P-FEDSKILL: A PERSONALIZED FEDERATED SKILL LEARNING FRAMEWORK UNDER TASK HETEROGENEITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill learning has become an important way to improve agent performance. However, agent trajectories often contain sensitive information and cannot be shared across clients, hindering collaborative skill learning. To tackle this issue, federated skill learning (FSL) has emerged as a solution by distilling trajectories into compact skill representations and uploading them to a central server for sharing. However, existing FSL methods typically share skills based on skill similarity without considering their contributions to heterogeneous tasks across clients, thereby degrading agent performance. To address this challenge, we propose P-FedSkill, a personalized FSL framework that comprises three key components: local skill extraction (LSE), personalized skill selection (PSS), and task-aware skill routing (TASR). Specifically, LSE first extracts skill patches (i.e., task applicability conditions and execution strategies) and task execution outcomes from trajectories. PSS then uses these outcomes to evaluate each skill patch's performance on client tasks and incorporates its task applicability condition to construct a personalized skill set for each client. Finally, TASR adaptively selects a task-relevant subset of skill patches based on their applicability conditions to guide agent execution. We further provide a theoretical guarantee for personalized skill selection in P-FedSkill, and extensive experiments across four benchmarks also demonstrate its effectiveness.

1 INTRODUCTION

With the proliferation of intelligent agents, massive volumes of interaction trajectories are being generated on edge devices (Wang et al., 2025a; Lv et al., 2026). According to the International Data Corporation (IDC), more than 1 billion AI agents are expected to be deployed globally by 2029, performing approximately 217 billion actions daily (Villars, 2025). This rapidly growing volume of interaction data is accelerating advances in agent technologies, leading to progress across various domains such as intelligent manufacturing (Chen et al., 2025), cybersecurity (Deng et al., 2024), and embodied intelligence (Hong et al., 2024).

Recently, skill learning has become an important approach to improving agent performance by extracting reusable task solutions from interaction trajectories (Yu et al., 2026). These solutions can be further used to guide agents in completing new tasks, thereby improving execution efficiency (Wang et al., 2025b). However, trajectories from individual clients typically span only a narrow range of tasks, limiting the solutions available for agent execution (Jiang et al., 2023). A straightforward idea is to collect trajectories from multiple clients to learn a broader set of skills. However, such trajectories often contain sensitive information and cannot be directly shared across clients due to privacy concerns, thereby hindering collaborative skill learning (Zhang & Yang, 2026).

To address this issue, federated skill learning (FSL) has emerged as a solution by distilling trajectories into compact skill representations and uploading them to a central server for sharing. For example, Jiang et al. (2023) proposed FedSkill, which clusters parameterized skill vectors by similarity for cross-client skill sharing. However, due to task heterogeneity, similar skills may contribute differently to task execution across clients, as illustrated in Fig. 1. For some clients, a shared skill can provide useful execution strategies, whereas for others, the same skill may be irrelevant to their tasks or even mislead the agent into taking incorrect actions (Wang et al., 2025b). Ignoring such

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

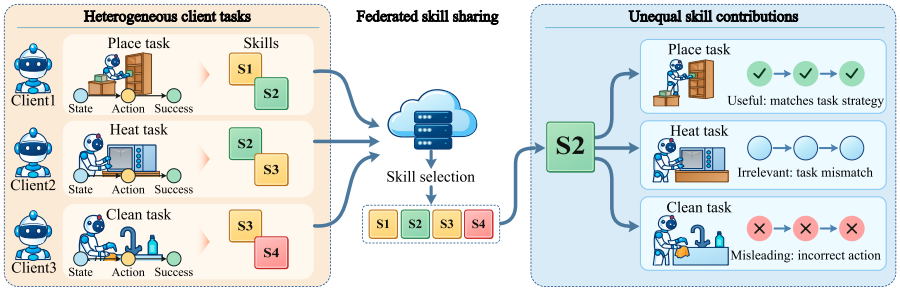


Figure 1: Illustration of unequal skill contributions under task heterogeneity.

differences in skill utility across clients may result in suboptimal skill sharing, making it difficult for clients to prioritize the skills most relevant to their tasks and ultimately degrading agent performance.

To combat the above challenge, in this paper, we propose a personalized federated skill learning framework, named P-FedSkill, to provide each client with a personalized skill set under heterogeneous task distributions. P-FedSkill consists of three key components: i) local skill extraction (LSE), ii) personalized skill selection (PSS), and iii) task-aware skill routing (TASR). Specifically, LSE groups local trajectories by task type and extracts skill patches (i.e., task applicability conditions and execution strategies) along with task execution outcomes. PSS then uses these outcomes to evaluate each skill patch's performance on client tasks and incorporates its task applicability condition to construct a personalized skill set for each client. Finally, TASR adaptively selects a task-relevant subset of skill patches based on their applicability conditions to guide agent execution. The main contributions of this paper are summarized as follows.

- We propose P-FedSkill, a personalized federated skill learning framework that enables personalized skill sharing under task heterogeneity across clients.
- We design LSE to extract skill patches from each client's local trajectories, providing diverse solutions to guide agent execution.
- We design PSS to construct a personalized skill set for each client, and devise TASR to adaptively select task-relevant skill patches from the personalized skill set to guide agent execution.
- We establish a finite-sample suboptimality bound that provides a theoretical guarantee for personalized skill selection in P-FedSkill under heterogeneous client tasks, and extensive experiments across four benchmarks further demonstrate its effectiveness.

The rest of the paper is organized as follows. Sec. 2 reviews the related work. Sec. 3 introduces the system design of P-FedSkill. Sec. 4 describes the experimental setup. Sec. 5 presents the results and analysis. Finally, Sec. 6 concludes the paper.

2 Related Work

2.1 Agent Skill Learning

Skill learning has emerged as a promising method for improving agent performance by extracting reusable skills from interaction trajectories (Fu et al., 2024; Zhou et al., 2025; Ding et al., 2026). Research efforts on agent skill learning predominantly focus on two major directions: experience abstraction and structured skill construction. For experience-based skill learning, Zhao et al. (2024) proposed ExpeL, which extracts reusable insights from successful and failed interaction trajectories and retrieves relevant experiences for unseen tasks. For structured skill construction, Klissarov et al. (2025) proposed MaestroMotif, which represents skills with initiation, execution, and termination components and uses LLM-generated code and reinforcement learning to construct and combine reusable skills. Beyond centralized skill learning, recent studies have explored federated skill learning to enable cross-client skill sharing without exposing private interaction trajectories. Jiang et al. (2023) proposed FedSkill to enable cross-client skill sharing by clustering parameterized vectors based on similarity and returning matched cluster centroids to each client without sharing raw trajectories.

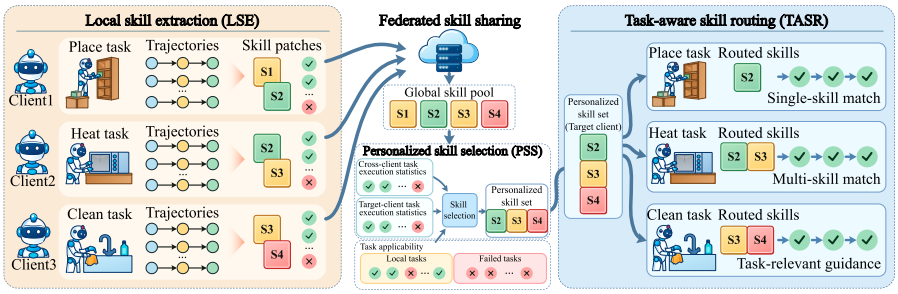


Figure 2: Illustration of the P-FedSkill framework.

2.2 Personalized Federated Learning under Heterogeneity

Personalized federated learning (PFL) has been extensively studied to address client heterogeneity in federated learning (Shen et al., 2025; Zheng et al., 2025; Huang & Shu, 2025; Liu et al., 2025; Yu et al., 2025; Wei et al., 2025). For example, Shen et al. (2025) proposed HPFL, which trains personalized plug-in modules on a shared backbone and dynamically selects suitable modules for target data at inference. Meanwhile, Liu et al. (2025) analyzed the generalization of PFL under non-convex objectives using uniform stability and decomposed the generalization error into aggregation and fine-tuning errors. While these studies have advanced personalized federated learning under client heterogeneity, they primarily address model-level personalization or theoretical analysis rather than reusable skill sharing across agents. As a result, personalized federated skill learning tailored to heterogeneous client tasks remains largely unexplored.

3 System Design

3.1 Overview

In this section, we present the architecture and workflow of the P-FedSkill framework. As illustrated in Fig. 2, P-FedSkill consists of three key modules: local skill extraction (LSE), personalized skill selection (PSS), and task-aware skill routing (TASR). The workflow of P-FedSkill comprises the following four stages: i) Each client interacts with its local environment to generate interaction trajectories; ii) LSE then extracts skill patches and task execution statistics from these trajectories and uploads them to the central server; iii) The uploaded skill patches are then processed by PSS to evaluate their performance on client tasks and construct a personalized skill set for each client. The resulting personalized skill sets are then sent back to the corresponding clients; iv) Each client receives its personalized skill set and employs TASR to select the relevant skill patches to guide agent execution. The complete procedure of P-FedSkill is summarized in **Algorithm 1** (Appendix A).

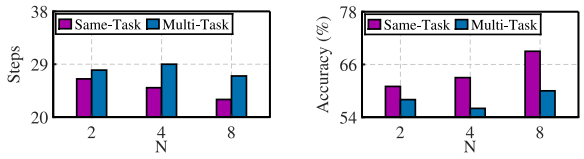
3.2 Local Skill Extraction

As discussed in Sec. 1, the narrow range of task types covered by local trajectories restricts the generalization of learned skills, thereby limiting agent performance. Given the diverse task solutions across clients, a straightforward solution is to leverage trajectories from multiple clients to learn more generalizable skills. However, raw trajectories often contain privacy-sensitive information, and directly collecting and sharing them may pose serious privacy risks (Vekaria et al., 2025), making this solution impractical for real-world deployment.

To address this challenge, we propose local skill extraction (LSE), which transforms trajectories into compact, reusable skill patches. Since local trajectories may span different task types, extracting skills from trajectories across multiple task types may introduce irrelevant or even conflicting execution strategies, thereby degrading agent performance. To investigate this insight, we conduct experiments on ALFWorld by varying the number of trajectories N for skill extraction. As shown in Fig. 3a, skills extracted from same-task trajectories lead to fewer steps for task completion as the number of trajectories increases. Fig. 3b further shows that such skills achieve higher accuracy than those extracted from multi-task trajectories. Based on these observations, LSE first groups each client's local trajectories by task type. Specifically, for client k, LSE partitions its trajectory set $\mathcal{T}_k$ into z disjoint groups according to their task types:

$$\mathcal{T}_{k,j} = \{\tau_{k,i} \in \mathcal{T}_k \mid \ell_{k,i} = j\}, \tag{1}$$

where $\ell_{k,i} \in \{1, \ldots, z\}$ denotes the task type of trajectory $\tau_{k,i}$, and $\mathcal{T}_{k,j}$ denotes the trajectory group corresponding to the j-th task type of client k.



(a) Steps. (b) Accuracy.

Figure 3: The impact of trajectory grouping on task completion steps and accuracy.

After grouping, each trajectory group contains multiple solutions for the same task type. Since raw trajectories contain extensive interaction details, directly using them to guide the agent may introduce task-irrelevant information, thereby reducing task execution efficiency. To mitigate this issue, LSE extracts skill patches from each trajectory group, with each skill patch comprising a task applicability condition and an execution strategy. Specifically, within each trajectory group $\mathcal{T}_{k,j}$, LSE uses the agent to extract the task applicability condition c_s from the task type and task descriptions, and derive the execution strategy π_s from the tool calls and action sequences shared across the trajectories while filtering trajectory-specific operations, yielding the skill patch s as

$$s = (c_s, \pi_s)\,. \tag{2}$$

Since different trajectories within the same group may adopt the same task solution, the above skill extraction process can produce redundant skill patches with identical task applicability conditions and execution strategies. To avoid redundant skill storage and sharing, LSE merges these duplicate skill patches while retaining their task execution outcomes. For a given skill patch s on client k, the numbers of successful and failed source trajectories, denoted by $n^{+}_{k,s}$ and $n^{-}_{k,s}$, are given by

$$n^{+}_{k,s} = \sum_{i \in \mathcal{I}_{k,s}} y_{k,i}, \qquad n^{-}_{k,s} = \sum_{i \in \mathcal{I}_{k,s}} (1 - y_{k,i})\,, \tag{3}$$

where $I_{k,s}$ denotes the index set of source trajectories identified by the agent as supporting skill patch s on client k, and $y_{k,i} \in \{0, 1\}$ denotes the task execution outcome recorded in trajectory $\tau_{k,i}$, with 1 and 0 indicating success and failure, respectively. Finally, each client uses the frozen text encoder $\phi(\cdot)$ (Chen et al., 2024) to encode the task description $d_{k,i}$ associated with $\tau_{k,i}$, and uploads the resulting representations to the server together with the corresponding skill patches and task execution statistics.

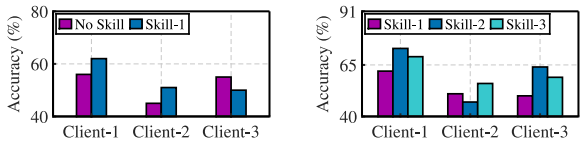
3.3 Personalized Skill Selection

As described in Sec. 3.2, LSE transforms local trajectories into reusable skill patches, enabling diverse task solutions to be shared across clients. However, due to the heterogeneity of client tasks, the same skill patch may contribute differently to agent performance across clients. To investigate this impact, we conduct experiments on ALFWorld by providing the same skill patch to different clients. Fig. 4a shows that the same skill patch does not improve agent performance on every client. Fig. 4b further shows that the best-performing skill patch varies across clients. Therefore, it is imperative to perform personalized skill selection for each client.

Motivated by these observations, we propose personalized skill selection (PSS) to evaluate the performance of each skill patch on client tasks based on its associated task execution outcomes and incorporate its task applicability condition to construct a personalized skill set for each client. PSS first collects skill patches uploaded by different clients and uses the agent to consolidate semantically equivalent skill patches to construct a global skill pool $\mathcal{S}$, while retaining their task execution statistics. Since the number of source trajectories associated with a skill patch may vary across clients, estimates based on limited local trajectories are sensitive to individual outcomes. The task execution statistics from other clients provide additional information for the same skill patch, thereby improving the reliability of its evaluation (Thapa & Li, 2025). Accordingly, PSS evaluates each skill patch using the cross-client performance score $\hat{p}_{-k,s}$, defined as

$$\hat{p}_{-k,s} = \frac{\sum_{j \in \mathcal{K}_s^{-k}} n^{+}_{j,s} + \varepsilon_0}{\sum_{j \in \mathcal{K}_s^{-k}} \left(n^{+}_{j,s} + n^{-}_{j,s}\right) + 2\varepsilon_0}, \tag{4}$$

where $\mathcal{K}_s^{-k}$ denotes the set of source clients for skill patch s, excluding target client k, and $\epsilon_0 > 0$ is a smoothing constant.



(a) Client performance. (b) Best skill patch.

Figure 4: The impact of skill patches on agent performance across different clients.

Building on $\hat{p}_{-k,s}$, PSS further uses task execution outcomes of the target client to evaluate the contribution of skill patches to local tasks. Specifically, PSS first constructs a Beta prior for each skill patch based on $\hat{p}_{-k,s}$:

$$\theta_{k,s} \sim \mathrm{Beta}\left(\eta\hat{p}_{-k,s}, \eta(1-\hat{p}_{-k,s})\right), \tag{5}$$

where $\theta_{k,s}$ denotes the task success probability associated with skill patch s on client k, and η denotes the prior strength. Since the Beta distribution is conjugate to the binomial likelihood, the Beta prior based on $\hat{p}_{-k,s}$ can be updated with the target client's local task execution outcomes for adaptive personalized skill evaluation. For clients with fewer local source trajectories, the evaluation relies more on the cross-client prior to reduce the influence of individual trajectory outcomes, while relying more on local task execution outcomes as more source trajectories become available. PSS then uses the task execution outcomes $D_{k,s} = \{y_{k,i}\}_{i \in I_{k,s}}$ of the target client to update the prior of $\theta_{k,s}$, yielding the posterior distribution

$$\theta_{k,s} \mid D_{k,s} \sim \mathrm{Beta}\left(\eta\hat{p}_{-k,s} + n^{+}_{k,s}, \eta(1-\hat{p}_{-k,s}) + n^{-}_{k,s}\right). \tag{6}$$

Based on the posterior distribution, the posterior mean $\mu_{k,s}$ estimates the task success probability associated with skill patch s on the target client, while the posterior standard deviation $\sigma_{k,s}$ quantifies its uncertainty. However, due to the limited number of local source trajectories, the posterior estimate may exhibit high uncertainty, thereby reducing the reliability of the skill evaluation. Therefore, PSS penalizes $\mu_{k,s}$ by $\sigma_{k,s}$ to obtain the performance score $\tilde{p}_{k,s}$:

$$\tilde{p}_{k,s} = \max\{0, \mu_{k,s} - \sigma_{k,s}\}. \tag{7}$$

Although Beta-based skill evaluation can mitigate the impact of limited local trajectories, a higher $\tilde{p}_{k,s}$ does not necessarily indicate greater potential to improve agent performance, as the corresponding skill patch may not match the client's local tasks or help address its failed tasks. To address this limitation, PSS further evaluates the applicability of each skill patch to local and failed tasks as

$$a_{k,s} = \rho(q_s, t_k) + \rho(q_s, f_k), \tag{8}$$

where $a_{k,s}$ denotes the applicability score of skill patch s for client k; q_s denotes the feature representation of the task applicability condition c_s obtained using the frozen text encoder $\phi(\cdot)$; t_k and f_k denote the aggregated feature representations of all local and failed local task descriptions of client k, respectively, obtained via the same encoder and mean pooling; and $\rho(\cdot,\cdot)$ denotes the normalized cosine similarity, linearly scaled to $[0, 1]$. For clients without failed tasks, the failed-task relevance term is set to zero. PSS then combines the performance score and applicability score for personalized skill selection:

$$v_{k,s} = \tilde{p}_{k,s} + a_{k,s}, \tag{9}$$

where $v_{k,s}$ denotes the personalized score of skill patch s for client k. Finally, PSS ranks the skill patches collected by the server according to $v_{k,s}$ and selects the top-K patches to construct the personalized skill set $\mathcal{S}_k$ for client k.

3.4 Task-Aware Skill Routing

Although the personalized skill set constructed by PSS provides each client with task-related skill patches, under task heterogeneity, a client may encounter multiple task types that require different execution strategies. Therefore, it is necessary to adaptively select relevant skill patches from the personalized skill set to guide the client agent for task execution.

To this end, we propose task-aware skill routing (TASR), which adaptively selects appropriate skill patches from the personalized skill set based on their relevance to the current task. Specifically, for the task t with task description d^t_k, TASR first obtains its feature representation x^t_k via $\phi(\cdot)$. It then

measures the relevance of the task applicability condition q_s to x_k^t and t_k, and combines the two relevance terms to compute the task-aware routing score of skill patch s:

$$R_{k,s}^t = \rho\big(q_s, x_k^t\big) + \rho(q_s, t_k)\,. \tag{10}$$

A larger $R_{k,s}^t$ indicates a higher routing priority. Based on the task-aware routing scores, the skill patches in the personalized skill set $\mathcal{S}_k$ are sorted in descending order of routing priority, yielding the ordered skill sequence $\{s_{k,(1)}^t, s_{k,(2)}^t, \ldots, s_{k,(|\mathcal{S}_k|)}^t\}$, given by

$$\left\{s_{k,(1)}^t, s_{k,(2)}^t, \ldots, s_{k,(|\mathcal{S}_k|)}^t\right\} = \underset{s\in\mathcal{S}_k}{\mathrm{sort}_d}\left\{R_{k,s}^t\right\}, \tag{11}$$

where $s_{k,(i)}^t$ denotes the skill patch with the i-th highest task-aware routing score, and $\mathrm{sort}_d(\cdot)$ returns the corresponding skill patches in descending order of their scores.

Due to task heterogeneity, client tasks may require different numbers of relevant skill patches for effective execution. Using a fixed number of skill patches may introduce redundant strategies under clear skill matching or omit potentially useful strategies under ambiguous skill matching. Therefore, it is necessary to select skill patches for different client tasks adaptively. The task-aware routing score reflects skill-task matching, while the gap between the top relevance scores provides a signal for adaptive skill routing. A substantially higher task relevance score indicates a clear match with the current task, suggesting that only a small number of highly relevant skills are needed for effective execution. In contrast, similar task relevance scores across multiple skill patches indicate ambiguous skill matching with the current task, suggesting that more candidate skills should be retained to avoid discarding potentially useful execution strategies. Therefore, for a personalized skill set containing only one skill patch, TASR directly routes the sole patch. Otherwise, we use the gap Δ_k^t between the highest and second-highest task relevance scores as the routing criterion, given by

$$\Delta_k^t = r_{k,(1)}^t - r_{k,(2)}^t, \tag{12}$$

where $r_{k,s}^t = \rho(q_s, x_k^t)$ denotes the relevance of skill patch s to the current task, and $r_{k,(1)}^t$ and $r_{k,(2)}^t$ denote the highest and second-highest task relevance scores, respectively. Based on Δ_k^t, TASR adaptively determines the number of routed skill patches as

$$m_k^t = \begin{cases} \min\{M_h, |\mathcal{S}_k|\}, & \Delta_k^t < \delta, \\ M_l, & \Delta_k^t \geq \delta. \end{cases} \tag{13}$$

where δ is the predefined relevance-gap threshold, and $1 \leq M_l < M_h$ denote the smaller and larger numbers of routed skill patches, respectively. TASR then selects the top-m_k^t skill patches from the ordered skill sequence to construct $\mathcal{S}_k^t = \{s_{k,(1)}^t, s_{k,(2)}^t, \ldots, s_{k,(m_k^t)}^t\}$ to guide agent execution.

3.5 Theoretical Analysis

We analyze the personalized skill selection of PSS under heterogeneous client tasks. For client k and skill patch s, let $n_{k,s} = n_{k,s}^+ + n_{k,s}^-$. The conditional oracle scores are $\tilde{p}_{k,s}^* = \theta_{k,s}$ and $v_{k,s}^* = \tilde{p}_{k,s}^* + a_{k,s}$. For M skill patches with $1 \leq K \leq M$, $\mathcal{S}_k$ and $\mathcal{S}_k^*$ contain the top-K patches according to $v_{k,s}$ and $v_{k,s}^*$, respectively, with $\epsilon_k = \max_s \epsilon_{k,s}$.

Assumption 1. *The task execution outcome satisfies* $y_{k,i} \mid \theta_{k,s} \sim \mathrm{Bernoulli}(\theta_{k,s})$ *for* $i \in \mathcal{I}_{k,s}$.

Assumption 2. *The task execution outcomes* $\{y_{k,i}\}_{i\in\mathcal{I}_{k,s}}$ *are conditionally independent given* $\theta_{k,s}$.

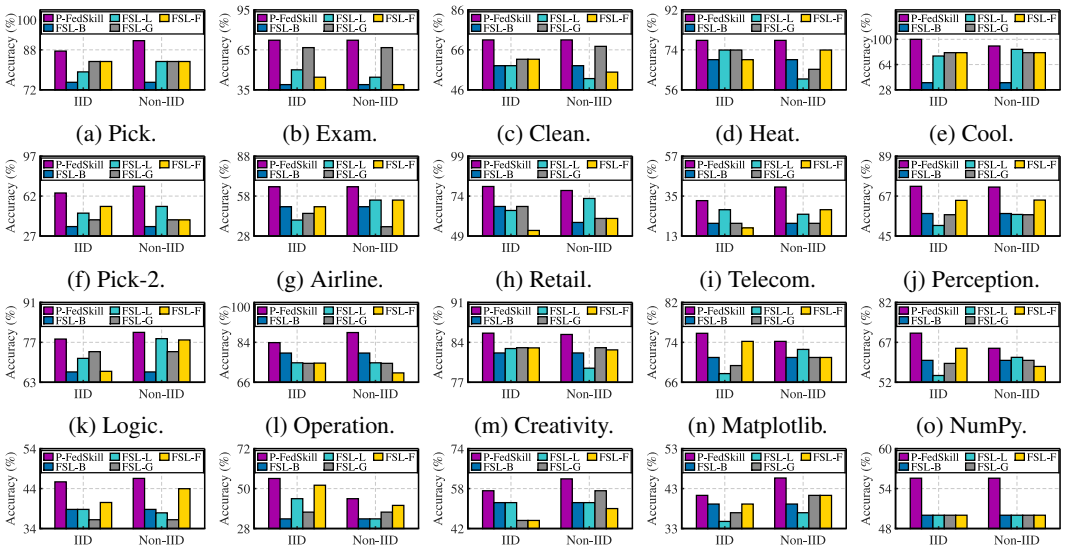
Assumption 3. *The prior strength satisfies* $0 < \eta < \infty$.

Lemma 1. *Under Assumptions 1–3, for any* $\xi \in (0,1)$*, with probability at least* $1-\xi$*,*

$$|\mu_{k,s} - \theta_{k,s}| \leq \frac{\eta\,|\hat{p}_{-k,s} - \theta_{k,s}| + \sqrt{\frac{n_{k,s}}{2}\log\frac{2}{\xi}}}{\eta + n_{k,s}}. \tag{14}$$

Lemma 2. *Under Assumptions 1–3, for any* $\xi \in (0,1)$*, with probability at least* $1-\xi$*,*

$$\left|v_{k,s} - v_{k,s}^*\right| \leq \epsilon_{k,s} = \frac{\eta\,|\hat{p}_{-k,s} - \theta_{k,s}| + \sqrt{\frac{n_{k,s}}{2}\log\frac{2M}{\xi}}}{\eta + n_{k,s}} + \frac{1}{2\sqrt{\eta + n_{k,s} + 1}}, \qquad \forall s. \tag{15}$$



(a) Pick. (b) Exam. (c) Clean. (d) Heat. (e) Cool.

(f) Pick-2. (g) Airline. (h) Retail. (i) Telecom. (j) Perception.

(k) Logic. (l) Operation. (m) Creativity. (n) Matplotlib. (o) NumPy.

(p) Pandas. (q) PyTorch. (r) SciPy. (s) Sklearn. (t) TensorFlow.

Figure 5: The performance of P-FedSkill on the ALFWorld (a–f), τ^2-Bench (g–i), GTA (j–m), and DS-1000 (n–t) datasets under IID and non-IID settings.

Theorem 1. *Under Assumptions 1–3, for any $\xi \in (0, 1)$, with probability at least $1 - \xi$, the personalized skill selection of PSS satisfies the following finite-sample suboptimality bound:*

$$0 \leq \sum_{s \in \mathcal{S}_k^*} v_{k,s}^* - \sum_{s \in \mathcal{S}_k} v_{k,s}^* \leq 2K\epsilon_k. \tag{16}$$

The proofs of Lemma 1, Lemma 2, and Theorem 1 are provided in Appendix B.

4 Experimental Setup

Datasets and Model. We adopt the ALFWorld (Shridhar et al., 2021), DS-1000 (Lai et al., 2023), GTA (Wang et al., 2024), and τ^2-Bench (Barres et al., 2026) datasets to evaluate the performance of P-FedSkill. Experiments are conducted under independent and identically distributed (IID) and non-IID settings. For the IID setting, tasks are randomly shuffled and evenly distributed across clients. For the non-IID setting, tasks are partitioned using a Dirichlet distribution with concentration parameter α. We use Qwen3.5-35B-A3B (Qwen Team, 2026) as the shared backbone agent for all methods and keep its model architecture and inference configuration fixed across experiments to ensure a fair comparison.

Baselines. To comprehensively evaluate the performance of P-FedSkill, we compare it against the following alternatives: i) **FSL-B** directly employs the backbone agent to perform tasks without using learned skills; ii) **FSL-L** employs only the skills learned from each client's local trajectories to guide task execution; iii) **FSL-G** aggregates the local skills from all clients into a shared global skill set; iv) **FSL-F** is an FSL variant based on FedSkill (Jiang et al., 2023), which clusters parameterized skill vectors by similarity and shares the skills matched to the cluster centroids across clients.

Hyper-parameters. In our experiments, we set the number of clients to 5. For the non-IID setting, the Dirichlet concentration parameter α is set to 0.1. For PSS, ϵ_0 and η are set to 1 and 4, respectively. For a fair comparison, the maximum candidate skill budget is set to $K = 6$ for all skill-based methods, while each baseline retains its original skill construction and selection mechanism. For TASR, δ, M_l, and M_h are set to 0.05, 1, and 3, respectively. Unless otherwise specified, all method hyperparameters remain unchanged.

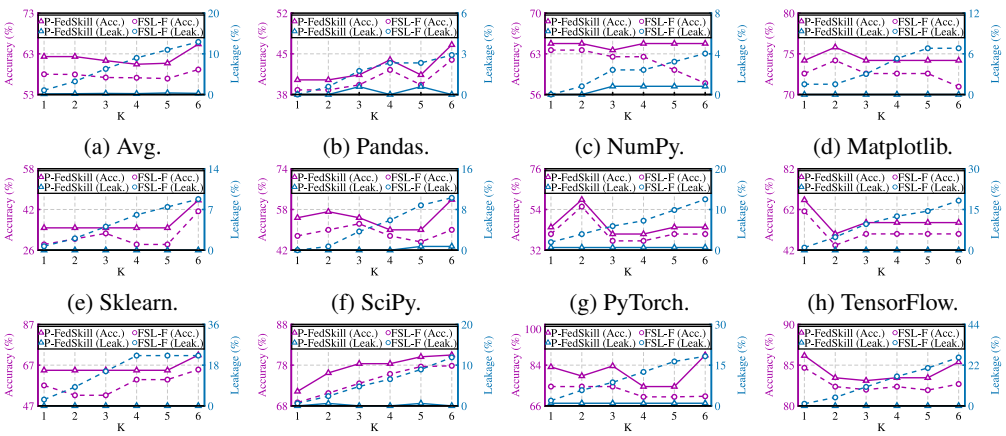
5 Results and Analysis

5.1 Superiority of P-FedSkill

As demonstrated in Fig. 5, P-FedSkill outperforms all baselines on four datasets under both IID and non-IID settings. Specifically, P-FedSkill achieves average accuracies of 78.94% (79.03%), 59.17%

Table 1: Task accuracy and leakage rate of FSL-F and P-FedSkill under the IID setting.

K	Method	DS-1000 (IID)														GTA (IID)							
		Pandas		NumPy		Matplotlib		Sklearn		SciPy		PyTorch		TensorFlow		Perception		Logic		Operation		Creativity	
		Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.	Acc.	Leak.
1	FSL-F	37.93	0.00	62.50	1.50	70.97	1.00	32.61	1.50	57.14	0.50	40.74	1.22	50.00	1.85	46.58	4.17	64.12	1.50	79.12	1.52	82.79	2.27
	P-FedSkill	**40.52**	**0.00**	**63.64**	**0.00**	**72.58**	**0.00**	**34.78**	**0.50**	**59.52**	**0.00**	**44.44**	**0.61**	**61.11**	**0.93**	**65.51**	**0.00**	**66.48**	**0.00**	**84.17**	**0.00**	**83.81**	**0.00**
2	FSL-F	37.07	1.00	61.36	2.00	69.36	1.50	32.61	3.00	52.38	1.00	40.74	3.05	50.00	4.63	52.50	9.72	69.17	3.00	79.12	3.79	83.02	4.55
	P-FedSkill	**39.66**	**0.00**	**62.50**	**0.50**	**72.58**	**0.50**	**36.96**	**0.50**	**54.76**	**0.00**	**44.44**	**0.61**	**55.56**	**0.93**	**65.51**	**0.00**	**73.15**	**0.50**	**83.69**	**0.76**	**84.20**	**0.00**
3	FSL-F	38.79	2.50	62.50	3.00	70.97	3.00	32.61	4.50	54.76	3.00	33.33	4.88	55.56	7.41	56.52	15.28	71.24	5.00	74.58	6.82	78.19	10.23
	P-FedSkill	**41.38**	**0.00**	**64.77**	**0.00**	**72.58**	**0.00**	**36.96**	**0.50**	**57.14**	**0.50**	**40.74**	**0.61**	**61.11**	**0.93**	**65.51**	**0.00**	**72.62**	**0.50**	**79.12**	**0.00**	**82.88**	**0.00**
4	FSL-F	41.38	3.00	62.50	3.00	72.58	4.00	32.61	6.00	50.00	4.50	48.15	7.32	55.56	12.04	64.21	22.22	71.22	7.00	70.03	10.61	78.19	14.77
	P-FedSkill	**45.69**	**0.00**	**64.77**	**0.50**	**74.19**	**0.00**	**36.96**	**0.50**	**52.38**	**0.00**	**51.85**	**0.61**	**61.11**	**0.93**	**65.51**	**0.00**	**72.60**	**0.50**	**79.12**	**0.00**	**82.88**	**0.00**
5	FSL-F	41.38	4.00	61.36	3.00	74.19	4.50	32.61	7.00	50.00	5.50	44.44	9.76	55.56	14.82	64.21	22.22	73.54	8.50	74.58	13.64	78.58	20.46
	P-FedSkill	**43.10**	**0.50**	**64.77**	**0.50**	**75.81**	**0.50**	**36.96**	**0.50**	**52.38**	**0.00**	**48.15**	**0.61**	**61.11**	**0.93**	**65.51**	**0.00**	**75.36**	**0.50**	**79.12**	**0.00**	**82.88**	**0.00**
6	FSL-F	40.52	4.50	64.77	3.00	74.19	5.00	39.13	9.00	45.24	7.00	51.85	10.98	50.00	19.44	64.64	22.22	66.84	11.00	74.64	16.67	83.03	26.14
	P-FedSkill	**45.69**	**0.50**	**70.45**	**0.50**	**75.81**	**0.00**	**41.30**	**0.50**	**57.14**	**0.00**	**55.56**	**0.61**	**55.56**	**0.93**	**72.38**	**0.00**	**78.15**	**0.50**	**83.87**	**0.00**	**85.61**	**0.00**



(a) Avg. (b) Pandas. (c) NumPy. (d) Matplotlib.

(e) Sklearn. (f) SciPy. (g) PyTorch. (h) TensorFlow.

(i) Perception. (j) Logic. (k) Operation. (l) Creativity.

Figure 6: Task accuracy and leakage rate of FSL-F and P-FedSkill under the non-IID setting.

(60.83%), 80.00% (81.56%), and 57.36% (56.15%) on ALFWorld, τ^2-Bench, GTA, and DS-1000 datasets under the IID (non-IID) setting. This performance advantage stems from personalized skill set construction and task-aware skill routing, which better align shared skills with heterogeneous client tasks. In contrast, FSL-F aggregates skills across clients based on representation similarity without considering client task heterogeneity, which may introduce irrelevant or even conflicting skills into local skill sets, thereby degrading agent performance.

5.2 Micro-benchmarking

Fig. 7 illustrates the impact of task heterogeneity on accuracy across the DS-1000 and GTA datasets. As α varies, the baselines exhibit noticeable performance fluctuations, whereas P-FedSkill consistently maintains high accuracy across different settings. This result demonstrates that P-FedSkill is robust to varying degrees of task heterogeneity. This performance advantage mainly stems from personalized skill selection based on task execution outcomes and task-aware skill routing, which jointly tailor skill selection to heterogeneous client tasks.

5.3 Privacy–Utility Analysis

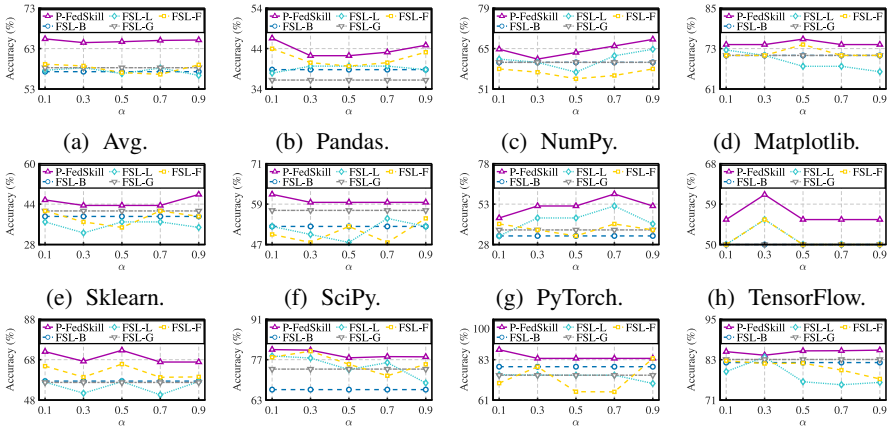
To evaluate the privacy risk of cross-client skill sharing, we conduct a Canary-based leakage experiment on DS-1000 and GTA datasets (Sun et al., 2025). Specifically, we inject unique privacy markers into local trajectories and measure leakage by counting their occurrences in the skills received by other clients. As shown in Tab. 1 and Fig. 6, FSL-F exhibits an increasing leakage rate as K grows, whereas P-FedSkill consistently maintains a near-zero leakage rate across different skill-sharing budgets while preserving competitive task accuracy.

5.4 Ablation Study

To evaluate the effectiveness of LSE, PSS, and TASR, we compare P-FedSkill with six ablation variants. Ind-LSE extracts skills independently from individual trajectories, while Mix-LSE extracts skills from mixed task trajectories. Gsr-PSS and Pbp-PSS construct a shared skill set for all clients

Table 2: Ablation study of LSE, PSS, and TASR on DS-1000 and GTA datasets.

Method	Setting	DS-1000									GTA					
		Mat.	Num.	Pan.	PyT.	Sci.	Sk.	TF	Avg.	Acc. Drop (pp)	Creat.	Logic	Oper.	Perc.	Avg.	Acc. Drop (pp)
Ind-LSE	IID	69.35	57.95	33.62	29.63	52.38	32.61	50.00	46.51	**10.85**	77.27	71.09	78.66	65.41	73.11	**6.89**
Mix-LSE	IID	66.13	61.36	38.79	37.04	54.76	30.43	50.00	48.36	**9.00**	83.08	63.68	74.56	65.46	71.69	**8.31**
LSE	**IID**	**75.81**	**70.45**	**45.69**	**55.56**	**57.14**	**41.30**	**55.56**	**57.36**	–	**85.61**	**78.15**	**83.87**	**72.38**	**80.00**	–
Ind-LSE	Non-IID	69.35	54.55	38.79	37.04	59.52	30.43	50.00	48.53	**7.62**	77.20	71.09	74.03	65.41	71.93	**9.63**
Mix-LSE	Non-IID	66.13	61.36	40.52	40.74	54.76	30.43	50.00	49.13	**7.02**	83.08	63.68	74.56	65.18	71.63	**9.93**
LSE	**Non-IID**	**74.19**	**64.77**	**46.55**	**44.44**	**61.90**	**45.65**	**55.56**	**56.15**	–	**85.38**	**80.49**	**88.41**	**71.97**	**81.56**	–
Gsr-PSS	IID	69.35	59.09	36.21	40.74	45.24	34.78	50.00	47.92	**9.44**	83.33	69.08	74.56	65.16	73.03	**6.97**
Pbp-PSS	IID	69.35	62.50	36.21	37.04	54.76	36.96	50.00	49.55	**7.81**	83.68	73.50	74.56	65.16	74.22	**5.78**
PSS	**IID**	**75.81**	**70.45**	**45.69**	**55.56**	**57.14**	**41.30**	**55.56**	**57.36**	–	**85.61**	**78.15**	**83.87**	**72.38**	**80.00**	–
Gsr-PSS	Non-IID	70.97	60.23	36.21	40.74	57.14	39.13	50.00	50.63	**5.52**	82.70	69.08	74.56	65.16	72.88	**8.68**
Pbp-PSS	Non-IID	69.35	61.36	37.07	37.04	50.00	32.61	50.00	48.20	**7.95**	82.01	73.50	74.56	65.16	73.81	**7.75**
PSS	**Non-IID**	**74.19**	**64.77**	**46.55**	**44.44**	**61.90**	**45.65**	**55.56**	**56.15**	–	**85.38**	**80.49**	**88.41**	**71.97**	**81.56**	–
Lex-TASR	IID	74.19	61.36	41.38	37.04	52.38	32.61	50.00	49.85	**7.51**	84.24	71.17	70.00	57.50	70.73	**9.27**
Fix-TASR	IID	74.19	61.36	41.38	40.74	52.38	32.61	50.00	50.38	**6.98**	83.19	71.17	70.00	57.50	70.47	**9.53**
TASR	**IID**	**75.81**	**70.45**	**45.69**	**55.56**	**57.14**	**41.30**	**55.56**	**57.36**	–	**85.61**	**78.15**	**83.87**	**72.38**	**80.00**	–
Lex-TASR	Non-IID	72.58	61.36	39.66	40.74	57.14	28.26	50.00	49.96	**6.19**	83.27	71.17	60.44	64.93	69.95	**11.61**
Fix-TASR	Non-IID	70.97	61.36	36.21	40.74	54.76	30.43	50.00	49.21	**6.94**	83.73	71.17	60.44	64.93	70.07	**11.49**
TASR	**Non-IID**	**74.19**	**64.77**	**46.55**	**44.44**	**61.90**	**45.65**	**55.56**	**56.15**	–	**85.38**	**80.49**	**88.41**	**71.97**	**81.56**	–



(a) Avg. (b) Pandas. (c) NumPy. (d) Matplotlib.

(e) Sklearn. (f) SciPy. (g) PyTorch. (h) TensorFlow.

(i) Perception. (j) Logic. (k) Operation. (l) Creativity.

Figure 7: Performance under varying task heterogeneity on the DS-1000 and GTA datasets.

using global task success rates and Beta posterior estimates, respectively. Lex-TASR uses keyword-based skill matching, while Fix-TASR selects a fixed top-ranked skill set based on the matching results. As shown in Table 2, all variants reduce task accuracy on DS-1000 and GTA under both IID and non-IID settings. Specifically, on GTA under the non-IID setting, Ind-LSE and Mix-LSE reduce the average task accuracy by 9.63 and 9.93 percentage points (pp), respectively, indicating that grouping trajectories by task type reduces interference across different tasks and avoids introducing irrelevant or conflicting execution strategies during skill extraction. Gsr-PSS and Pbp-PSS also exhibit noticeable performance degradation, validating the importance of personalized skill selection. Finally, Lex-TASR and Fix-TASR lead to drops of 11.61 and 11.49 pp, respectively, showing that effective skill routing requires not only accurate selection of task-relevant skills but also adaptive adjustment of the number of routed skills according to task-specific requirements.

6 Conclusion

In this paper, we have proposed a personalized federated skill learning framework, named P-FedSkill, aimed at facilitating personalized skill sharing under task heterogeneity across clients. P-FedSkill comprises three key components: LSE, PSS, and TASR. LSE extracts skill patches and task execution outcomes from trajectories. PSS then uses these outcomes and task applicability conditions to construct a personalized skill set for each client, while TASR adaptively selects task-relevant skill patches to guide agent execution. We further establish a finite-sample suboptimality bound, providing a theoretical guarantee for personalized skill selection in P-FedSkill. Extensive experiments across four benchmarks demonstrate the effectiveness of P-FedSkill.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and translation, literature retrieval and reference checking, drafting and refining manuscript content, providing feedback on methodological descriptions, and refining mathematical and proof presentation. All AI-assisted content was reviewed, revised, and verified by the authors. The research problem, final methodology, experimental design, implementation, results, and conclusions were determined and validated by the authors, who take full responsibility for the final manuscript.

REFERENCES

V. Barres, H. Dong, S. Ray, X. Si, and K. Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. In *Proc. ICML*, 2026.

G. Chen, H. Yang, B. Yu, and H. Ren. Intelligent OPC Engineer Assistant for Semiconductor Manufacturing. In *Proc. AAAI*, 2025.

J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of ACL*, 2024.

G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass. PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing. In *Proc. USENIX Security*, 2024.

R. Ding, W. Cheng, M. Shao, and C. Zhao. SkillGen: Learning Domain Skills for In-Context Sequential Decision Making. In *Proc. AAAI*, 2026.

H. Fu, P. Sharma, E. Stengel-Eskin, G. Konidaris, N. Le Roux, M.-A. Côté, and X. Yuan. Language-Guided Skill Learning with Temporal Variational Inference. In *Proc. ICML*, 2024.

Y. Hong, Z. Zheng, P. Chen, Y. Wang, J. Li, and C. Gan. MultiPLY: A Multisensory Object-Centric Embodied Large Language Model in 3D World. In *Proc. CVPR*, 2024.

G. Huang and T. Shu. Federated Oriented Learning: A Practical One-Shot Personalized Federated Learning Framework. In *Proc. ICML*, 2025.

Y. Jiang, W. Yu, D. Song, L. Wang, W. Cheng, and H. Chen. FedSkill: Privacy Preserved Interpretable Skill Learning via Imitation. In *Proc. KDD*, 2023.

M. Klissarov, M. Henaff, R. Raileanu, S. Sodhani, P. Vincent, A. Zhang, P.-L. Bacon, D. Precup, M. C. Machado, and P. D'Oro. MaestroMotif: Skill Design from Artificial Intelligence Feedback. In *Proc. ICLR*, 2025.

Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, W.-T. Yih, D. Fried, S. Wang, and T. Yu. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. In *Proc. ICML*, 2023.

Y. Liu, Q. Li, J. Tan, Y. Shi, L. Shen, and X. Cao. Understanding the Stability-Based Generalization of Personalized Federated Learning. In *Proc. ICLR*, 2025.

R. Lv, J. Mo, T. Chu, C. Rao, H. Jing, J. Teng, J. Chen, S. Zhang, L. Ding, S. Fang, H. Lin, Z. Dang, C. Ma, and L. Zhao. M^2-Miner: Multi-Agent Enhanced MCTS for Mobile GUI Agent Data Mining. In *Proc. ICLR*, 2026.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

L. Shen, Z. Tang, L. Wu, Y. Zhang, X. Chu, T. Qin, and B. Han. Hot-Pluggable Federated Learning: Bridging General and Personalized FL via Dynamic Selection. In *Proc. ICLR*, 2025.

M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proc. ICLR*, 2021.

J. Sun, B. Suleiman, I. Ullah, and I. Razzak. Effectiveness of Privacy-Preserving Algorithms in LLMs: A Benchmark and Empirical Analysis. In *Proc. WWW*, 2025.

M. S. Thapa and R. Li. Harnessing Heterogeneous Statistical Strength for Personalized Federated Learning via Hierarchical Bayesian Inference. In *Proc. ICML*, 2025.

Y. Vekaria, A. L. Canino, J. Levitsky, A. Ciechonski, P. Callejo, A. M. Mandalari, and Z. Shafiq. Big Help or Big Brother? Auditing Tracking, Profiling, and Personalization in Generative AI Assistants. In *Proc. USENIX Security*, 2025.

R. Villars. Agent Adoption: The IT Industry's Next Great Inflection Point. International Data Corporation (IDC), 2025. Available: `https://www.idc.com/resource-center/blog/agent-adoption-the-it-industrys-next-great-inflection-point/`.

J. Wang, Z. Ma, Y. Li, S. Zhang, C. Chen, K. Chen, and X. Le. GTA: A Benchmark for General Tool Agents. In *Proc. NeurIPS*, 2024.

T. Wang, Z. Wu, J. Liu, J. Hao, J. Wang, and K. Shao. DistRL: An Asynchronous Distributed Reinforcement Learning Framework for On-Device Control Agent. In *Proc. ICLR*, 2025a.

Z. Z. Wang, J. Mao, D. Fried, and G. Neubig. Agent Workflow Memory. In *Proc. ICML*, 2025b.

T. Wei, B. Mei, J. Lyu, R. Zhang, F. Zhou, and Y. Sun. Personalized Bayesian Federated Learning with Wasserstein Barycenter Aggregation. In *Proc. NeurIPS*, 2025.

S. Yu, G. Li, W. Shi, and P. Qi. PolySkill: Learning Generalizable Skills Through Polymorphic Abstraction for Continual Learning. In *Proc. ICLR*, 2026.

X. Yu, Z. He, Y. Sun, L. Xue, and R. Li. Understanding the Statistical Accuracy-Communication Trade-Off in Personalized Federated Learning with Minimax Guarantees. In *Proc. ICML*, 2025.

Y. Zhang and D. Yang. Searching for Privacy Risks in LLM Agents via Simulation. In *Proc. ICLR*, 2026.

A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang. ExpeL: LLM Agents Are Experiential Learners. In *Proc. AAAI*, 2024.

H. Zheng, Z. Hu, L. Yang, M. Zheng, A. Xu, and B. Wang. FedCALM: Conflict-Aware Layer-Wise Mitigation for Selective Aggregation in Deeper Personalized Federated Learning. In *Proc. CVPR*, 2025.

Y. Zhou, Q. Yang, K. Lin, M. Bai, X. Zhou, Y.-X. Wang, S. Levine, and L. E. Li. Proposer-Agent-Evaluator (PAE): Autonomous Skill Discovery for Foundation Model Internet Agents. In *Proc. ICML*, 2025.

A Algorithm of P-FedSkill

To better illustrate the proposed method, this section presents the complete algorithmic procedure of P-FedSkill. **Algorithm 1** summarizes the end-to-end workflow and the corresponding operations of each component.

B Proofs of Theoretical Results

This section provides the complete proofs of Lemma 1, Lemma 2, and Theorem 1 presented in Sec. 3.5. The analysis establishes the finite-sample behavior of personalized skill selection by bounding the estimation error of the posterior mean, propagating this error to the personalized score, and deriving the resulting top-K selection suboptimality bound.

Algorithm 1: P-FedSkill framework.

Require: $\{\mathcal{T}_k\}$, $\{d_k^t\}$, ϕ, ρ, ϵ_0, η, K, δ, M_l, M_h.
Output : $\{S_k\}$, $\{S_k^t\}$.

`Local skill extraction`**:**
- **foreach** *client* k **do**
  - **for** $j = 1, \ldots, z$ **do**
    - $\mathcal{T}_{k,j} \leftarrow \{\tau_{k,i} \in \mathcal{T}_k \mid \ell_{k,i} = j\}$;
    - **if** $\mathcal{T}_{k,j} \neq \emptyset$ **then** Extract skill patches $s = (c_s, \pi_s)$ from $\mathcal{T}_{k,j}$;
  - Merge duplicate skill patches with identical c_s and π_s;
  - **foreach** *skill patch* s **do**
    - $I_{k,s} \leftarrow \{i \mid \tau_{k,i}$ is associated with $s\}$, $n_{k,s}^+ \leftarrow \sum_{i \in I_{k,s}} y_{k,i}$, $n_{k,s}^- \leftarrow \sum_{i \in I_{k,s}} (1 - y_{k,i})$;
  - Upload $\{s, n_{k,s}^+, n_{k,s}^-\}$ and $\{\phi(d_{k,i})\}_{\tau_{k,i} \in \mathcal{T}_k}$ to the server;

`Personalized skill selection`**:**
- Consolidate uploaded skill patches into $\mathcal{S}$ and set $q_s \leftarrow \phi(c_s)$ for all $s \in \mathcal{S}$;
- **foreach** *client* k **do**
  - $t_k \leftarrow \mathrm{MeanPool}\left(\{\phi(d_{k,i})\}_{\tau_{k,i} \in \mathcal{T}_k}\right)$;
  - **if** $\exists\, i : \tau_{k,i} \in \mathcal{T}_k,\ y_{k,i} = 0$ **then** $f_k \leftarrow \mathrm{MeanPool}\left(\{\phi(d_{k,i}) \mid \tau_{k,i} \in \mathcal{T}_k,\ y_{k,i} = 0\}\right)$;
  - **foreach** $s \in \mathcal{S}$ **do**
    - **if** $I_{k,s} = \emptyset$ **then** $n_{k,s}^+ \leftarrow 0, n_{k,s}^- \leftarrow 0$;
    - $\mathcal{K}_s^{-k} \leftarrow \{j \neq k \mid I_{j,s} \neq \emptyset\}$, $\hat{p}_{-k,s} \leftarrow \frac{\sum_{j \in \mathcal{K}_s^{-k}} n_{j,s}^+ + \epsilon_0}{\sum_{j \in \mathcal{K}_s^{-k}} (n_{j,s}^+ + n_{j,s}^-) + 2\epsilon_0}$;
    - $\alpha_{k,s} \leftarrow \eta \hat{p}_{-k,s} + n_{k,s}^+$, $\beta_{k,s} \leftarrow \eta(1 - \hat{p}_{-k,s}) + n_{k,s}^-$;
    - $\mu_{k,s} \leftarrow \frac{\alpha_{k,s}}{\alpha_{k,s} + \beta_{k,s}}$, $\sigma_{k,s} \leftarrow \sqrt{\frac{\alpha_{k,s}\beta_{k,s}}{(\alpha_{k,s} + \beta_{k,s})^2(\alpha_{k,s} + \beta_{k,s} + 1)}}$;
    - $\tilde{p}_{k,s} \leftarrow \max\{0, \mu_{k,s} - \sigma_{k,s}\}$;
    - **if** $\exists\, i : \tau_{k,i} \in \mathcal{T}_k,\ y_{k,i} = 0$ **then**
      - $a_{k,s} \leftarrow \rho(q_s, t_k) + \rho(q_s, f_k)$;
    - **else**
      - $a_{k,s} \leftarrow \rho(q_s, t_k)$;
    - $v_{k,s} \leftarrow \tilde{p}_{k,s} + a_{k,s}$;
  - $S_k \leftarrow \mathrm{TopK}_{s \in \mathcal{S}}\{v_{k,s}\}$;

`Task-aware skill routing`**:**
- **foreach** *client* k *and current task* t **do**
  - $x_k^t \leftarrow \phi(d_k^t)$;
  - **foreach** $s \in S_k$ **do**
    - $R_{k,s}^t \leftarrow \rho(q_s, x_k^t) + \rho(q_s, t_k)$, $r_{k,s}^t \leftarrow \rho(q_s, x_k^t)$;
  - $\{s_{k,(i)}^t\}_{i=1}^{|S_k|} \leftarrow \mathrm{sort}_{s \in S_k}^{\downarrow}\{R_{k,s}^t\}$;
  - **if** $|S_k| = 1$ **then**
    - $m_k^t \leftarrow 1$;
  - **else**
    - $r_{k,(1)}^t, r_{k,(2)}^t \leftarrow \mathrm{Top2}_{s \in S_k}\{r_{k,s}^t\}$, $\Delta_k^t \leftarrow r_{k,(1)}^t - r_{k,(2)}^t$;
    - $m_k^t \leftarrow \begin{cases} \min\{M_h, |S_k|\}, & \Delta_k^t < \delta, \\ M_l, & \Delta_k^t \geq \delta. \end{cases}$;
  - $S_k^t \leftarrow \{s_{k,(i)}^t\}_{i=1}^{m_k^t}$;

B.1 Proof of Lemma 1

From Eq. (6), the posterior mean is

$$\mu_{k,s} = \frac{\eta \hat{p}_{-k,s} + n^+_{k,s}}{\eta + n_{k,s}}. \tag{17}$$

Hence,

$$\mu_{k,s} - \theta_{k,s} = \frac{\eta(\hat{p}_{-k,s} - \theta_{k,s}) + (n^+_{k,s} - n_{k,s}\theta_{k,s})}{\eta + n_{k,s}}. \tag{18}$$

By the triangle inequality,

$$|\mu_{k,s} - \theta_{k,s}| \le \frac{\eta\,|\hat{p}_{-k,s} - \theta_{k,s}|}{\eta + n_{k,s}} + \frac{\left|n^+_{k,s} - n_{k,s}\theta_{k,s}\right|}{\eta + n_{k,s}}. \tag{19}$$

For $n_{k,s} > 0$, by Eq. (3),

$$n^+_{k,s} = \sum_{i \in \mathcal{I}_{k,s}} y_{k,i}. \tag{20}$$

Under Assumption 1 and Assumption 2, Hoeffding's inequality gives

$$\Pr\left(\left|n^+_{k,s} - n_{k,s}\theta_{k,s}\right| \ge \sqrt{\frac{n_{k,s}}{2}\log\frac{2}{\xi}}\right) \le \xi. \tag{21}$$

Thus, with probability at least $1 - \xi$,

$$\left|n^+_{k,s} - n_{k,s}\theta_{k,s}\right| \le \sqrt{\frac{n_{k,s}}{2}\log\frac{2}{\xi}}. \tag{22}$$

Substituting Eq. (22) into Eq. (19) yields Eq. (14). For $n_{k,s} = 0$, Eq. (17) gives $\mu_{k,s} = \hat{p}_{-k,s}$, and Eq. (14) holds directly. □

B.2 Proof of Lemma 2

For the Beta posterior in Eq. (6), let

$$\alpha_{k,s} = \eta \hat{p}_{-k,s} + n^+_{k,s}, \qquad \beta_{k,s} = \eta(1 - \hat{p}_{-k,s}) + n^-_{k,s}. \tag{23}$$

Then,

$$\alpha_{k,s} + \beta_{k,s} = \eta + n_{k,s}. \tag{24}$$

The posterior variance is

$$\sigma^2_{k,s} = \frac{\alpha_{k,s}\beta_{k,s}}{(\alpha_{k,s} + \beta_{k,s})^2(\alpha_{k,s} + \beta_{k,s} + 1)}. \tag{25}$$

Since

$$\alpha_{k,s}\beta_{k,s} \le \frac{(\alpha_{k,s} + \beta_{k,s})^2}{4}, \tag{26}$$

we obtain

$$\sigma_{k,s} \le \frac{1}{2\sqrt{\eta + n_{k,s} + 1}}. \tag{27}$$

Since $\theta_{k,s} \in [0,1]$, we have $\tilde{p}^*_{k,s} = \theta_{k,s} = \max\{0, \theta_{k,s}\}$. Using the 1-Lipschitz property of $\max\{0, x\}$ in Eq. (7) gives

$$\begin{aligned} \left|\tilde{p}_{k,s} - \tilde{p}^*_{k,s}\right| &\le |\mu_{k,s} - \sigma_{k,s} - \theta_{k,s}| \\ &\le |\mu_{k,s} - \theta_{k,s}| + \sigma_{k,s}. \end{aligned} \tag{28}$$

Applying Lemma 1 with confidence parameter ξ/M and taking a union bound over the M skill patches yields, with probability at least $1 - \xi$,

$$|\mu_{k,s} - \theta_{k,s}| \le \frac{\eta\,|\hat{p}_{-k,s} - \theta_{k,s}| + \sqrt{\frac{n_{k,s}}{2}\log\frac{2M}{\xi}}}{\eta + n_{k,s}}, \qquad \forall s. \tag{29}$$

Substituting Eq. (27) and Eq. (29) into Eq. (28) gives

$$\left|\tilde{p}_{k,s} - \tilde{p}^*_{k,s}\right| \le \epsilon_{k,s}, \qquad \forall s. \tag{30}$$

Since $v_{k,s}$ and $v^*_{k,s}$ share the same applicability score $a_{k,s}$,

$$\left|v_{k,s} - v^*_{k,s}\right| = \left|\tilde{p}_{k,s} - \tilde{p}^*_{k,s}\right| \le \epsilon_{k,s}, \qquad \forall s. \tag{31}$$

□

B.3 Proof of Theorem 1

By Lemma 2, with probability at least $1 - \xi$,

$$\left|v_{k,s} - v^*_{k,s}\right| \le \epsilon_k, \qquad \forall s. \tag{32}$$

Since $\mathcal{S}_k$ contains the top-K skill patches according to $v_{k,s}$,

$$\sum_{s \in \mathcal{S}^*_k} v_{k,s} \le \sum_{s \in \mathcal{S}_k} v_{k,s}. \tag{33}$$

Using Eq. (32) and Eq. (33), we obtain

$$\begin{aligned} \sum_{s \in \mathcal{S}^*_k} v^*_{k,s} &\le \sum_{s \in \mathcal{S}^*_k} v_{k,s} + K\epsilon_k \\ &\le \sum_{s \in \mathcal{S}_k} v_{k,s} + K\epsilon_k \\ &\le \sum_{s \in \mathcal{S}_k} v^*_{k,s} + 2K\epsilon_k. \end{aligned} \tag{34}$$

Since $\mathcal{S}^*_k$ contains the top-K skill patches according to $v^*_{k,s}$,

$$\sum_{s \in \mathcal{S}^*_k} v^*_{k,s} \ge \sum_{s \in \mathcal{S}_k} v^*_{k,s}. \tag{35}$$

Combining Eq. (34) and Eq. (35), we obtain

$$0 \le \sum_{s \in \mathcal{S}^*_k} v^*_{k,s} - \sum_{s \in \mathcal{S}_k} v^*_{k,s} \le 2K\epsilon_k. \tag{36}$$

□

C Experimental Implementation Details

C.1 Datasets

To evaluate P-FedSkill under different task types and heterogeneous client tasks, we conduct experiments on four agent benchmarks, including ALFWorld, τ^2-Bench, GTA, and DS-1000. These benchmarks span embodied task execution, conversational tool use, general-purpose tool-agent capabilities, and code generation, covering diverse task structures and execution requirements.

- **ALFWorld (Embodied Task Execution).** ALFWorld (Shridhar et al., 2021) is a text-based embodied agent benchmark that requires an agent to continuously interact with the environment according to natural-language task objectives. We consider six task types, including Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place, covering different object interactions and multi-step task scenarios.
- **τ^2-Bench (Tool Use).** τ^2-Bench (Barres et al., 2026) is an interactive agent benchmark designed for realistic service scenarios, where agents complete user tasks through multi-turn interactions and tool calls. We consider three domains, including Airline, Retail, and Telecom, which involve different task objectives, interaction procedures, and tool-use requirements.

- **GTA (General Tool Agents).** GTA (Wang et al., 2024) is a benchmark for evaluating general-purpose tool agents across four capability categories: Perception, Logic, Operation, and Creativity. These categories cover diverse requirements in perception, reasoning, tool operation, and content generation.
- **DS-1000 (Code Generation).** DS-1000 (Lai et al., 2023) is a data-science code generation benchmark consisting of Python programming tasks from seven libraries, including Matplotlib, NumPy, Pandas, PyTorch, SciPy, Scikit-learn, and TensorFlow. These tasks involve different APIs and programming structures across software libraries.

C.2 Model Configurations

To ensure controlled and reproducible comparisons, all methods use the same backbone agent, while a fixed text encoder is consistently adopted wherever semantic representations are required. The compared methods differ only in their skill learning, sharing, and selection mechanisms. Unless otherwise specified, all agent-based operations use the same backbone agent and inference configuration.

- **Qwen3.5-35B-A3B.** We use Qwen3.5-35B-A3B as the backbone agent for all compared methods (Qwen Team, 2026). Qwen3.5-35B-A3B adopts a sparse Mixture-of-Experts (MoE) architecture with approximately 35B total parameters and 3B activated parameters per forward pass. Its language model contains 40 layers and 256 routed experts, with 8 routed experts and one shared expert activated for each token. The model supports long-context processing, instruction following, and tool interaction, enabling its use as a unified backbone across different agent benchmarks. The backbone model remains unchanged for all compared methods, while only their skill learning, sharing, and selection mechanisms are varied.
- **BGE-M3.** We use the pretrained BGE-M3 model (Chen et al., 2024) as a frozen text encoder $\phi(\cdot)$ for all skill applicability conditions and task descriptions. The representations q_s and x_k^t are obtained by encoding c_s and d_k^t with $\phi(\cdot)$, respectively, while t_k and f_k are obtained by mean pooling the encoded descriptions of all local tasks and failed local tasks, respectively. The encoder remains frozen throughout all experiments, with the same encoding and aggregation settings used across all clients and datasets.

C.3 Experimental Configurations

This subsection details the experimental configurations used for all evaluations, including data splits, client task assignments, inference settings, and implementation environments.

- **Data Split.** We strictly separate the data used for trajectory generation and skill learning from the final test data. For ALFWorld, we use the official training data for trajectory generation and skill learning, and evaluate the methods on the official unseen tasks. For τ^2-Bench and GTA, within each domain or task type, respectively, 60% of the tasks are used for trajectory generation and skill learning, while the remaining 40% are used for final evaluation. For DS-1000, since no independent training set is provided, we group samples according to their software library and original problem source, and use 60% of the samples for trajectory generation and skill learning and the remaining 40% for final evaluation. Different variants derived from the same original problem are always assigned to the same split to prevent source-related tasks from appearing in both the skill-learning and test sets.
- **Client Task Assignment.** All experiments involve five clients. Under the IID setting, tasks are evenly distributed across clients. Under the non-IID setting, tasks are partitioned according to their task types, and a Dirichlet distribution is used to determine the task proportions assigned to different clients, thereby constructing heterogeneous local task distributions. The concentration parameter is set to $\alpha = 0.1$ in the main experiments. In the task heterogeneity evaluation, we further consider $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ to evaluate the performance under different degrees of task heterogeneity. All reported results are averaged over three runs with different random seeds under identical data splits and client task assignments across methods.

- **Inference Configuration.** All compared methods use the same inference configuration. The inference temperature is set to 0. For interactive tasks, the maximum number of execution steps is set to 100, and the maximum length of each model generation is set to 512 tokens.
- **Implementation Environment.** All experiments are conducted on a server equipped with five NVIDIA L40 GPUs, each with approximately 46 GB of memory. The server runs Ubuntu 22.04.5 LTS, and the experiments are implemented with Python 3.10.20 and PyTorch 2.11.0 using CUDA 13.0 runtime for GPU acceleration. The large language model environment includes Transformers 4.56.2, vLLM 0.21.0, and Accelerate 1.13.0. All compared methods are evaluated under the same hardware and software environment with the same Qwen3.5-35B-A3B backbone agent and inference configurations to ensure consistent experimental conditions.

C.4 Privacy Evaluation Settings

To evaluate the potential privacy leakage caused by cross-client skill sharing, we further conduct Canary-based privacy evaluations on DS-1000 and GTA datasets.

- **Canary Injection.** For each client in each task domain, we select up to 10 local training trajectories and inject a unique task-irrelevant Canary marker into each selected trajectory (Sun et al., 2025). Different trajectories are assigned distinct Canary markers, allowing us to trace whether private information is propagated to other clients through skill sharing. The Canary markers are inserted only into local trajectories used for skill learning and do not alter task labels, reference answers, or test tasks.
- **Skill Sharing Budget.** We use K to denote the number of skills retained by each client and consider $K \in \{1, 2, 3, 4, 5, 6\}$. For each value of K, FSL-F and P-FedSkill use the same data split, client task assignment, and backbone agent. By varying K, we jointly evaluate task accuracy and privacy leakage under different skill-sharing scales.
- **Privacy Leakage Rate (Leakage, %).** For each Canary, we examine whether its complete unique marker appears in the skills received by clients other than its source client. If the complete Canary marker appears in the skill content received by a non-source client, the corresponding cross-client check is counted as one privacy leakage event. A Canary retained in the skills of its source client is not considered cross-client leakage. Let N_c denote the total number of Canary markers in the corresponding task domain, and let K_c denote the number of clients. Each Canary is checked against the other $K_c - 1$ clients. Therefore, the total number of cross-client checks is

$$N_{\text{check}} = N_c(K_c - 1). \tag{37}$$

The privacy leakage rate is defined as

$$\text{Leakage} = \frac{N_{\text{leak}}}{N_{\text{check}}} \times 100\%, \tag{38}$$

where N_{leak} denotes the number of cross-client checks in which the complete Canary marker is detected. In our experiments, $K_c = 5$, and each Canary is therefore independently checked against the remaining four clients. A lower leakage rate indicates that fewer injected Canary markers are propagated to other clients through cross-client skill sharing.