dimensional embeddings), with BGE-M3, the encoder used in CARD. With BGE-small, the hypernetwork receives the dual-standardized profile statistics $[\mu_u; \log v_u]$, resulting in a 768-dimensional input. With BGE-M3, we follow CARD's protocol and feed the raw 1024-dimensional encoder representation directly into the hypernetwork without additional standardization. The hypernetwork input dimension is adjusted accordingly, while its depth, hidden size, output mode, buffer sampling strategy, basis LoRAs, prompt construction, and training schedule are kept unchanged. This comparison intentionally reflects two practical design choices: a lightweight buffer-statistics representation and a larger direct-encoding representation. Results are shown in Table 18.

Table 18: Effect of history encoder size and profile representation on LongLaMP tasks.

				LongLaMP-1		LongLaMP-2		LongLaMP-3	
Encoder	**Params**	**Mem.**	**Hyp. in-dim**	R-1	R-L	R-1	R-L	R-1	R-L
BGE-small (ours)	33M	67MB	768 (statistics)	.4565	.3055	.2692	.1482	.3421	.1811
BGE-M3	568M	1.1GB	1024 (raw)	.4588	.3067	.2717	.1506	.3401	.1805

The performance difference should be interpreted together with deployment cost. In a static benchmark, user histories can be encoded once; however, in an online personalization setting, profile representations must be updated as users generate new interactions. A larger encoder therefore increases recurring computation and energy consumption, embedding-cache footprint, and cold-start onboarding latency. For example, the per-item fp16 cache size increases from 768 bytes with BGE-small embeddings to 2048 bytes with BGE-M3 embeddings. We therefore use the lightweight encoder as the default choice, while noting that the proposed framework is encoder-agnostic and can adopt a larger encoder when the available resource budget justifies the additional cost.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution of PriSM as a prior-guided shared-basis LoRA mixture framework for sparse-user personalization. These claims are supported by experiments on LaMP and LongLaMP generation benchmarks, LaMP classification tasks, sparse-history and OOD evaluations, and ablation and mechanism analyses. The stated scope is also consistent with the experiments, which focus on text-based personalization under limited user history.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a limitations paragraph in the conclusion. It notes that the current cluster-conditioned prior may derive much of its benefit from prior strength rather than cluster-specific basis preferences, motivating richer and more adaptive prior structures in future work. It also states that the study focuses on text-only personalization and does not evaluate multimodal personalization.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results such as theorems, lemmas, or proofs. The mathematical expressions are used to define the PriSM parameterization, Dirichlet-inspired posterior update, posterior-mean mixing rule, and training objective.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the information needed to reproduce the main experiments, including the backbone model, datasets, user-based split, evaluation metrics, prompt construction, LoRA configuration, Dirichlet prior setting, hypernetwork architecture, user-profile construction, training schedule, optimization details, and inference procedure. Additional implementation details are reported in the appendix. Code is publicly available through the project repository, and the benchmark datasets are publicly available from the official LaMP and LongLaMP sources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available through the project repository, with instructions for preprocessing, training, and evaluation. The paper does not introduce a new dataset; it uses the publicly available LaMP and LongLaMP benchmarks, which can be obtained from their official sources. The repository and appendix provide the necessary implementation details, hyperparameters, and data path instructions needed to reproduce the main experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental settings in the main text and appendix, including the benchmark datasets, user-based split, evaluation metrics, backbone model, prompt construction, LoRA target modules and rank, optimizer, learning rates, training schedule, Dirichlet prior parameters, hypernetwork architecture, user-profile construction, and inference procedure.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports point estimates for the main benchmark results, following prior work, while providing confidence intervals or run-to-run variability for selected analyses.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the main hardware and software environment used for the experiments, including the GPU types, Python version, PyTorch version, and CUDA version.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The research uses publicly available personalization benchmarks and does not introduce new human-subject data collection, sensitive private data, or harmful deployment scenarios.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The paper focuses on the technical development and empirical evaluation of PriSM and does not include a dedicated discussion of broader societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained model, image generator, scraped dataset, or sensitive user dataset. The release is limited to code and instructions for reproducing experiments with publicly available benchmarks and existing backbone models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources for the LaMP and LongLaMP benchmarks, the Qwen3-8B backbone model, the BGE-small encoder, and the baseline methods. The released repository provides instructions for obtaining the benchmark data from the official sources and for using the existing models according to their respective licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset introduced by this work is the PriSM code release. The repository provides instructions for reproducing the main experiments, including data preparation, preprocessing, training, and evaluation. No new dataset or third-party model weights are redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or new research with human subjects. It uses publicly available benchmark datasets and does not collect new annotations, conduct user studies, or recruit human participants.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or new research with human subjects. It uses publicly available benchmark datasets and does not recruit participants, collect new human-subject data, or conduct user studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper studies LLM personalization and explicitly describes the use of Qwen3-8B as the frozen backbone language model in the experimental setup. The role of the LLM, the personalization mechanism, and the inference procedure are described in the method and appendix. Any LLM use for writing or editing, if applicable, does not affect the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.