BACAM: BEHAVIOR-AWARE CONTINUAL AGENT MERGING FOR MULTI-TURN INTERACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging offers a way to integrate the capabilities of specialized experts, but existing agent merging methods typically require all of them to be available at once. We study continual agent merging, which integrates incoming experts sequentially without retaining previously merged experts. Yet merging in parameter space or feature subspaces does not ensure that the merged model acquires an incoming expert's behavior on interaction trajectories. Moreover, updates toward a new expert can disrupt the merged model's previously integrated interactive behavior. Therefore, we propose Behavior-Aware Continual Agent Merging (BACAM), which learns parameter-wise merging gates from candidate-generated trajectories using expert-guided behavioral supervision. Task-level stability–plasticity control and tensor-level conflict-aware update budgets limit interference with existing capabilities while allowing new ones to be acquired. The learned gates are folded into the model weights without additional inference-time parameters. Across four interactive tasks—web shopping, tool use, information retrieval, and embodied interaction—BACAM achieves an average success rate of 62.82%, exceeding the strongest evaluated merging baseline by 21.69 percentage points.

1 INTRODUCTION

Large language model (LLM) agents can reason and act through multi-turn interactions with tools and environments (Yao et al., 2022b; Liu et al., 2024). Recent work has produced specialized experts for tool use, information search, embodied interaction, and web shopping (Qian et al., 2025; Chen et al., 2025; Feng et al., 2025; Shridhar et al., 2020; Yao et al., 2022a). However, current LLM-based agents often exhibit limited cross-environment robustness (Zeng et al., 2024). Joint multi-task RL can train an agent across environments, but requires substantial engineering and computation (Chen et al., 2026; Luo et al., 2025; Liu et al., 2025).

Model merging offers a more efficient approach for models to adapt to diverse environments: it combines specialized models directly in parameter space to support a broader range of tasks, without retraining on a mixture of all tasks (Yang et al., 2026). Existing methods such as Task Arithmetic, TIES-Merging, DARE, and TSV-Merge have demonstrated the feasibility of integrating multiple task-specific capabilities through parameter merging (Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2024; Gargiulo et al., 2025).

Model merging has increasingly been extended to LLM-based agents, where current approaches primarily fall into two categories: parameter-space and subspace-based merging. Parameter-space approaches modify model parameters directly, either separating shared and task-specific regions for selective aggregation (Yuan et al., 2026) or transferring behavior-critical neurons identified from activation patterns (Feng et al., 2026). Subspace-based approaches preserve existing agent behavior by identifying behavior-critical feature subspaces and projecting new task vectors into the corresponding null spaces (Wang et al., 2026). These approaches require jointly available experts for batch merging; repeating the merge as new tasks arise becomes costly (Tang et al., 2025).

To this end, we formulate a novel task: continual agent merging for interactive environments. As shown in Figure 1, batch merging combines all experts at once, whereas continual merging updates the current model as new experts arrive without retaining earlier experts.

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

This continual setting presents two coupled challenges. First, merging in parameter space or feature subspaces does not ensure that the merged model acquires the incoming expert's interactive capability. The merged model's actions determine subsequent observations and may lead to trajectories different from those of the expert. Parameter merging must therefore transfer the expert's behavior to the trajectories induced by the merged model itself. Second, each merge moves the parameters toward the incoming expert, while previously integrated capabilities are encoded in the current merged model. Even a small parameter change can alter an action on an earlier task and redirect the subsequent trajectory, disrupting established interactive behavior. The merge must therefore acquire the incoming expert's capability while preserving the current model's established interactive behavior.

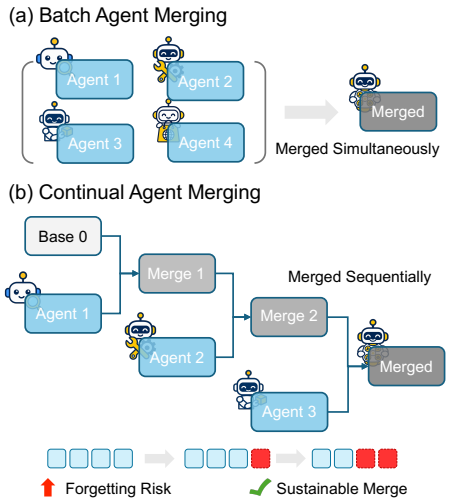

Figure 1: (a) Batch agent merging combines all experts at once. (b) Continual agent merging integrates each incoming expert into the current merged model. Red blocks indicate potential forgetting of earlier capabilities.

To address these challenges, we propose Behavior-Aware Continual Agent Merging (BACAM), which learns parameter-wise gates between the current merged model and each incoming expert. BACAM comprises two components. First, task-level stability–plasticity control supervises gate updates on candidate-generated interaction trajectories: the incoming expert guides new-task acquisition, while the current merged model constrains behavioral drift on earlier tasks. Second, tensor-level conflict-aware plasticity budgeting assigns smaller update budgets to tensors where these goals conflict, protecting existing capabilities while allowing adaptation in less sensitive regions. Together, these components enable behavior-aware and conflict-sensitive continual agent merging. We evaluate BACAM across expert counts and merging orders. Under the default order, BACAM has the highest Avg SR at each expert count and exceeds WUDI-Merging by 21.69 percentage points after four experts. Our contributions are:

1. We formulate continual agent merging for multi-turn interactive tasks and assess capability acquisition and retention over complete interaction trajectories.
2. We introduce BACAM, which learns parameter-wise merging gates from behavioral supervision on candidate-generated trajectories. Task-level constraints guide new-task acquisition while limiting drift on earlier tasks, and tensor-level conflict-aware budgets restrict updates where these goals interfere.
3. Across four interactive tasks, BACAM achieves the highest average success rate among the evaluated merging methods at each stage of the default merging sequence. We also track capability acquisition and retention across four arrival orders and use ablations to measure the contribution of each control component.

2 Related Work

2.1 Model Merging

Model merging combines specialized models in parameter space to integrate their capabilities (Yang et al., 2026). Task Arithmetic represents fine-tuning updates as task vectors and adds them together (Ilharco et al., 2022), but direct addition can cause interference. TIES-Merging trims small updates and resolves sign conflicts (Yadav et al., 2023), while DARE sparsifies updates before merging (Yu et al., 2024). TSV-Merge instead reduces interference by decorrelating task singular vectors (Gargiulo et al., 2025). Rather than applying predefined rules to task updates, another line of work optimizes the merge directly: WUDI-Merging minimizes interference among task vectors (Cheng et al., 2025), AdaMerging optimizes coefficients using prediction entropy (Yang et al.,

2024b), and Expert Merging learns coefficients through unsupervised expert alignment (Zhang et al., 2026). These methods improve the combination of model parameters, but do not directly assess whether an agent retains its behavior over multi-turn interactions.

2.2 Agent Model Merging

Agent model merging applies these ideas to specialized LLM agents. RAM separates sparse RL task vectors into shared and task-specific regions, then selectively aggregates them to preserve domain-specific behavior (Yuan et al., 2026). ARM identifies role-conditioned neurons from activation patterns and transfers them to a generalist agent (Feng et al., 2026). M2A takes a subspace approach: it identifies behavior-critical features and projects new task vectors into their null spaces to limit disruption of existing capabilities (Wang et al., 2026). These methods merge jointly available source models and do not address sequentially arriving experts.

2.3 Continual Model Merging

Continual model merging integrates task-specific models sequentially. OPCM projects incoming task vectors onto directions orthogonal to accumulated updates (Tang et al., 2025). DOP applies separate projections to protect existing knowledge and acquire new capabilities without task data (Yang et al., 2025). MINGLE combines gated low-rank experts for test-time adaptation (Qiu et al., 2025), while NUFILT couples null-space filtering with a lightweight adapter to recover useful new-task directions (Qiu et al., 2026). These methods primarily evaluate non-interactive prediction tasks. Whether they preserve an agent's earlier capabilities over full interactions remains unclear: one changed action can redirect the subsequent observations and decisions.

3 Behavior-Aware Continual Agent Merging

3.1 Problem Definition

We consider an ordered sequence of interactive tasks $\tau = 1, \ldots, T$, such as web shopping, tool use, information retrieval, and embodied interaction. Task τ provides a specialized expert E_τ, an interaction environment, and unlabeled instances $\mathcal{D}_\tau$ without accompanying ground-truth responses or expert demonstration trajectories. All experts share the same architecture and tokenizer. Let M_t be the merged model at stage t. We start with $M_1 = E_1$. At each subsequent stage $t \geq 2$, we merge the current model M_{t-1} with the incoming expert E_t to obtain M_t. Earlier experts are no longer retained, but the instances and environments for all integrated tasks remain available for collecting trajectories, following the idea of replaying retained task inputs (Marouf et al., 2025). Behavioral supervision is provided by the incoming expert and the current merged model on candidate-generated trajectories.

The goal is for M_t to acquire E_t's capability on the new task while preserving M_{t-1}'s capabilities on every earlier task $\tau < t$. We evaluate both acquisition and preservation in multi-turn interactions. Appendix A gives the formal interaction model.

3.2 Framework Overview

BACAM constructs a merged candidate from M_{t-1} and E_t and rolls it out on the new and earlier tasks (Figure 2). Within task-level control, **New-task Plasticity Control (NPC)** uses E_t to guide capability acquisition, while **Old-task Stability Control (OSC)** uses M_{t-1} to limit behavioral drift on earlier tasks. **Tensor-Level Conflict-Aware Plasticity Budgeting (CAPB)** restricts gate updates in tensors where these two goals conflict.

To define the merged candidate, let $\theta_{t-1}^{\text{merged}}$ and θ_t denote the parameters of M_{t-1} and E_t, respectively, and set $\Delta\theta_t = \theta_t - \theta_{t-1}^{\text{merged}}$. A parameter-wise gate g determines the candidate's weights:

$$\theta_t^{\text{merged}}(g) = \theta_{t-1}^{\text{merged}} + g \odot \Delta\theta_t, \qquad g \in [0,1]^{|\theta|}. \tag{1}$$

The current merged model and incoming expert remain frozen, and only g is optimized. For parameter j, $g_j = 0$ retains the old-model value and $g_j = 1$ selects the new-expert value. Compared

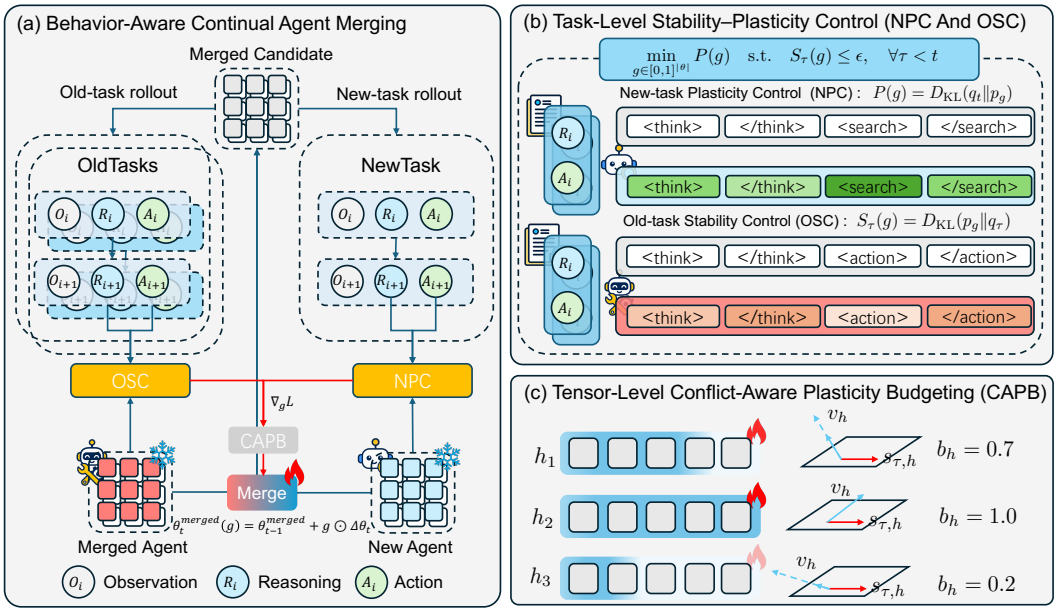

Figure 2: Overview of BACAM. (a) BACAM constructs a merged candidate from the current merged model and incoming expert, collects interaction trajectories, and optimizes the gates through task- and tensor-level control. (b) On candidate-generated histories, new-task plasticity control (NPC) uses forward KL to acquire the incoming expert's behavior, while old-task stability control (OSC) constrains reverse KL to the current merged model to preserve existing behavior. (c) Tensor-level conflict-aware plasticity budgeting (CAPB) uses local gradient probes to identify conflicts and restrict gate updates toward the incoming expert in conflicting tensors.

with coarse-grained merging coefficients, the parameter-wise gate can express different stability–plasticity trade-offs within the same tensor (Yang et al., 2024b; Zhang et al., 2026).

3.3 Task-Level Stability–Plasticity Control

Behavioral supervision on candidate trajectories. The candidate's actions affect subsequent observations, so fixed prompts or expert-generated trajectories may not cover the histories it visits. For each task $\tau \le t$, we roll out the candidate policy $\pi_{\theta_t^{\text{merged}}(g)}$ in the task environment using instances from $\mathcal{D}_\tau$, obtaining trajectories $\mathcal{R}_\tau$. We write the decision points of each trajectory ξ as $((x_i, y_i))_{i=1}^{|\xi|}$, where x_i contains the interaction history and current observation, and y_i is the candidate's reasoning and action. The incoming expert supervises task t; the current merged model supervises tasks $\tau < t$. At response token k of decision point i, the candidate and teacher distributions are

$$\begin{aligned} p_{g,i,k} &= \pi_{\theta_t^{\text{merged}}(g)}(\cdot \mid x_i, y_{i,<k}), \\ q_{t,i,k} &= \pi_{\theta_t}(\cdot \mid x_i, y_{i,<k}), \\ q_{\tau,i,k} &= \pi_{\theta_{t-1}^{\text{merged}}}(\cdot \mid x_i, y_{i,<k}), \quad \tau < t. \end{aligned} \tag{2}$$

At each token, the candidate and teacher are evaluated on the same candidate-generated prefix; only the candidate interacts with the environment (Agarwal et al., 2024; Song & Zheng, 2026).

Behavior-aware loss aggregation. To form task-level losses from these distributions, we account for the different roles of response tokens. Reasoning text affects the environment indirectly, whereas search queries, function calls, and environment actions directly determine the next observation. Using task-format markers, we partition the response positions at decision point i into critical behavior tokens C_i and remaining tokens N_i. Let $\mathcal{B}_i \subseteq \{C_i, N_i\}$ contain the nonempty groups. For a token-level divergence D, we average uniformly within each group and then across the available groups:

$$\ell_i^D(g; q_\tau) = \mathbb{E}_{B \in \mathcal{B}_i} \mathbb{E}_{k \in B} \left[D(p_{g,i,k}, q_{\tau,i,k}) \right]. \tag{3}$$

This prevents ordinary text tokens from overwhelming sparse but consequential action tokens. We also give each trajectory equal weight:

$$Z_\tau^D(g; q_\tau) = \mathbb{E}_{\xi \in \mathcal{R}_\tau} \mathbb{E}_{i=1}^{|\xi|} \left[\ell_i^D(g; q_\tau) \right], \tag{4}$$

Within a trajectory, we average over its decision points, so longer interactions do not dominate the loss. We use this aggregation for both $P(g)$ and $S_\tau(g)$, with different KL directions for the new and earlier tasks.

Plasticity and stability losses. For NPC, we use forward KL from the incoming expert to the candidate:

$$P(g) = Z_t^{D_{\mathrm{F}}}(g; q_t), \qquad D_{\mathrm{F}}(p_{g,i,k}, q_{t,i,k}) = D_{\mathrm{KL}}(q_{t,i,k} \| p_{g,i,k}), \tag{5}$$

This penalizes the candidate when it assigns insufficient probability to tokens preferred by the incoming expert, encouraging it to acquire the expert's behavior on the histories it visits (Agarwal et al., 2024).

For OSC on each earlier task $\tau < t$, we use reverse KL from the candidate to the current merged model:

$$S_\tau(g) = Z_\tau^{D_{\mathrm{R}}}(g; q_\tau), \qquad D_{\mathrm{R}}(p_{g,i,k}, q_{\tau,i,k}) = D_{\mathrm{KL}}(p_{g,i,k} \| q_{\tau,i,k}), \tag{6}$$

This penalizes probability mass assigned to tokens that the current merged model considers unlikely. Limiting $S_\tau(g)$ therefore constrains behavioral drift on task τ and provides the stability signal for preserving its previously integrated capability (Gu et al., 2024).

Task-level objective and optimization. To acquire the incoming capability while preserving earlier ones, BACAM minimizes $P(g)$ subject to a behavioral-drift budget ϵ for each old task:

$$\min_{g \in [0,1]^{|\theta|}} P(g) \quad \text{s.t.} \quad S_\tau(g) \le \epsilon, \quad \forall \tau < t, \tag{7}$$

We optimize this constrained objective using a Lagrangian with one nonnegative multiplier λ_τ for each old task:

$$\mathcal{L}(g, \lambda) = P(g) + \sum_{\tau < t} \lambda_\tau \big(S_\tau(g) - \epsilon \big), \tag{8}$$

Gate updates descend $\mathcal{L}$, while projected dual-ascent steps increase λ_τ when $S_\tau(g) > \epsilon$ and decrease it when the drift is below budget. Each old task therefore adjusts its own stability weight as merging proceeds. The update rules are summarized in Algorithm 1 and detailed in Appendix A.3.

3.4 Tensor-Level Conflict-Aware Plasticity Budgeting

Each task-level multiplier sets one stability weight for an old task across the whole model. It cannot distinguish tensors in which acquiring the new task would disrupt that task. CAPB therefore assigns a separate plasticity budget to each tensor to limit such updates.

Tensor-level conflict estimation. We estimate local conflict after a temporary step toward the new task. During each refresh round, the collected trajectories remain fixed while we compute gradients. Let g_{start} be the current gate and $v = \nabla_g P(g_{\mathrm{start}})$ the new-task gradient. The probe gate is

$$g_{\mathrm{probe}} = \mathrm{clip}\left(g_{\mathrm{start}} - \eta_{\mathrm{probe}} \frac{v}{\|v\|_2}, 0, 1 \right). \tag{9}$$

Here, $\mathrm{clip}(\cdot, 0, 1)$ clips each gate component to $[0, 1]$. Let v_h be the part of v corresponding to tensor h. At g_{probe}, we compute the old-task stability gradient $s_{\tau,h} = \nabla_{g_h} S_\tau(g_{\mathrm{probe}})$. A negative inner product $\langle v_h, s_{\tau,h} \rangle$ means that further movement along $-v_h$ would locally increase old-task drift. We measure this conflict by

$$c_{\tau,h} = \max\left(0, \frac{-\langle v_h, s_{\tau,h} \rangle}{\|v_h\|_2 \|s_{\tau,h}\|_2 + \delta} \right). \tag{10}$$

Here, $\delta > 0$ is a small constant for numerical stability.

Algorithm 1: Behavior-Aware Continual Agent Merging

Input: $\theta_{t-1}^{\text{merged}}$, θ_t, unlabeled data $\{\mathcal{D}_\tau\}_{\tau=1}^{t}$, rounds N

Initialize $\Delta\theta_t \leftarrow \theta_t - \theta_{t-1}^{\text{merged}}$, $g \leftarrow \mathbf{0}$, $\lambda_\tau \leftarrow 0$ for $\tau < t$;
for $n = 1, \ldots, N$ **do**
  $\theta_t^{\text{merged}}(g) \leftarrow \theta_{t-1}^{\text{merged}} + g \odot \Delta\theta_t$;
  Collect $\mathcal{R}_\tau$ from $\mathcal{D}_\tau$, $\tau = 1, \ldots, t$;
  Construct tensor-wise plasticity budgets $\{b_h\}$ (Eqs. 9–11);
  for *each gate-update step* **do**
    $P(g) \leftarrow Z_t^{D_{\text{F}}}(g; q_t)$;
    $S_\tau(g) \leftarrow Z_\tau^{D_{\text{R}}}(g; q_\tau)$, $\forall \tau < t$;
    $\mathcal{L}(g, \lambda) \leftarrow P(g) + \sum_{\tau<t} \lambda_\tau (S_\tau(g) - \epsilon)$;
    $\Delta g \leftarrow -\eta_g \nabla_g \mathcal{L} / \|\nabla_g \mathcal{L}\|_2$;
    $g_h \leftarrow \text{clip}(g_h + b_h[\Delta g_h]_+ + [\Delta g_h]_-, 0, 1)$, $\forall h$;
    $\lambda_\tau \leftarrow \text{clip}(\lambda_\tau + \eta_\lambda [S_\tau(g) - \epsilon], 0, \lambda_{\max})$, $\forall \tau < t$;
return $\theta_t^{\text{merged}}(g)$;

Plasticity budget construction. The conflict score captures direction but not how strongly the old task responds to changes in tensor h. We therefore combine $c_{\tau,h}$ with the gradient magnitudes $V_h = \|v_h\|_2$ and $U_{\tau,h} = \|s_{\tau,h}\|_2$:

$$b_{\tau,h} = 1 - \frac{c_{\tau,h} U_{\tau,h}}{V_h + U_{\tau,h} + \delta}, \qquad b_h = \min_{\tau<t} b_{\tau,h}. \tag{11}$$

The budget satisfies $0 \le b_h \le 1$. Tensors with no detected conflict have $b_h = 1$; an opposing gradient and stronger old-task sensitivity reduce the budget. Taking the minimum gives each tensor the most restrictive budget across old tasks. The probe only estimates these budgets; the actual gate updates start from g_{start}.

Budgeted gate updates. Let Δg be the normalized descent step computed from the task-level objective $\mathcal{L}$, as in Algorithm 1. We apply the tensor budgets only to positive gate updates, which move parameters toward the new expert:

$$\Delta g_{\text{safe},h} = b_h[\Delta g_h]_+ + [\Delta g_h]_-, \qquad g_h \leftarrow \text{clip}(g_h + \Delta g_{\text{safe},h}, 0, 1)\,, \tag{12}$$

where $[z]_+ = \max(z, 0)$ and $[z]_- = \min(z, 0)$ are applied elementwise. Negative updates move the gate back toward the old model and are therefore not restricted by the plasticity budget.

Algorithm 1 summarizes the rollout, budget, and gate-update steps. After N refresh rounds, the learned gate is folded into the model parameters, introducing no additional inference-time parameters or computation.

4 Experiments

4.1 Experimental Setup

Backbones. We merge four specialized experts built on Qwen2.5-7B-Instruct (Yang et al., 2024a): ReSearch for information retrieval (Chen et al., 2025), ToolRL for tool use (Qian et al., 2025), and two GiGPO experts for WebShop and ALFWorld (Feng et al., 2025). They share the same architecture and tokenizer. The default merging order is WebShop → Tool → Search → ALFWorld, with the WebShop expert as the initial model.

Evaluations. We evaluate shopping with WebShop (Yao et al., 2022a), tool use with BFCL v3 `multi_turn_base` (Patil et al., 2025), information retrieval with MuSiQue (Trivedi et al., 2022), and embodied interaction with ALFWorld (Shridhar et al., 2020). The primary metric is success rate (SR, % ↑) under each benchmark's criterion; for ALFWorld, we report both IID (`valid_seen`) and OOD (`valid_unseen`) SR. Avg SR is the mean of the five reported columns, counting the

Table 1: Final SR (%) after four experts. Continual methods use WTSA; batch methods use the same experts. Avg averages five task columns. Yellow highlights expert domains; blue highlights BACAM. **Bold**/underline mark the best/second-best merging results.

Method	WebShop	Tool	Search	ALFWorld		Avg ↑
				IID	OOD	
Base model and individual experts						
Qwen2.5-7B-Instruct	0.00	7.00	12.49	20.71	15.67	11.18
WebShop expert	78.00	4.00	7.34	24.29	20.90	26.90
Tool expert	0.00	30.00	12.74	10.71	13.43	13.38
Search expert	0.80	11.00	24.34	9.29	10.45	11.18
ALFWorld expert	1.80	16.00	14.92	98.57	91.04	44.47
General model merging						
Weight Average	4.80	0.00	21.76	35.00	37.31	19.78
Task Arithmetic	32.00	2.00	20.82	66.43	58.21	35.89
TIES-Merging	38.40	0.00	15.17	53.57	56.72	32.77
TA w/ DARE	34.00	0.00	20.38	69.29	55.22	35.78
TIES w/ DARE	36.20	1.00	16.96	59.29	55.97	33.88
TSV-Merge	39.40	1.00	23.10	68.57	66.42	39.70
Iso-C	24.60	1.00	23.85	59.29	55.97	32.94
WUDI-Merging	38.20	0.00	**25.14**	72.14	70.15	41.13
AdaMerging	16.80	1.00	15.27	62.86	59.70	31.13
AdaMerging++	21.40	1.00	18.69	62.86	60.45	32.88
Expert Merging	5.60	1.00	18.84	28.57	32.09	17.22
Agent model merging						
RAM	34.60	1.00	21.02	56.43	52.99	33.21
RAM++	37.20	1.00	20.33	61.43	50.00	33.99
Continual model merging						
OPCM	22.60	1.00	21.52	46.43	41.79	26.67
NUFILT	10.00	1.00	20.53	36.43	32.84	20.16
BACAM (ours)	**77.20**	**38.00**	23.75	**88.57**	**86.57**	**62.82**

two ALFWorld splits separately. We also report WebShop task score (TS ↑) and average interaction steps (Round ↓) for WebShop and ALFWorld. All methods use the same held-out instances and benchmark settings. Dataset sizes, scoring rules, and generation settings are given in Appendix B.

Baselines. We compare methods with different access to task instances during merging. Weight averaging (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2022), TIES-Merging (Yadav et al., 2023), DARE variants (Yu et al., 2024), TSV-Merge (Gargiulo et al., 2025), Iso-C (Marczak et al., 2025), WUDI-Merging (Cheng et al., 2025), RAM and RAM++ (Yuan et al., 2026), OPCM (Tang et al., 2025), and NUFILT (Qiu et al., 2026) operate on model parameters without task instances. AdaMerging and AdaMerging++ (Yang et al., 2024b) instead minimize prediction entropy on unlabeled prompts, while Expert Merging (Zhang et al., 2026) aligns hidden states and output distributions using unlabeled calibration data. We also evaluate the base model and each individual expert. The final-model comparison includes both batch and continual methods using the same four experts. Table 5 details their merge-time data and supervision.

Implementation details. We select 256 candidate-generated states per task stream in each round and 24 trajectories per stream for local probing. The gate learning rate is 2×10^{-2} with cosine decay, and the stability budget is $\epsilon = 5 \times 10^{-3}$. The dual update uses $\eta_\lambda = \lambda_{\max} = 1$. Training uses 8× RTX 5090 GPUs. Additional settings are provided in Appendix B.

4.2 Performance Comparison

Overall performance. BACAM achieves an Avg SR of 62.82%, exceeding the strongest measured baseline, WUDI-Merging, by 21.69 percentage points (Table 1). The largest differences are on Tool and WebShop: BACAM reaches 38.00% Tool SR, compared with 0–2% for the evaluated merging baselines, and 77.20% WebShop SR, compared with 39.40% for TSV-Merge. On Search, BACAM scores 23.75%, below WUDI-Merging's 25.14%.

Comparison with specialists. BACAM remains close to the WebShop and Search experts, trailing them by 0.80 and 0.59 percentage points, respectively. Tool SR exceeds its expert by 8.00 points, whereas ALFWorld IID/OOD SR remains 10.00/4.47 points below its expert. Additional TS and Round results are reported in Table 7.

Two- and three-expert merging. Appendix Table 6 reports the first two merging stages in the default order. After adding Tool, BACAM reaches 32.00% Tool SR while retaining 78.60% WebShop SR; after adding Search, these scores are 37.00% and 78.40%, with 24.10% Search SR. The other merging methods reach at most 2.00% Tool SR at either stage. These results show that BACAM acquires the incoming capabilities while preserving the initial WebShop capability through the first three experts.

4.3 Continual Merging Analysis

We evaluate BACAM under four expert orders: WTSA, ASTW, TWAS, and SAWT, and track SR after each integration (Figure 3). Appendix Table 8 gives the full scores.

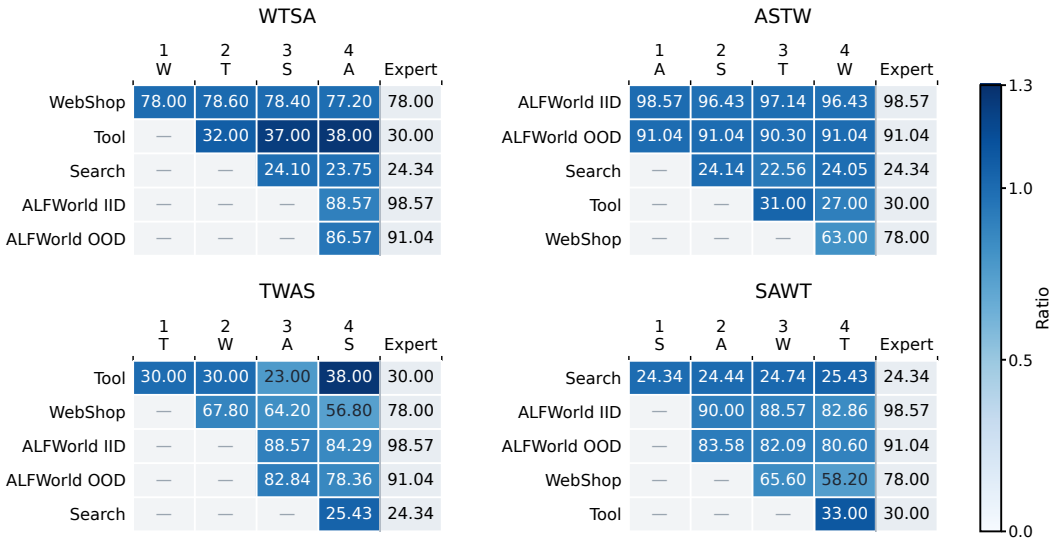

Figure 3: BACAM's stage-wise SR (%) under four arrival orders. W, T, S, and A denote WebShop, Tool, Search, and ALFWorld. Cell values are SR; shading shows SR relative to the corresponding expert. — marks tasks not yet integrated.

Acquisition and retention. In WTSA, adding Tool raises its SR from the initial WebShop expert's 4.00% to 32.00%, while WebShop SR changes from 78.00% to 78.60%. Subsequent integrations bring Tool SR to 38.00%, while final WebShop SR is 77.20%, only 0.80 points below its initial value. Search changes from 24.10% at integration to 23.75% after ALFWorld is added. ALFWorld enters last and reaches 88.57/86.57% IID/OOD SR. BACAM does not fully acquire the ALFWorld expert's capability at this integration stage.

Effect of merging order. Merging order has a substantial but task-dependent effect. Final Avg SR ranges from 56.02% in SAWT to 62.82% in WTSA, a difference of 6.80 points. Across the four orders, WebShop varies by 20.40 points, whereas Search varies by only 1.68 points. The initial expert's own-task capability is largely preserved: WebShop in WTSA ends 0.80 points below its expert, and ALFWorld in ASTW ends 2.14/0.00 points below its IID/OOD expert scores. By contrast, adding WebShop later leaves a 10.20–15.00-point gap at integration, and adding ALFWorld later leaves an 8.57–10.00-point IID gap; Tool and Search reach or approach their expert scores. Thus, order mainly affects the acquisition and subsequent retention of certain incoming capabilities rather than uniformly degrading all tasks.

Table 2: Four-expert ablation SR (%) under WTSA. WUDI-Merging and OPCM are reference methods.

Method	WebShop	Tool	Search	ALFWorld		Avg ↑
				IID	OOD	
Reference methods						
WUDI-Merging	38.20	0.00	25.14	72.14	70.15	41.13
OPCM	22.60	1.00	21.52	46.43	41.79	26.67
(1) Task-level behavioral constraints						
Forward KL for all tasks	76.80	30.00	23.30	89.29	85.07	60.89
Reverse KL for all tasks	77.00	36.00	24.05	87.86	85.82	62.14
Uniform token weighting	77.00	35.00	23.55	87.14	84.33	61.40
(2) Tensor-level plasticity budgets						
w/o conflict budget	77.40	34.00	22.81	87.86	85.82	61.58
BACAM (ours)	77.20	38.00	23.75	88.57	86.57	62.82

Parameter adaptation. Under TWAS, gate movement decreases across successive integrations (0.035, 0.025, and 0.013), indicating increasingly conservative updates along this sequence. MLP tensors account for 72–77% of total movement, while value projections have the largest average gate changes within attention. LayerNorm receives the tightest plasticity budgets. During ALFWorld integration, BACAM further protects WebShop more strongly than Tool in middle-layer value projections (Appendix Figure 4). These patterns show that adaptation depends on both stage and tensor type; further diagnostics appear in Appendix D.1.

4.4 Ablation on Each Component

We assess BACAM's two levels of control after the full WebShop–Tool–Search–ALFWorld sequence (Table 2). The variants retain behavioral supervision while changing the KL directions, token weighting, or tensor-level budgets throughout merging.

(1) **Task-level behavioral constraints.** Using forward KL for all tasks slightly improves ALFWorld IID SR from 88.57% to 89.29%, but lowers its OOD SR from 86.57% to 85.07% and reduces Tool SR from 38.00% to 30.00%. Reverse KL for all tasks slightly improves Search SR but lowers SR on the other tasks. Neither uniform KL direction matches BACAM's overall performance: BACAM achieves 62.82% Avg SR, compared with 60.89% for forward KL and 62.14% for reverse KL. Replacing token-group weighting with uniform token weighting reduces Avg SR to 61.40%, with the largest decreases on Tool and ALFWorld OOD. These comparisons support using different KL directions for acquisition and preservation, together with explicit weighting of behavior-critical tokens.

(2) **Tensor-level plasticity budgets.** Removing the conflict budget while retaining task-level constraints reduces Avg SR from 62.82% to 61.58%. The largest decrease is on Tool, from 38.00% to 34.00%; Search and ALFWorld SR also decline, while WebShop SR rises slightly from 77.20% to 77.40%. The overall improvement with the budget suggests that tensor-level conflict control complements the task-level objective, although its benefit is not uniform across tasks.

5 Conclusion

We studied continual agent merging in multi-turn interactive environments. Our method, BACAM, learns parameter-wise merging gates through task-level behavioral constraints and tensor-level conflict-aware plasticity budgets. Supervision on candidate-generated trajectories guides capability acquisition and preservation, while the budgets restrict updates in conflicting tensors. Experiments across four interactive tasks show that BACAM outperforms the evaluated merging baselines across different expert counts. In the default four-expert order, BACAM achieves 62.82% average success rate, exceeding the strongest baseline by 21.69 percentage points. These findings support the promise of behavior-guided merging for interactive agents, while identifying robustness to expert merging order as a key direction for future work.

AI USE STATEMENT

Generative AI tools assisted with improving the clarity and readability of the manuscript. Their suggestions helped refine wording, simplify complex sentences, and improve transitions between paragraphs. The tools also assisted in identifying potentially relevant literature for the background and related-work discussion. The authors selected the final wording and references and take responsibility for the content of the paper.

REPRODUCIBILITY STATEMENT

Section 3 specifies the merging objective, update rules, and algorithm. Section 4.1 describes the experts, benchmarks, metrics, and implementation settings. Appendix B gives the evaluation protocol and merging configurations; Appendix C reports additional results; and Appendix D provides gate and conflict diagnostics.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. ReSearch: Learning to reason with search for LLMs via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Xuefei Cao, Yuanhao Xiong, Haibo Tong, et al. Scaling agent learning via experience synthesis. In *International Conference on Learning Representations*, volume 2026, pp. 121394–121420, 2026.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2025.

Zhuoka Feng, Kang Chen, Sihan Zhao, Kai Xiong, Yaoning Wang, Minshen Yu, Junjie Nian, Changyi Xiao, Yixin Cao, and Yugang Jiang. ARM: Role-conditioned neuron transplantation for training-free generalist LLM agent merging. *arXiv preprint arXiv:2601.07309*, 2026.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705. IEEE, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, et al. Gem: A gym for agentic llms. *arXiv preprint arXiv:2510.01051*, 2025.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost Van De Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *arXiv preprint arXiv:2502.04959*, 2025.

Imad Eddine Marouf, Enzo Tartaglione, Stéphane Lathuilière, and Joost Van De Weijer. Ask and remember: A questions-only replay strategy for continual visual question answering. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18078–18089. IEEE, 2025.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2025.

Zihuan Qiu, Yi Xu, Chiyuan He, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Mingle: Mixture of null-space gated low-rank experts for test-time continual model merging. *Advances in Neural Information Processing Systems*, 38:143841–143878, 2025.

Zihuan Qiu, Lei Wang, Yang Cao, Runtong Zhang, Bing Su, Yi Xu, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Null-space filtering for data-free continual model merging: Preserving stability, promoting plasticity. In *International Conference on Learning Representations*, volume 2026, pp. 128492–128518, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Merging models on the fly without retraining: A sequential approach to scalable continual model merging. *arXiv preprint arXiv:2501.09522*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Junjian Wang, Xin Zhou, Qiran Xu, and Kun Zhan. M2A: Synergizing mathematical and agentic reasoning in large language models. *arXiv preprint arXiv:2605.09879*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, volume 2024, pp. 22743–22763, 2024b.

Enneng Yang, Anke Tang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, and Jie Zhang. Continual model merging without data: Dual projections for balancing stability and plasticity. *Advances in Neural Information Processing Systems*, 38:39275–39305, 2025.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are Super Mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 57755–57775, 2024.

Xiangchi Yuan, Dachuan Shi, Chunhui Zhang, Zheyuan Liu, Shenglong Yao, Soroush Vosoughi, and Wenke Lee. Behavior knowledge merge in reinforced agentic models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33007–33028, 2026.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024.

Dengming Zhang, Xiaowen Ma, Zhenliang Ni, Zhenkai Wu, Han Shu, Xin Jiang, and Xinghao Chen. Expert merging: Model merging with unsupervised expert alignment and importance-guided layer chunking. In *International Conference on Learning Representations*, volume 2026, pp. 21780–21803, 2026.

A Formal Setup and Optimization Details

A.1 Continual Merging Setup

We consider T ordered tasks whose experts share the same architecture and tokenizer. We initialize $M_1 = E_1$ and perform merging stages $t = 2, \ldots, T$. At stage t, task indices $\tau < t$ denote previously integrated tasks and $\tau = t$ denotes the incoming task. Each $\mathcal{D}_\tau$ contains unlabeled instances for task τ. BACAM has access to these data and their interaction environments, the current merged model M_{t-1}, and the incoming expert E_t. The goal is to construct a unified model M_t that acquires the capability of E_t while retaining the capabilities of M_{t-1}:

$$M_t = \mathrm{Merge}(M_{t-1}, E_t). \tag{13}$$

This objective involves two competing requirements. *Plasticity* requires the merged model to adapt its decision rule to the current task, whereas *stability* requires the update to preserve the decision rules that support previously integrated tasks. Because M_t serves as the current model at the next stage, degradation at any stage can accumulate along the merging sequence.

For interactive agents, neither parameter distance nor output discrepancies on isolated histories adequately characterize this objective. An action taken early in an interaction changes the subsequent environment feedback and, consequently, the histories visited later. Continual model merging must therefore constrain the behavioral changes induced by parameter updates, rather than parameter changes alone. We use policy-generated interaction trajectories as the behavioral representation that connects parameter-space optimization to task-level capabilities.

A.2 Policy-Induced Interaction Trajectories

Following the ReAct paradigm (Yao et al., 2022b), we represent each interaction step as an observation–reasoning–action triplet, as illustrated in Figure 2. At step i, the agent receives an observation o_i, produces a reasoning trace r_i, and executes an action a_i. The environment then returns the next observation:

$$r_i \sim \pi_\theta(\cdot \mid h_i), \qquad a_i \sim \pi_\theta(\cdot \mid h_i, r_i), \qquad o_{i+1} \sim \mathcal{E}^\tau(\cdot \mid h_i, a_i), \tag{14}$$

where $h_i = (o_0, r_0, a_0, \ldots, o_i)$ is the interaction history and $\mathcal{E}^\tau$ denotes the dynamics of task environment τ. A complete trajectory is therefore

$$\xi = (o_0, r_0, a_0, \ldots, o_{L-1}, r_{L-1}, a_{L-1}, o_L). \tag{15}$$

The policy and environment jointly induce the trajectory distribution

$$\mathbb{P}^\tau_{\pi_\theta}(\xi) = \rho^\tau_0(o_0) \prod_{i=0}^{L-1} \pi_\theta(r_i \mid h_i)\pi_\theta(a_i \mid h_i, r_i)\mathcal{E}^\tau(o_{i+1} \mid h_i, a_i), \tag{16}$$

where ρ^τ_0 is the initial-observation distribution. For an autoregressive language model, the reasoning and action at step i form a token sequence $z_i = (r_i, a_i) = (y_{i,1}, \ldots, y_{i,L_i})$, whose probability factorizes as

$$\pi_\theta(r_i, a_i \mid h_i) = \prod_{\ell=1}^{L_i} \pi_\theta(y_{i,\ell} \mid h_i, y_{i,<\ell}). \tag{17}$$

Agent behavior is thus characterized by the distribution over observation–reasoning–action trajectories, rather than by an isolated model output. An incorrect action changes the next observation and can redirect all subsequent reasoning and actions.

A.3 Gate and Dual Updates

At the beginning of each refresh round, the current candidate generates trajectories for the new and previously integrated tasks. These trajectories and their teacher distributions remain fixed during the gate updates in that round. Gradients are propagated through the candidate parameters but not through trajectory collection. The local conflict probe estimates the tensor budgets without updating the optimizer or dual variables, and the gate is restored before the actual updates.

For completeness, the projected dual update used for each old-task constraint is

$$\lambda_\tau \leftarrow \mathrm{clip}(\lambda_\tau + \eta_\lambda[S_\tau(g) - \epsilon], 0, \lambda_{\max})\,. \tag{18}$$

The parameterization in Equation 1 gives the gate gradient by the chain rule:

$$\nabla_g \mathcal{L} = \Delta\theta_t \odot \nabla_{\theta_t^{\mathrm{merged}}(g)} \mathcal{L}. \tag{19}$$

The normalized descent step before applying tensor budgets is

$$\Delta g = -\eta_g \frac{\nabla_g \mathcal{L}(g, \lambda)}{\|\nabla_g \mathcal{L}(g, \lambda)\|_2}. \tag{20}$$

B Experimental Protocol and Merging Configurations

B.1 Evaluation Protocol

All models are evaluated on the same fixed sets. WebShop uses 500 synthetic shopping goals over a 1,000-product catalog; an episode succeeds when its task score reaches 1.0. BFCL v3 uses 100 held-out `multi_turn_base` tasks and the official checker: every turn must pass its state and response checks. MuSiQue uses 2,017 held-out questions; Search success requires normalized substring matching of the final answer (Sub-EM). ALFWorld uses all 140 `valid_seen` games for IID and 134 `valid_unseen` games for OOD evaluation. Round is the mean number of interaction steps across all WebShop or ALFWorld episodes, including failures. Table 3 gives the generation settings for each benchmark.

Table 3: Evaluation settings. ALFWorld instance counts are shown as IID/OOD; – indicates that a setting does not apply or has no fixed limit.

Setting	WebShop	Tool	Search	ALFWorld
Evaluation instances	500	100	2,017	140/134
Temperature	0.4	0.001	0	0.4
Max. output tokens	512	4,096	8,192	512
Context length	8,192	32,768	8,192	8,192
Recent history steps	2	–	–	2
Max. environment steps	15	–	–	50
Max. tool steps per turn	–	20	–	–
Retrieved passages	–	–	5	–

B.2 Merging Configurations

BACAM optimization. Table 4 reports the settings shared by the four measured orders. These runs differ in expert arrival order. Training instances are unlabeled, and teacher token distributions provide the behavioral targets. The teacher representation retains the top 32 token probabilities and the aggregate probability of the remaining vocabulary, with distillation temperature 1.0.

Table 4: Optimization settings shared by the measured BACAM runs.

Setting	Value
Rollout-refresh rounds per stage	3
Gate updates per round	20 / 30 / 50
Training states per task stream per round	256
States per update per task stream	16
Probe trajectories per task stream	24
Gate learning rate / schedule	2×10^{-2} / cosine
Gate range	$[0, 1]$
Stability budget ϵ	5×10^{-3}
Dual update	$\eta_\lambda = \lambda_{\max} = 1$
Teacher top-k / temperature	32 / 1.0
Training hardware	8× RTX 5090 GPUs

Baseline configurations. Table 5 distinguishes task-instance access from the optimization signal in our implementations. RAM and RAM++ use only task-vector statistics, whereas WUDI-Merging and NUFILT optimize objectives derived from model parameters without task instances. AdaMerging and AdaMerging++ use first-turn prompts without multi-turn environment rollouts. Expert Merging and BACAM both use teacher outputs, but on expert-generated and candidate-generated trajectories, respectively. No ground-truth response labels are used for merging, and all calibration instances are separate from the held-out evaluation set. Data-collection and update schedules are method-specific, not budget-matched.

Table 5: Task-instance access and optimization signals during merging. Instance-dependent methods use unlabeled data; teacher targets are model-generated.

Method	Task instances	Optimization signal
RAM / RAM++	None	Shared/unique task-vector statistics
Other instance-free baselines	None	Parameter-derived rules or objectives; no sample-level targets
AdaMerging / AdaMerging++	Unlabeled first-turn prompts	Prediction entropy; no teacher targets
Expert Merging	Expert-generated trajectories (256 states/task)	Expert hidden states and output distributions
BACAM	Candidate-generated trajectories (256 states/task/round)	Token distributions from the incoming expert and current merged model

C Additional Results

C.1 Two- and Three-Expert Results

Table 6: SR (%) after merging two or three experts. Avg is computed within each expert group. Continual methods follow WebShop → Tool → Search; batch methods merge the same expert subset in each group. Blue shading marks BACAM. **Bold** and underlined values mark the highest and second-highest measured results per column; ties share the same mark.

Method	2 experts			3 experts			
	WebShop	Tool	Avg	WebShop	Tool	Search	Avg
General model merging							
Weight Average	26.60	1.00	13.80	7.00	0.00	22.21	9.74
Task Arithmetic	26.60	1.00	13.80	24.60	1.00	21.86	15.82
TIES-Merging	31.20	0.00	15.60	30.60	0.00	16.31	15.64
TSV-Merge	35.40	1.00	18.20	36.80	0.00	23.75	20.18
WUDI-Merging	34.40	0.00	17.20	33.60	0.00	**24.84**	19.48
Agent model merging							
RAM	34.60	0.00	17.30	33.00	0.00	21.27	18.09
RAM++	33.80	0.00	16.90	32.20	2.00	21.17	18.46
Continual model merging							
OPCM	33.20	1.00	17.10	22.00	1.00	20.62	14.54
NUFILT	9.80	0.00	4.90	8.20	0.00	20.53	9.58
BACAM (ours)	**78.60**	**32.00**	**55.30**	**78.40**	**37.00**	24.10	**46.50**

C.2 Task Scores and Interaction Lengths

Table 7 complements the SR comparison with every reported TS and Round measurement. TS retains information about partial shopping-task completion, whereas SR requires complete success. Round includes successful and failed episodes, so comparisons should consider both task performance and interaction length. NUFILT results for WTSA and the earlier TSWA run are listed separately.

Table 7: Task score and interaction length for all evaluated models. WebShop TS is on a 0–100 scale. Round averages interaction steps over all episodes, including failures. SR results appear in Table 1.

Method	ALFWorld Round ↓		WebShop	
	IID	OOD	TS ↑	Round ↓
Base model and individual experts				
Qwen2.5-7B-Instruct	44.34	46.19	1.79	14.78
Search expert	48.00	47.10	4.45	14.49
Tool expert	47.32	47.19	1.96	14.85
ALFWorld expert	8.43	12.52	25.16	11.46
WebShop expert	42.02	42.78	88.41	6.11
General model merging				
Weight Average	37.61	37.45	26.87	12.05
Task Arithmetic	24.19	28.20	56.66	9.79
TIES-Merging	28.64	29.00	62.00	9.52
TA w/ DARE	24.05	30.68	56.41	9.82
TIES w/ DARE	27.11	28.84	59.76	9.54
TSV-Merge	22.90	24.67	65.18	9.01
Iso-C	27.17	29.38	52.86	9.94
WUDI-Merging	21.46	24.31	54.24	10.01
AdaMerging	26.71	28.36	49.50	9.74
AdaMerging++	26.20	26.20	53.15	9.98
Expert Merging	38.73	39.84	29.00	11.89
Agent model merging				
RAM	29.09	30.15	61.54	9.06
RAM++	27.48	31.49	61.09	9.31
Continual model merging				
OPCM	32.69	35.47	44.76	10.76
NUFILT	37.49	39.31	32.73	11.79
NUFILT (TSWA reference)	37.50	36.94	28.83	11.95
BACAM (ours)	14.01	14.69	88.16	6.09

C.3 Stage-wise Scores and Capability Acquisition

Table 8 reports the scores underlying Figure 3, together with WebShop TS.

Table 8: BACAM after each integration for the measured orders. All task columns report SR (%) except the final WebShop TS column. – marks a task whose expert has not yet been integrated.

Order	Stage	WebShop	Tool	Search	ALFWorld		WebShop
		SR	SR	SR	IID SR	OOD SR	TS
WTSA	1: W	78.00	–	–	–	–	88.41
	2: + T	78.60	32.00	–	–	–	88.58
	3: + S	78.40	37.00	24.10	–	–	88.32
	4: + A	77.20	38.00	23.75	88.57	86.57	88.16
ASTW	1: A	–	–	–	98.57	91.04	–
	2: + S	–	–	24.14	96.43	91.04	–
	3: + T	–	31.00	22.56	97.14	90.30	–
	4: + W	63.00	27.00	24.05	96.43	91.04	77.51
TWAS	1: T	–	30.00	–	–	–	–
	2: + W	67.80	30.00	–	–	–	80.95
	3: + A	64.20	23.00	–	88.57	82.84	79.09
	4: + S	56.80	38.00	25.43	84.29	78.36	71.39
SAWT	1: S	–	–	24.34	–	–	–
	2: + A	–	–	24.44	90.00	83.58	–
	3: + W	65.60	–	24.74	88.57	82.09	79.09
	4: + T	58.20	33.00	25.43	82.86	80.60	73.69

D GATE AND CONFLICT DIAGNOSTICS

D.1 TWAS ORDER

Under TWAS, gate movement declines across successive integrations, but its allocation across tensors remains structured. MLP tensors account for most movement, while attention gains share as the sequence progresses. Plasticity budgets vary more across capabilities and stages than gate movement does, indicating that the controller adapts protection to the incoming capability.

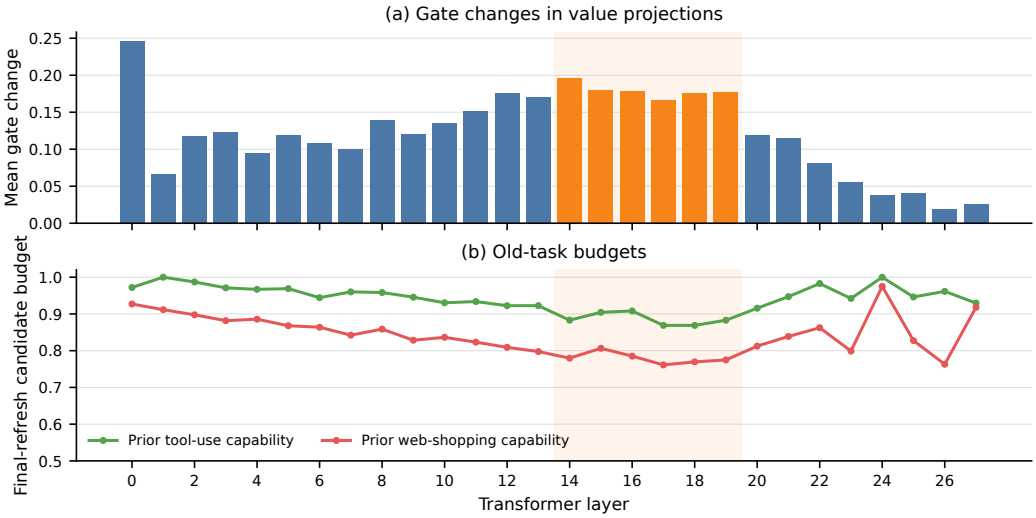

Figure 4: Value-projection tensors during ALFWorld integration in TWAS. (a) Mean gate change by layer, with layers 14–19 highlighted. (b) Final-refresh candidate budgets for Tool and WebShop; lower budgets indicate stronger protection. WebShop is more tightly protected across the highlighted layers.

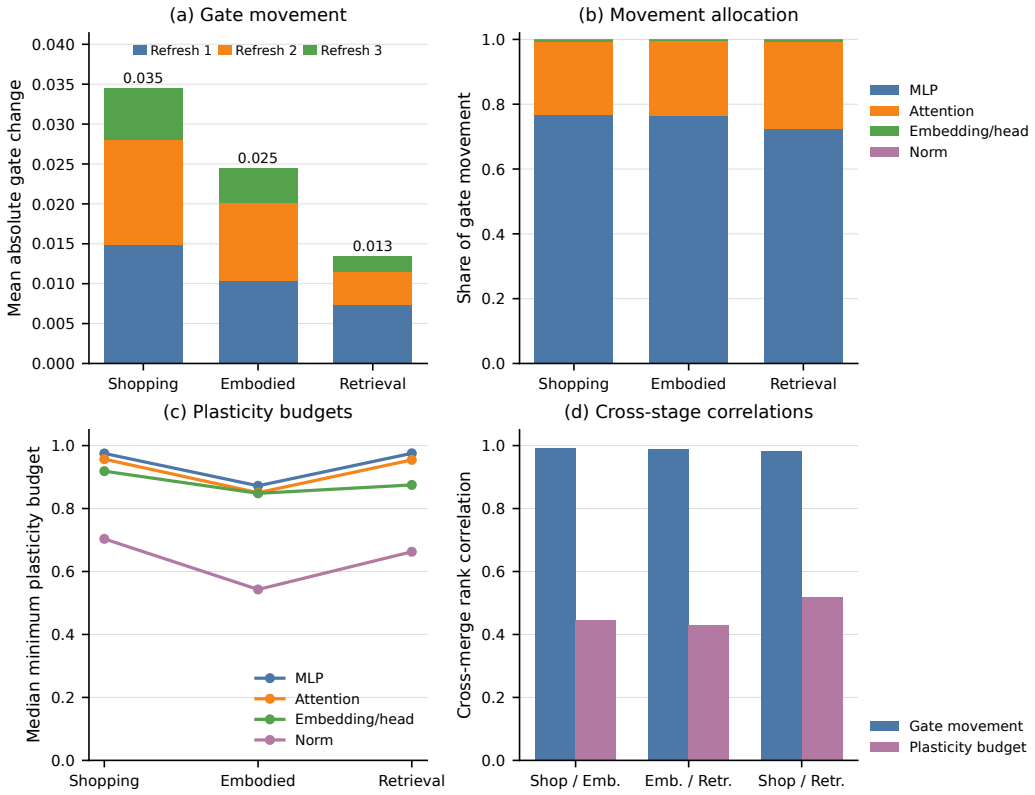

Figure 5: Gate movement and plasticity budgets across TWAS integrations, ordered by incoming WebShop, ALFWorld, and Search experts. (a) Per-refresh mean absolute gate change, which declines across stages; (b) movement share by tensor family, dominated by MLP; (c) median minimum budget by family, where lower values mean stronger protection; and (d) cross-stage Spearman rank correlations, showing more persistent movement than budgets.

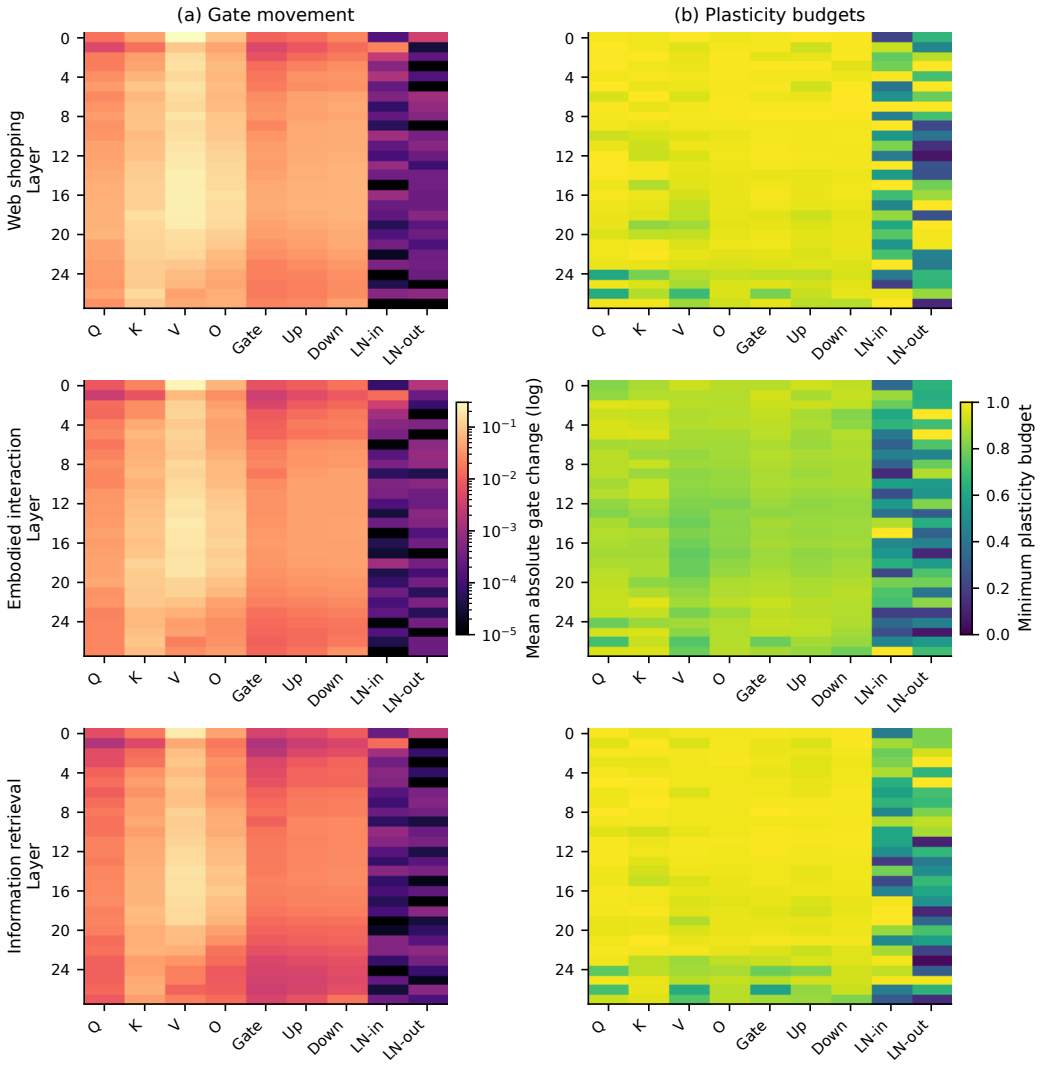

Figure 6: Layer-wise diagnostics across TWAS integrations. Rows follow incoming WebShop, ALFWorld, and Search experts; columns show (a) mean absolute gate movement and (b) minimum plasticity budget by layer and weight type. Lower budgets indicate stronger protection, concentrated in selected tensors.

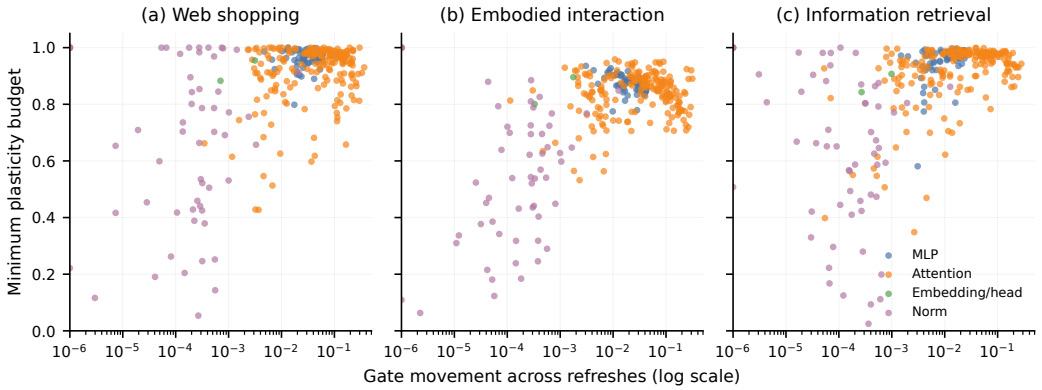

Figure 7: Tensor-level gate movement versus minimum plasticity budget across TWAS integrations. Each point is one tensor; (a)–(c) correspond to incoming WebShop, ALFWorld, and Search experts. Lower budgets indicate stronger protection.

D.2 SAWT Order

The same tensor-level diagnostics are reported for the Search–ALFWorld–WebShop–Tool order.

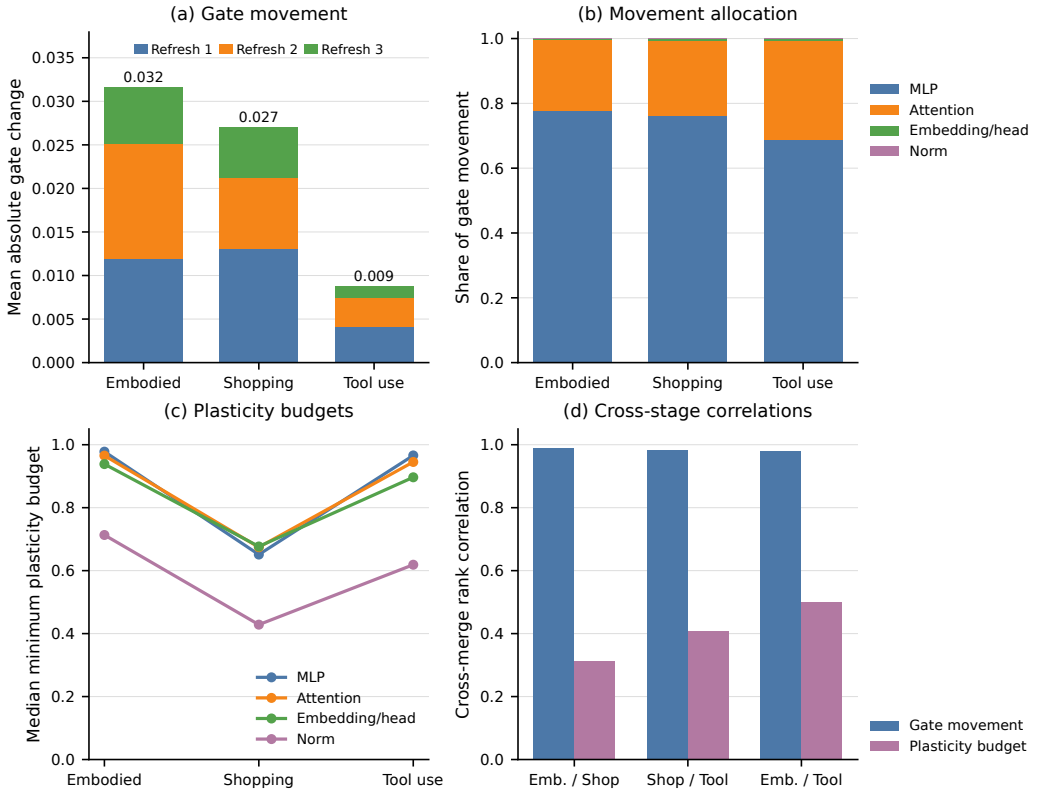

Figure 8: Gate movement and plasticity budgets across SAWT integrations, ordered by incoming ALFWorld, WebShop, and Tool experts. (a) Per-refresh mean absolute gate change, which declines across stages; (b) movement share by tensor family, dominated by MLP; (c) median minimum budget by family, where lower values mean stronger protection; and (d) cross-stage Spearman rank correlations, showing more persistent movement than budgets.

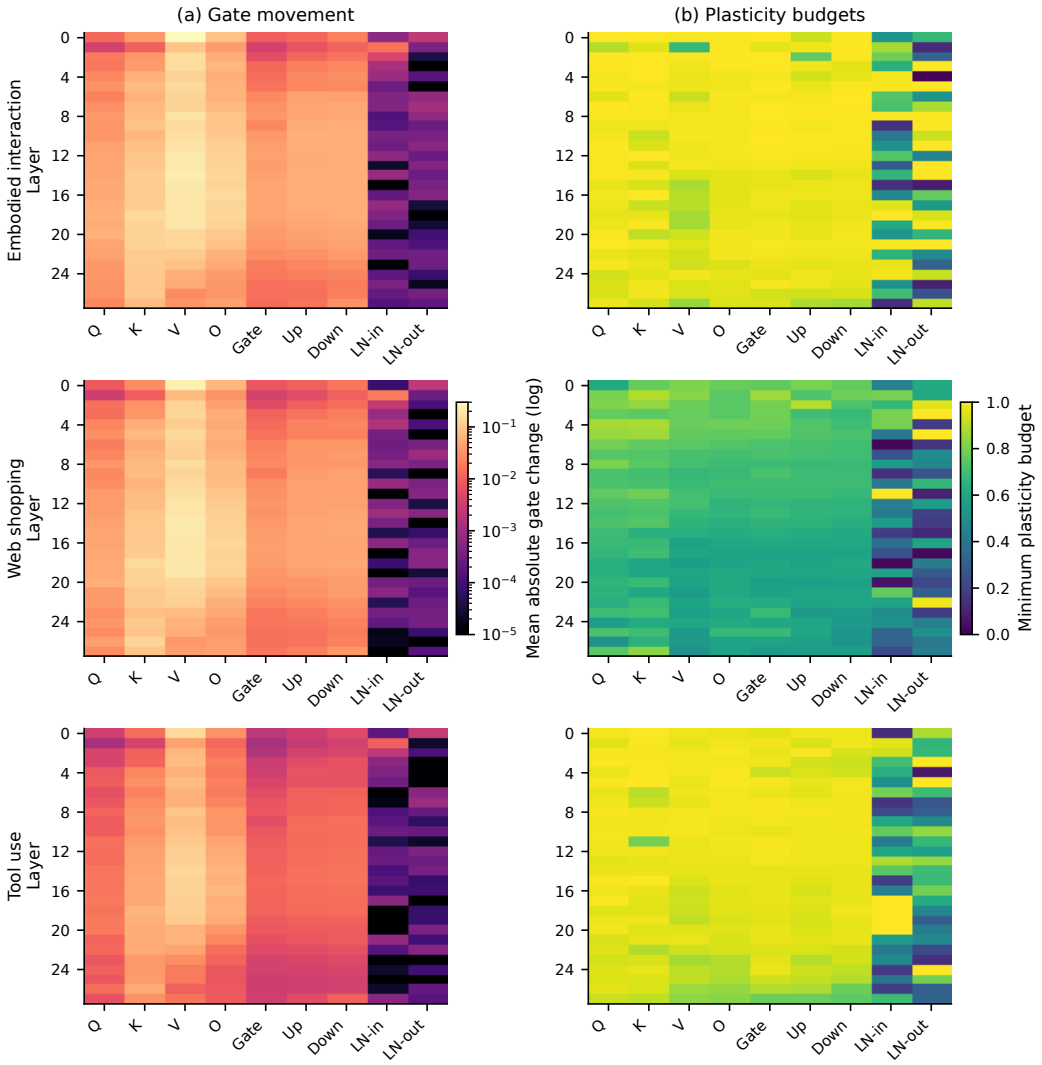

Figure 9: Layer-wise diagnostics across SAWT integrations. Rows follow incoming ALFWorld, WebShop, and Tool experts; columns show (a) mean absolute gate movement and (b) minimum plasticity budget by layer and weight type. Lower budgets indicate stronger protection.

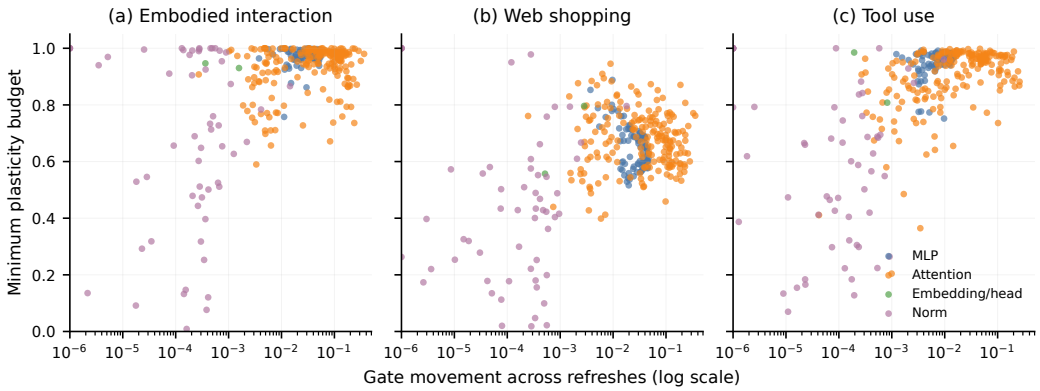

Figure 10: Tensor-level gate movement versus minimum plasticity budget across SAWT integrations. Each point is one tensor; (a)–(c) correspond to incoming ALFWorld, WebShop, and Tool experts. Lower budgets indicate stronger protection.

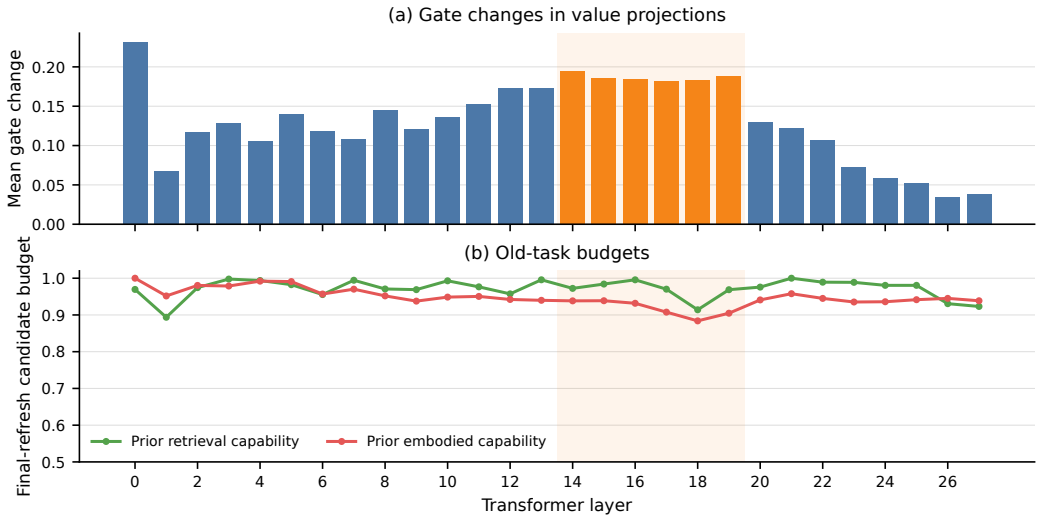

Figure 11: Value-projection tensors during WebShop integration in SAWT. (a) Mean gate change by layer, with layers 14–19 highlighted. (b) Final-refresh candidate budgets for Search and ALFWorld; lower budgets indicate stronger protection. ALFWorld is more tightly protected across the highlighted layers.

E QUALITATIVE EXAMPLES

The examples compare BACAM with WUDI-Merging on matched evaluation instances. Each example summarizes the decisive observation–reasoning–action step; reasoning is paraphrased from the recorded trajectory rather than quoted verbatim.

WebShop: tracking a remaining product constraint

Observation: The requested curtain is open. The color "sea mist" has been selected, but the required 52-by-84-inch size remains to be selected.

BACAM: Correct

Reasoning: The size constraint is still unmet.
Action: Select 52-by-84-inch size, then purchase.
Outcome: Success; task score 1.00.

WUDI-Merging: Incorrect

Reasoning: Select the requested color.
Action: Select "sea mist" again, then purchase.
Outcome: Size omitted; task score 0.80.

Tool use: selecting the function that matches the request

Observation: The user asks for the latest trading details of Quasar Ltd.

BACAM: Correct

Reasoning: Resolve the company's ticker, then retrieve its stock information.
Action: `get_symbol_by_name`, then `get_stock_info`.
Outcome: Correct under the BFCL evaluator.

WUDI-Merging: Incorrect

Reasoning: Treat the request as asking for past transactions.
Action: Call `get_transaction_history`.
Outcome: Tool response does not match the request.

Search: resolving a multi-hop clue before answering

Observation: The question asks for the largest national park in the country identified by a hunting-versus-ecotourism spending clue.

BACAM: Correct

Reasoning: Resolve the country clue before searching for its largest park.
Action: Search for Tanzania's largest national park.
Outcome: Ruaha National Park (correct).

WUDI-Merging: Incorrect

Reasoning: Assume the country is the United States.
Action: Search for the largest US national park.
Outcome: Yellowstone National Park (incorrect).

ALFWorld: responding to the latest environment state

Observation: The agent has picked up the bowl from the desk and must examine it with a desk lamp.

BACAM: Correct

Reasoning: The lamp is not at the desk; check another location.
Action: Go to the dresser, then use the lamp.
Outcome: Success in 4 steps.

WUDI-Merging: Incorrect

Reasoning: Look for the lamp after picking up the bowl.
Action: Return to the desk, then repeat invalid lamp commands.
Outcome: No success within 50 steps.

Across these matched examples, BACAM follows the remaining task constraints and environment feedback at the decisive step, whereas WUDI-Merging diverges before completion.