ARM: Role-Conditioned Neuron Transplantation for Training-Free Generalist LLM Agent Merging

Anonymous ACL submission

Abstract

Interactive large language model agents have advanced rapidly, but most remain specialized to a single environment and fail to adapt robustly to other environments. Model merging offers a training-free alternative by integrating multiple experts into a single model. In this paper, we propose Agent-Role Merging (ARM), an activation-guided, role-conditioned neuron transplantation method for model merging in LLM agents. ARM improves existing merging methods from static natural language tasks to multi-turn agent scenarios, and over the generalization ability across various interactive environments. This is achieved with a well designed 3-step framework: 1) constructing merged backbones, 2) selection based on it role-conditioned activation analysis, and 3) neuron transplantation for fine-grained refinements. Without gradient-based optimization, ARM improves cross-benchmark generalization while enjoys efficiency. Across diverse domains, the model obtained via ARM merging outperforms prior model merging methods and domain-specific expert models, while demonstrating strong out-of-domain generalization.

1 Introduction

Recently, we have witnessed the surge of agents by fine-tuning large language models (LLMs) in interactive environments, such as web browsing and operating systems (Liu et al., 2023; Zheng et al., 2025). These LLM-based agents can think, plan, and act through external tools to accomplish real-world tasks, making them practically valuable (Yao et al., 2023; Qin et al., 2024). Despite these advances, current LLM-based agents often exhibit limited cross-environment robustness (Yao et al., 2024; Wang et al., 2024). Models tuned for one environment often degrade sharply when deployed in another one with different tool schemas, action interfaces, or trajectory distributions (Yao et al., 2024; Wang et al., 2024). A straightforward solution is to further fine-tune a single model across all environments, but this introduces substantial engineering and optimization complexity (e.g., curriculum/order effects across environments, heterogeneous tool interfaces, and extensive debugging) and incurs huge training costs (Zhang et al., 2025; Li et al., 2025a).

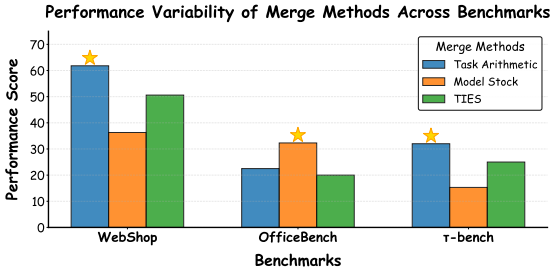

Figure 1: Performance variability of common training-free merge heuristics across interactive agent benchmarks.

In this paper, we focus on a training-free alternative, model merging. It aims at combining multiple checkpoints of the same architecture — often specialists fine-tuned for different environments (marked as expert in the rest of paper) — into a single model that aims to inherit the strengths of each (Ilharco et al., 2023; Yadav et al., 2023). Without additional training, model merging offers a practical path to greatly improving capabilities and reducing the burden of maintaining many specialized checkpoints (Wortsman et al., 2022; Ilharco et al., 2023). A growing literature studies training-free merging, from simple parameter-space compositions (Ilharco et al., 2023) to interference-aware recipes such as TIES-Merging (Yadav et al., 2023). Recent work further investigates activation merging (e.g., AIM (Nobari et al., 2025), NeuronMerge (Gu et al., 2025)), leveraging internal signal tracing to mitigate inter-model interference. However, the above methods are predominantly developed on static, single-turn tasks, and few works target interactive agent settings.

To this end, we propose agentic merging to com-

bine multiple LLMs that generalize well across interactive environments. We highlight two critical challenges. First, how can we preserve general capabilities reliably? Different base model families exhibit different internal mechanism and activation features. Their behaviors can become highly unstable across benchmarks. As shown in Figure 1, widely used heuristics exhibit pronounced cross-benchmark variance, with no single heuristic consistently strong across environments. This motivates a stability-aware backbone selection strategy prior to any fine-grained intervention. Second, how can we avoid capability conflicts during merging? This is a core challenge for model merging, and multi-turn agent trajectories exacerbate it. Small deviations in role-critical spans (e.g., tool-call formatting, action serialization, or final-answer JSON) can cascade into repeated failures and negative transfer across environments.

Therefore, we design Agent-Role Merging (ARM), an activation-guided, role-conditioned neuron transplantation framework for training-free model merging. To tackle the first challenge, ARM introduces a dynamic backbone construction and selection stage. It constructs a small candidate pool of merged backbones using standard weight-space merge operators, then dynamically selects a strong one using a well-designed strategy based on mechanism analysis. Note that this step remains training-free and avoids costly operations while maximizing the reserved capabilities. To tackle the second challenge, ARM performs fine-grained neuron transplantation at the level of role-critical behaviors. Specifically, ARM conducts role-conditioned activation tracing on a small calibration set to identify key neurons for specific abilities (e.g., tool calls, actions, and final-answer JSON), and then selectively transplants these neurons from the corresponding expert into the chosen backbone. We also use a conflict-aware policy to reduce negative transfer in multi-turn settings.

We evaluate ARM on multiple widely used agentic benchmarks, and show that it yields the strongest single merged generalist across both Qwen3 and Qwen2.5 expert pools, improving average performance and worst-suite robustness while maintaining strong out-of-domain generalization compared to prior training-free merging baselines.

Our contributions are summarized as follows:

- We propose to curate and select merged backbones dynamically for reliable general capability reservation.
- We propose a fine-grained neuron transplantation mechanism for agentic LLM merging towards better generalization.
- Extensive experiments on four in-domain suites and two out-of-domain benchmarks demonstrate that ARM consistently improves generalist performance and robustness over strong weight-space and activation-aware training-free baselines using a single merged checkpoint.

2 Related Work

Training generalist agents. One route to cross-environment generalization is to train a single generalist agent using large-scale multi-task trajectories, often via online interaction and reinforcement learning. AgentRL (Zhang et al., 2025) explores scaling agentic RL in multi-turn settings, and Chain-of-Agents (Li et al., 2025a) studies multi-agent distillation and agentic RL for foundation agents. While effective, such pipelines require expensive interaction and task coverage; our goal is complementary: training-free composition of existing specialists.

Model merging beyond static-task heuristics. Model merging combines multiple fine-tuned models into a single model without additional gradient updates, ranging from weight averaging and model soups (Wortsman et al., 2022), task arithmetic and task vectors (Ilharco et al., 2023) to interference-aware recipes such as TIES-Merging (Yadav et al., 2023). Recent work also explores activation-aware merging, such as AIM (Nobari et al., 2025) and NeuronMerge (Gu et al., 2025), to mitigate interference by tracing internal signals. However, most existing approaches are developed and validated on static, single-turn NLP tasks. They lack activation-based criteria for selecting a strong merged backbone and do not incorporate conflict-aware policies to protect role-critical circuits when importing benchmark-specific behaviors. Our framework complements these efforts by using role-conditioned activation tracing for backbone selection and conflict-aware, role-salient neuron transplantation for mitigating negative transfer in interactive agents.

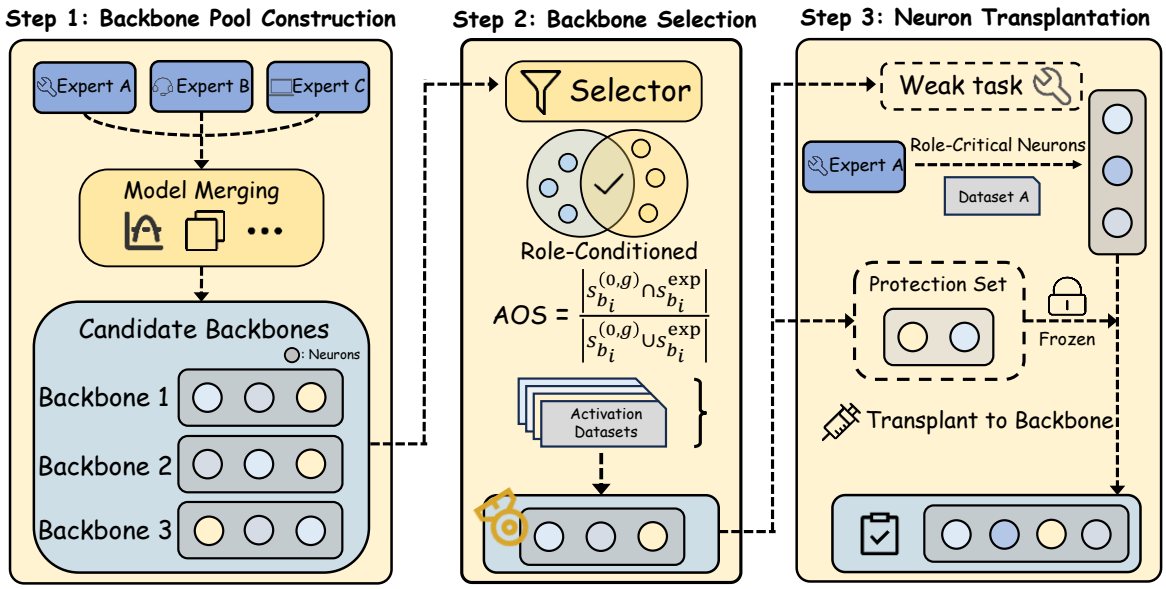

Figure 2: **Overview of Agent-Role Merging (ARM). Step 1: Backbone pool construction.** We apply multiple training-free weight-space merge operators to benchmark-specialized experts to obtain a pool of candidate merged backbones. **Step 2: Backbone selection.** A selector computes the *Activation-Overlap Score (AOS)* using role-conditioned MLP activations on a lightweight calibration set, and chooses the candidate backbone that maximizes mean AOS across benchmarks. **Step 3: Neuron transplantation.** For benchmarks where the selected backbone remains weak, we transplant a small top-$k\%$ subset of donor (expert) MLP neurons into the backbone while strictly protecting neurons salient for other benchmarks to avoid negative transfer. The resulting single model consolidates expert capabilities across benchmarks without end-to-end retraining.

3 Method

In this section, we present Agent-Role Merging (ARM), a training-free pipeline for consolidating benchmark-specialized experts into a single multi-benchmark agent model. ARM proceeds in three phases: (i) **Backbone Pool Construction**, which constructs a pool of merged backbones via training-free weight-space merging, (ii) **Backbone Selection** selects the backbone that best preserves expert role-salient neurons using Activation-Overlap Score (AOS), and (iii) **Neuron Transplantation** repairs remaining capability gaps via conflict-aware neuron transplantation while strictly protecting neurons that are important for any other benchmark.

3.1 Backbone Pool Construction

To compare different merging strategies and select the best backbone, we first need to construct a set of candidate backbones, where each backbone represents a single merged model. Thus, we consider N benchmark-specialized experts $\{M_{b_i}^{\text{exp}}\}_{i=1}^{N}$ fine-tuned from the same base LLM, and thus sharing the same architecture and tokenizer. $\mathcal{B}$ denotes the set of benchmarks, and $M_{b_i}^{\text{exp}}$ denote the expert corresponding to the i-th benchmark $b_i \in \mathcal{B}$ with a one-to-one mapping between benchmarks and experts. Our goal is to obtain a single merged model that performs well across all benchmarks in $\mathcal{B}$ without additional gradient-based retraining. Since different training-free weight-space merge operators can yield markedly different trade-offs, we construct a small pool of candidate merged backbones by applying a set of standard merging operators $\mathcal{G}$, such as uniform averaging (Wortsman et al., 2022), task arithmetic (Ilharco et al., 2023), and TIES-Merging (Yadav et al., 2023). Each operator $g \in \mathcal{G}$ produces a candidate backbone:

$$M^{(0,g)} = g\left(\{M_{b_i}^{\text{exp}}\}_{i=1}^{N}\right). \quad (1)$$

3.2 Backbone Selection

To select the backbone that best preserves expert role-salient neurons for model merging, we first need to compare candidate merged backbones, which requires a criterion that identifies the parameters supporting benchmark-critical behaviors. To achieve this, we propose role-conditioned MLP activations, which serve as a lightweight, training-free criterion, to approximate these circuits and summarize them as top-k neuron sets.

Specifically, a small calibration set $\mathcal{D}_{\text{cal}}$ is sampled from splits that are disjoint from evaluation/test sets (see Section 4). $\mathcal{D}_{\text{cal}}^{(b_i)} \subseteq \mathcal{D}_{\text{cal}}$ denote the calibration trajectories for benchmark b_i. For a given model M with L blocks, $\mathbf{z}_\ell(t) \in \mathbb{R}^{d_{\text{ff}}}$

denotes the post-activation vector of the MLP in block ℓ at token position t. $T_{b_i,r}(x)$ denotes the role-r token positions in trajectory x for benchmark b_i. Trajectories without a given role segment (i.e., $T_{b_i,r}(x) = \emptyset$) are ignored when estimating saliency for (b_i, r). Therefore, Role-conditioned saliency is defined as the expected per-role mean activation:

$$s_{\ell,j}(M; b_i, r) = \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}^{(b_i)}} \left[\underset{t \in T_{b_i,r}(x)}{\text{mean}} \left| z_{\ell,j}(t) \right| \right]. \quad (2)$$

Next, the top-k fraction of neurons is selected per layer:

$$S_\ell(M; b_i, r) = \textsc{TopK}_j(s_{\ell,j}(M; b_i, r),\ \lceil k\, d_{\text{ff}} \rceil), \quad (3)$$

where $S(M; b_i, r) = \{(\ell, j) : j \in S_\ell(M; b_i, r)\}$ is the role-salient set. An element $n = (\ell, j) \in S(M; b_i, r)$ is referred to as a neuron index. For brevity, $S(M; b_i) \equiv S(M; b_i, r_{b_i})$ is used when the target role is clear from the benchmark.

Hereafter, we design an *Activation-Overlap Score (AOS)* to select a backbone that preserves role-salient neurons from the corresponding experts across benchmarks. Specifically, for each benchmark b_i, we denote the role-salient neuron sets of the expert model and the candidate backbone as $S_{b_i}^{\text{exp}}$ and $S_{b_i}^{(0,g)}$, respectively.

Based on these definitions, the Activation-Overlap Score of a candidate backbone on benchmark b_i is defined as:

$$\text{AOS}(M^{(0,g)}; b_i) = \frac{\left| S_{b_i}^{(0,g)} \cap S_{b_i}^{\text{exp}} \right|}{\left| S_{b_i}^{(0,g)} \cup S_{b_i}^{\text{exp}} \right|}. \quad (4)$$

Finally, we select the backbone with the highest mean AOS:

$$g^\star = \arg\max_{g \in \mathcal{G}} \frac{1}{|\mathcal{B}|} \sum_{b_i \in \mathcal{B}} \text{AOS}(M^{(0,g)}; b_i), \quad (5)$$
$$M^{(0)} = M^{(0,g^\star)}.$$

Using the AOS, the selector favors backbones that best preserve experts' role-salient neurons, yielding a robust starting point without exhaustive evaluation of every merge candidate.

3.3 Neuron Transplantation

To repair remaining capability gaps and protect neurons that are important for any other benchmarks, we propose conflict-aware neuron transplantation.

Specifically, we first conduct capability-gap diagnosis to identify weak benchmarks $\mathcal{B}_{\text{weak}} \subseteq \mathcal{B}$ introduced by role-specific regressions in model merging. A held-out development set $\mathcal{D}_{\text{dev}}$ (e.g., benchmarks with the largest performance gaps between $M^{(0)}$ and the corresponding expert) is selected to apply transplantation for $b_i \in \mathcal{B}_{\text{weak}}$. This development evaluation is used only to decide where transplantation is applied (not to train parameters). For each benchmark b_i, we use its corresponding expert as the donor and denote it by $M_{b_i}^{\text{don}} \equiv M_{b_i}^{\text{exp}}$.

Next, considering that weight-space merging is a global operation and can blur or overwrite benchmark-specific circuits. To correct specific failures without retraining the entire model, we perform localized edits by transplanting a small number of donor MLP neurons into the selected backbone. To achieve this, we refine $M^{(0)}$ by selectively transplanting a small subset of MLP neurons from donors into the backbone. For block ℓ, the MLP parameters are denoted as $W_{\text{in}}^\ell \in \mathbb{R}^{d_{\text{ff}} \times d}$, in which $b_{\text{in}}^\ell \in \mathbb{R}^{d_{\text{ff}}}$, and $W_{\text{out}}^\ell \in \mathbb{R}^{d \times d_{\text{ff}}}$. Neuron (ℓ, j) corresponds to row j of W_{in}^ℓ, entry j of b_{in}^ℓ, and column j of W_{out}^ℓ. A hard transplantation from donor M^{don} into model M performs:

$$\begin{aligned} W_{\text{in}}^\ell[j,:] &\leftarrow W_{\text{in}}^{\ell,\text{don}}[j,:], \\ b_{\text{in}}^\ell[j] &\leftarrow b_{\text{in}}^{\ell,\text{don}}[j], \\ W_{\text{out}}^\ell[:,j] &\leftarrow W_{\text{out}}^{\ell,\text{don}}[:,j], \end{aligned} \quad (6)$$

For gated MLPs (e.g., SwiGLU), a neuron index j corresponds to the same index across the gate and up projections as well as the down projection; we transplant the corresponding rows and columns accordingly. We apply transplantation only to a small set of neurons, keeping the rest of the network unchanged to reduce collateral interference, similar in spirit to localized editing methods that aim to confine behavioral changes (Meng et al., 2022a,b).

Finally, since naively transplanting all donor-salient neurons can overwrite neurons that the backbone already relies on for other benchmarks, causing negative transfer, we employ conflict-aware transplantation policy to *strictly* protect backbone neurons that are salient for any *other* benchmark, and only transplant donor neurons that do not belong to those protected sets.

For each benchmark b_i, we define the set of backbone neurons that are salient for any other benchmark's critical role:

$$\mathcal{P}_{-b_i} = \bigcup_{b_i' \in \mathcal{B},\ b_i' \neq b_i} S(M^{(0)}; b_i'), \quad (7)$$

Algorithm 1 ARM: Candidate merging, backbone selection, and conflict-aware neuron transplantation

Require: Expert models $\{M^{(i)}\}_{i=1}^{N}$; merge operators $\mathcal{G}$; benchmarks $\mathcal{B}$;
Ensure: Merged model $M^\star$.

```
Set donor M_b^don ← M_b^exp for each b ∈ B.
Compute donor saliency S_b^don ← S(M_b^don; b) for each b ∈ B.
Construct candidate backbones {M^(0,g)}_{g∈G}.
for g ∈ G do
    Compute backbone saliency S_b^(0,g) ← S(M^(0,g); b) for each b ∈ B.
    Score(g) ← (1/|B|) Σ_{b∈B} AOS(M^(0,g); b).
end for
g* ← arg max_{g∈G} Score(g).
M ← M^(0,g*).
Assign backbone saliency S(M; b) ← S_b^(0,g*) for each b ∈ B.
Select weak benchmarks B_weak ⊆ B on D_dev.
for b ∈ B_weak do
    P_{-b} ← ⋃_{b'∈B, b'≠b} S(M; b').
    T_b ← {n ∈ S_b^don | n ∉ P_{-b}}.
    Transplant neurons in T_b from M_b^don into M.
end for
return M.
```

where we use the shorthand $S(M; b_i) \equiv S(M; b_i, r_{b_i})$. When repairing benchmark b, we start from the donor's role-salient neurons $S(M_b^{\text{don}}; b)$ and exclude all neurons that are salient for any other benchmark:

$$\mathcal{T}_b = \left\{ n \in S(M_b^{\text{don}}; b) \;\middle|\; n \notin \mathcal{P}_{-b} \right\}. \tag{8}$$

With this conflict-aware transplantation strategy, we target capability gaps while protecting the stability of other benchmarks and minimizing negative transfer.

4 Experiments

4.1 Experiment Setup

Expert models. We adopt Qwen3-8B (Yang et al., 2025) as the primary backbone architecture and merge three benchmark-specialized SFT experts released in the SIMIA framework (Li et al., 2025b), namely Simia-Tau-SFT-Qwen3-8B, Simia-OfficeBench-SFT-Qwen3-8B, and Simia-AgentBench-SFT-Qwen3-8B. Each expert is fine-tuned on synthesized multi-turn trajectories with benchmark-specific tool interactions, including airline and retail tool calls in a τ^2-Bench-style environment (Barres et al., 2025), OfficeBench multi-application workflows, and AgentBench-style operating system and WebShop tasks. For additional comparisons, we also evaluate an expert pool based on Qwen2.5-7B trained with the same SIMIA recipe.

Baselines. We compare ARM with strong training-free weight-space merging baselines, including uniform averaging, Model Stock (Jang et al., 2024), task arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023), and TIES+DARE (Yu et al., 2024b) (all implemented in MERGEKIT (Goddard et al., 2024)), as well as WIDEN (Yu et al., 2024a), AIM (Nobari et al., 2025), and NeuronMerge (Gu et al., 2025). For detailed baseline settings, see Appendix A.1.

Benchmarks. To evaluate the generalization capability of our method, we conduct experiments under both in-domain and out-of-domain settings: 1) **In-domain benchmarks** include τ-bench (Yao et al., 2024), OfficeBench (Wang et al., 2024), WebShop, and Operating System (Both from AgentBench (Liu et al., 2023))) **Out-of-domain benchmarks** include DB-bench (Zheng et al., 2025) and AlfWorld (Shridhar et al., 2020). Further details on benchmark composition and settings are provided in Appendix A.2.

Calibration data and role spans. To compute role-conditioned saliency (Section 3.2), we construct a calibration set that is disjoint from all evaluation and test splits, containing a total of 699 tasks and 1240 trajectories. This calibration set is used solely for forward-pass activation tracing without gradient updates. Details of its composition are provided in Appendix A.3.

4.2 Main Results

Tables 1 and 2 summarize the overall results on the Qwen3-8B and Qwen2.5-7B expert pools, from which we draw the following observations:

(1) **ARM yields the strongest single merged generalist across both expert pools.** It is the only approach that consistently surpasses the BEST-of-Three oracle on both backbones. We attribute this to ARM's two-stage design: AOS-based backbone selection avoids starting from an unstable merge, and the subsequent localized repair targets only the remaining suite-specific deficiencies.

Model	WebShop	OS	τ-bench	OfficeBench	DB	AlfWorld	AVG
Qwen3-8B	30.5	13.2	32.0	2.9	41.3	24.0	24.0
Simia-Tau	44.8	16.0	43.8	2.9	45.7	32.0	30.9
Simia-OfficeBench	51.1	25.7	16.1	37.5	43.7	44.0	36.4
Simia-AgentBench	64.8	29.2	15.9	3.9	45.7	42.0	33.6
BEST-of-Three (oracle)	64.8	29.2	43.8	37.5	45.7	44.0	44.2
Average	**63.8** (-1.5%)	27.1 (-7.2%)	19.1 (-56.4%)	**49.8** (+32.8%)	44.7 (-2.2%)	46.0 (+4.5%)	41.8 (-5.4%)
Model Stock	36.3 (-44.0%)	15.3 (-47.6%)	32.3 (-26.3%)	3.9 (-89.6%)	43.0 (-5.9%)	22.0 (-50.0%)	25.5 (-42.3%)
Task Arithmetic	61.8 (-4.6%)	27.8 (-4.8%)	22.5 (-48.6%)	15.5 (-58.7%)	38.3 (-16.2%)	10.0 (-77.3%)	29.3 (-33.7%)
TIES	50.6 (-21.9%)	25.0 (-14.4%)	20.0 (-54.3%)	43.3 (+15.5%)	46.3 (+1.3%)	32.0 (-27.3%)	36.2 (-18.1%)
TIES+DARE	54.7 (-15.6%)	20.8 (-28.8%)	23.6 (-46.1%)	18.3 (-51.2%)	37.0 (-19.0%)	14.0 (-68.2%)	28.1 (-36.4%)
WIDEN	59.0 (-9.0%)	29.0 (-0.7%)	22.4 (-48.9%)	22.5 (-40.0%)	36.3 (-20.6%)	14.0 (-68.2%)	30.5 (-31.0%)
NeuronMerge	32.7 (-49.5%)	16.2 (-44.5%)	**32.7** (-25.3%)	2.9 (-92.3%)	40.3 (-11.8%)	28.0 (-36.4%)	25.5 (-42.3%)
AIM	61.7 (-4.8%)	31.2 (+6.8%)	22.6 (-48.4%)	38.0 (+1.3%)	**47.7** (+4.4%)	36.0 (-18.2%)	39.5 (-10.6%)
ARM (ours)	62.9 (-2.9%)	**35.4** (+21.2%)	28.5 (-34.9%)	44.9 (+19.7%)	**47.7** (+4.4%)	**48.0** (+9.1%)	**44.6** (+0.9%)

Table 1: Main results with Qwen3-8B experts. τ-bench and OfficeBench report suite averages. WebShop and OS are AgentBench tasks. DB-bench and AlfWorld are **out-of-domain** benchmarks. AVG is the mean over the six aggregates (BEST-of-Three is an oracle expert selector baseline). Parentheses show relative change compared to BEST-of-Three for each aggregate. Best merged model per column is bold; ARM is highlighted.

Model	WebShop	OS	τ-bench	OfficeBench	DB	AlfWorld	AVG
Qwen2.5-7B-Instruct	55.9	31.2	18.1	24.2	50.3	58.0	39.6
Simia-Tau	44.3	12.5	28.7	2.9	37.7	18.0	24.0
Simia-OfficeBench	45.1	18.8	12.6	40.0	36.3	44.0	32.8
Simia-AgentBench	65.6	31.2	21.4	2.9	23.0	10.0	25.7
BEST-of-Three (oracle)	65.6	31.2	28.7	40.0	37.7	44.0	41.2
Average	**67.2** (+2.4%)	30.8 (-1.3%)	18.9 (-34.1%)	4.8 (-88.0%)	38.0 (+0.8%)	48.0 (+9.1%)	34.6 (-16.0%)
Model Stock	63.0 (-4.0%)	31.9 (+2.2%)	16.4 (-42.9%)	36.4 (-9.0%)	51.0 (+35.3%)	62.0 (+40.9%)	43.5 (+5.6%)
Task Arithmetic	66.3 (+1.1%)	29.9 (-4.2%)	14.5 (-49.5%)	6.8 (-83.0%)	41.7 (+10.6%)	42.0 (-4.5%)	33.5 (-18.7%)
TIES	62.0 (-5.5%)	16.7 (-46.5%)	13.3 (-53.7%)	2.9 (-92.8%)	22.3 (-40.8%)	14.0 (-68.2%)	21.9 (-46.8%)
TIES+DARE	50.8 (-22.6%)	13.9 (-55.4%)	15.2 (-47.0%)	2.9 (-92.8%)	21.7 (-42.4%)	12.0 (-72.7%)	19.4 (-52.9%)
WIDEN	41.7 (-36.4%)	11.8 (-62.2%)	18.0 (-37.3%)	2.3 (-94.3%)	17.6 (-53.3%)	10.0 (-77.3%)	16.9 (-59.0%)
NeuronMerge	60.7 (-7.5%)	**35.4** (+13.5%)	18.5 (-35.5%)	30.9 (-22.8%)	**51.3** (+36.1%)	56.0 (+27.3%)	42.1 (+2.2%)
AIM	62.6 (-4.6%)	31.2 (+0.0%)	20.9 (-27.2%)	18.1 (-54.8%)	42.3 (+12.2%)	46.0 (+4.5%)	36.9 (-10.4%)
ARM (ours)	64.2 (-2.1%)	28.5 (-8.7%)	**22.0** (-23.3%)	**40.3** (+0.8%)	**51.3** (+36.1%)	**68.0** (+54.5%)	**45.7** (+10.9%)

Table 2: Results with Qwen2.5-7B experts. Metrics follow Table 1. Parentheses show relative change compared to BEST-of-Three for each aggregate.

(2) **Weight-space merging is highly brittle in interactive agent suites.** Across both backbones, common merge operators show pronounced cross-suite trade-offs, where gains on some environments come with severe regressions on others. This suggests that global parameter blending can easily perturb role-critical behaviors, and such small deviations may cascade into long-horizon failures in multi-turn trajectories.

(3) **ARM improves cross-environment robustness by isolating role-critical circuits and mitigating negative transfer.** Compared to both weight-space and activation-aware baselines, ARM tends to better preserve performance on role-sensitive suites while retaining strong out-of-domain generalization. This is mainly due to (i) role-conditioned tracing that focuses saliency on benchmark-critical spans, and (ii) conflict-aware protection during transplantation that prevents overwriting neurons needed by other environments, thereby reducing destructive interference.

4.3 Ablation Study

We ablate ARM to validate the contribution of each component and to characterize robustness and practical overhead in interactive agent suites. Our design targets two failure modes highlighted in Section 1: (i) backbone instability across weight-space merge operators, and (ii) destructive interference

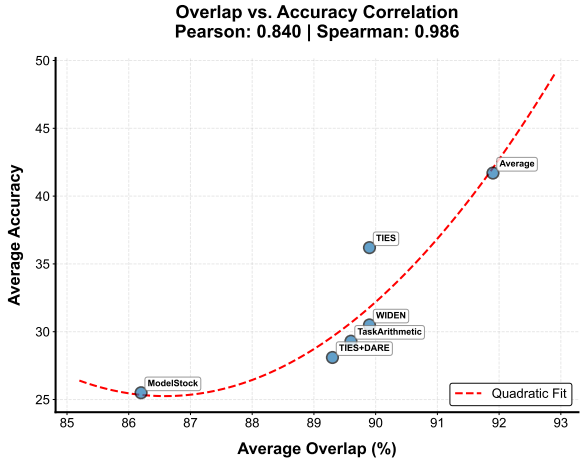

Figure 3: On the Qwen3-8B candidate pool,Activation-Overlap Score (AOS) correlates positively with overall performance (AVG) across candidate merge backbones. Higher AOS tends to yield better AVG, enabling AOS-based selection of a strong initialization without benchmark-specific tuning.

in multi-turn trajectories, where small errors on role-critical spans (tool calls, action serialization, structured outputs) can cascade into repeated failures. Unless otherwise stated, AOS and saliency statistics are computed on the disjoint calibration set (Section 4), without using evaluation data.

Effectiveness of AOS as a lightweight proxy for backbone quality. The goal of AOS is to provide a training-free, benchmark-agnostic selection signal that correlates with downstream cross-suite performance, avoiding full interactive evaluation for every merge candidate. Figure 3 shows a positive relationship between AOS (measured on calibration trajectories) and overall performance (AVG) across candidate backbones. In our candidate pools, selecting the highest-AOS backbone identifies the best-performing initialization in hindsight: Average for Qwen3-8B and Model Stock for Qwen2.5-7B. These results support AOS as a practical criterion for reliably choosing a strong starting point before any neuron-level intervention, which is particularly important given the high variability of merge operators in agent environments.

Role segmentation reduces cross-benchmark interference. A core motivation of ARM is that agent generalization is often bottlenecked by failures on role-critical spans (e.g., tool-call formats or structured final answers), rather than generic language tokens. To test whether role-conditioning makes the traced neuron sets more benchmark-specific, we compare role-conditioned tracing against a role-agnostic variant that computes saliency over all response tokens. Figure 4 visualizes the top-10% salient neurons and highlights neurons shared across benchmark-specific sets. Role-conditioned tracing yields substantially lower cross-benchmark overlap: the overlap rate drops from 61% to 41% on Qwen3-8B, and from 50% to 43% on Qwen2.5-7B. This suggests that restricting tracing to benchmark-critical role spans produces more specialized neuron sets, which is desirable for localized transplantation: fewer shared neurons implies fewer accidental edits to capabilities needed by other environments.

Conflict-aware protection improves robustness. Neuron transplantation can repair benchmark-specific regressions, but directly transplanting all donor-salient neurons may overwrite neurons that are also important for other environments, leading to negative transfer. ARM mitigates this risk via conflict-aware set subtraction (Section 3.3), which removes donor neurons that overlap with the aggregated salient set from the remaining benchmarks.

We ablate this protection by comparing ARM against an unprotected variant and sweeping the per-layer top-k fraction used to define role-salient neurons. Figure 5 shows that the unprotected variant is more sensitive to k: performance decreases more rapidly as k grows, whereas conflict-aware protection yields consistently higher performance and a flatter degradation trend across a wide range of k. Overall, the results suggest that conflict-aware subtraction improves robustness to the choice of k and helps limit negative transfer when the intervention scope increases.

Generalization metrics beyond AVG. To better characterize cross-environment generalization, we report two robustness-oriented summaries in addition to AVG: Worst-suite (WS), the minimum over the six benchmark aggregates, and an oracle-normalized harmonic mean (RHM), which emphasizes balanced performance across suites. Table 3 shows that the AOS-selected initialization is dominated by its weakest suite, resulting in low WS. ARM improves this worst-case robustness while also increasing AVG, raising WS from 19.1 to 28.5 on Qwen3-8B and from 16.4 to 22.0 on Qwen2.5-7B, and substantially improving RHM. Overall, these summaries indicate that ARM yields a more balanced generalist model that approaches the oracle selector more uniformly by alleviating the weakest-suite bottleneck.

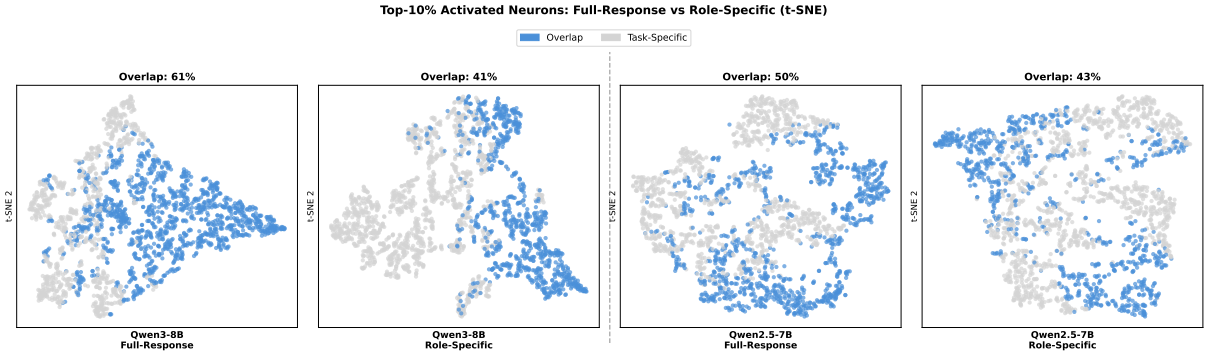

Figure 4: Role-conditioned tracing reduces cross-benchmark overlap of salient neurons. We visualize top-10% salient neurons and highlight neurons shared across benchmark-specific sets. Compared to full-response tracing, role-conditioned tracing yields lower overlap, indicating reduced cross-environment entanglement of the traced circuits.

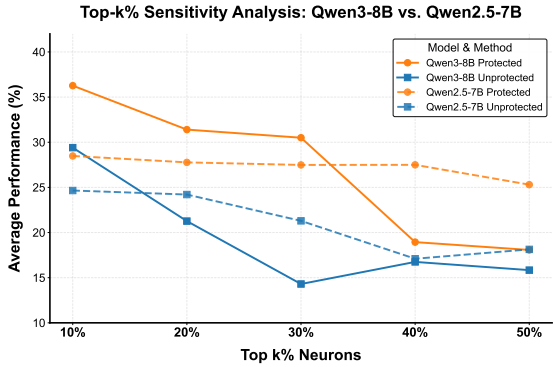

Figure 5: Sensitivity to the top-k fraction used to define role-salient neurons, averaged across the in-domain suites OS, OFFICEBENCH, and τ-BENCH. Conflict-aware protection remains consistently stronger than the unprotected variant as k increases, indicating improved robustness to the saliency threshold.

Backbone	Model	AVG (%)	WS (%)	RHM
Qwen3-8B	AOS-selected	41.8	19.1	0.84
Qwen3-8B	ARM	**44.6**	**28.5**	**0.98**
Qwen2.5-7B	AOS-selected	43.5	16.4	0.95
Qwen2.5-7B	ARM	**45.7**	**22.0**	**1.04**

Table 3: Generalization summaries for the AOS-selected initialization backbone and ARM. AVG is the unweighted mean over six benchmark aggregates. WS is the minimum over the six aggregates. RHM is the harmonic mean of the six aggregate scores normalized by BEST-of-Three. For Qwen3-8B, the AOS-selected initialization is Average; for Qwen2.5-7B, it is Model Stock.

Failure analysis on role-critical errors. A key motivation of ARM is that cross-environment failures in interactive agents are often triggered by localized errors on role-critical spans that can cascade across multi-turn trajectories. To make this failure mode measurable, we leverage the benchmark-specific deterministic parsers already used in our pipeline to identify these spans (tool-call spans for τ-bench, final-answer JSON spans for OfficeBench, and action schema spans for AgentBench), and we inspect representative episodes where the AOS-selected merged backbone fails due to span-level violations. Compared to the AOS-selected backbone, ARM typically repairs the earliest blocking violation, allowing subsequent tool execution to proceed with minimal changes to the remaining trajectory. We provide side-by-side trajectory excerpts and parser-flagged error annotations in Appendix B.

Efficiency and overhead. ARM is training-free and only requires forward-pass activation tracing on a lightweight calibration set; detailed compute, storage, and edit locality statistics are reported in Appendix C.

5 Conclusion

We presented **Agent-Role Merging (ARM)**, a training-free framework for consolidating benchmark-specialized LLM agents into a single generalist checkpoint. ARM addresses two failure modes of agentic model merging: (i) instability across weight-space merge operators, and (ii) destructive interference on role-critical behaviors in multi-turn trajectories. To this end, ARM selects a strong merged initialization using an Activation-Overlap Score computed from role-conditioned activation tracing, and then performs conflict-aware transplantation of a small set of role-salient MLP neurons to repair weak environments while protecting capabilities needed elsewhere. Across both Qwen3-8B and Qwen2.5-7B expert pools, ARM achieves the best overall merged model and substantially improves worst-suite robustness. These results suggest that targeting role-critical circuits enables localized, training-free edits that mitigate negative transfer in interactive agent suites.

6 Limitations

ARM is training-free, but it makes several assumptions that limit its applicability and leave room for future work. First, ARM requires access to homologous expert checkpoints that share the same architecture and tokenizer; it does not directly apply to merging heterogeneous model families or black-box APIs.

Second, ARM relies on activation-level signals to identify role-salient circuits, yet diagnostics tailored to multi-turn interactive agent behaviors remain relatively under-explored. Future advances in activation-based interpretability for agentic settings would likely enable more accurate interventions and further improve performance.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. tau^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Wangyun Gu, Qianghua Gao, Li-Xin Zhang, Xu Shen, and Jieping Ye. 2025. NeuronMerge: Merging models via functional neuron groups. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9015–9037, Vienna, Austria. Association for Computational Linguistics.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *International Conference on Learning Representations*.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. 2024. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pages 207–223. Springer.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, and 11 others. 2025a. Chain-of-Agents: End-to-end agent foundation models via multi-agent distillation and agentic RL. *arXiv preprint arXiv:2508.13167*.

Yuetai Li, Huseyin A. Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. 2025b. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2023. Agentbench: Evaluating LLMs as agents. In *International Conference on Learning Representations*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. ROME: Locating and editing factual associations in GPT. arXiv preprint. *Preprint*, arXiv:2202.05262. NeurIPS 2022; arXiv:2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. MEMIT: Mass-editing memory in a transformer. arXiv preprint. *Preprint*, arXiv:2210.07229. ICLR 2023; arXiv:2210.07229.

Amin Heyrani Nobari, Kaveh Alimohammadi, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. 2025. Activation-informed merging of large language models. *Preprint*, arXiv:2502.02421.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*.

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

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024. OfficeBench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, and Binyuan Hui. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024a. Extend model merging from fine-tuned to pre-trained large language models via weight disentanglement. *arXiv preprint arXiv:2408.03092*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024b. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *Preprint*, arXiv:2311.03099.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. 2025. AgentRL: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *arXiv preprint arXiv:2510.04206*.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. 2025. LifelongAgentBench: Evaluating LLM agents as lifelong learners. *arXiv preprint arXiv:2505.11942*.

A Detailed Experiment Settings

A.1 Baseline Settings

We use publicly available implementations for all baselines whenever possible. Hyperparameters are set as follows:

- **Model Stock** (Jang et al., 2024): we follow the global coefficient setting with `filter_wise=false`, as recommended in the original paper.

- For all other methods—including uniform averaging, task arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023), WIDEN (Yu et al., 2024a), AIM (Nobari et al., 2025), and NeuronMerge (Gu et al., 2025)—we use the default hyperparameters provided by the respective official or paper-reproduced implementations.

No benchmark-specific hyperparameter tuning is performed for any baseline.

A.2 Benchmark Settings

We use the official evaluation harness for each suite. For τ-bench, the user simulator is deterministic with GPT-4.1 at temperature 0, while the evaluated agent uses temperature 0.2 with top-p=1.0 and fixed seeds to control task-order shuffling and sampling. For OfficeBench, AgentBench, DB-bench, and AlfWorld, the benchmark defaults are used with temperature 0.7 and top-p=1.0. The maximum number of new tokens is set to 512 for OfficeBench and 1024 for AgentBench, DB-bench, and AlfWorld. For AlfWorld, we use the standard unseen split with a maximum of 35 steps and 1-shot prompting.

We list the evaluation benchmarks used in this work in Table 4. For benchmarks with multiple subsets (i.e., τ-bench and OfficeBench), we report the macro-averaged results across all subsets.

Domain	Benchmark	Subset	# Tasks
In-domain	τ-bench	Airline	50
		Retail	115
	OfficeBench	2-apps	51
		3-apps	55
	WebShop	–	200
	Operating System	–	144
Out-of-domain	DB-bench	–	300
	AlfWorld	–	50

Table 4: Statistics of In-domain and Out-of-domain Evaluation Datasets.

A.3 Calibration Set Settings

To compute role-conditioned saliency (Section 3.2), a small calibration set $\mathcal{D}_{\text{cal}}$ is constructed from splits that are disjoint from our test set. The composition of the calibration set is listed in Table 5. Deterministic, benchmark-specific parsers are used to trace benchmark-critical spans, including tool-call spans for τ-bench, final-answer JSON spans for OfficeBench, and action schema and argument spans for AgentBench.

B Case Studies: Repairing Role-Critical Failure Cascades

Setup. We analyze representative failure cases to illustrate how merge-induced deviations on role-critical spans can cascade into long-horizon failures in interactive environments. We focus on suites whose role-critical spans are deterministically identifiable by benchmark-specific parsers used in our pipeline: final-answer JSON spans for

Dataset		
τ-bench Retail (train)	500	500
OfficeBench 1-app	93	372
WebShop (dev)	26	208
Operating System (dev)	80	160
Total	**699**	**1240**

Table 5: Composition of the calibration set. To balance the impact of datasets with different sizes, we sample a varying number of trajectories for each dataset, as indicated by the ratio of trajectories to tasks.

Table 6: OfficeBench invalid final JSON episode rate (Qwen2.5-7B pool).

Model	Invalid Episodes	Rate
Backbone (Model Stock)	9/106	8.5%
ARM	5/106	4.7%

OfficeBench, tool-call spans for τ-bench, and action schema spans for OS and WebShop. Unless otherwise noted, we compare the AOS-selected initialization backbone against ARM under the same decoding and evaluation settings.

B.1 OfficeBench: Structured Output (JSON) Violations

Span-level violation rate. Table 6 reports the fraction of episodes with invalid final structured outputs that cannot be parsed by the evaluator. ARM reduces invalid episodes from 8.5% to 4.7%.

Representative failure modes. Across failures, the backbone commonly violates the required action/answer structure by nesting a JSON action inside a string field, emitting the entire `<think><answer>` template as a string, or producing malformed escapes that break JSON parsing, often followed by an invalid got_stuck action.

Case studies. We show three representative OfficeBench episodes. In each, the backbone fails due to a structured-output violation, while ARM preserves the required schema and completes the workflow.

OfficeBench Task 2-14: reward meetings for top-3 students

Backbone (Model Stock) – Fail

```
{"action": " {\"app\": \"shell\", \"action\": \"command\",

  \"command\": \"sort -nr -k2,2 score.xlsx | head -3\"}
     "}
{"action": "{\"app\": \"system\", \"action\": \"got_stuck
     \"}"}
[Error] Malformed action string. Expected <think>...</
     think><answer>[JSON]</answer>.
Result: Fail
```

Table 7: Common structured-output error patterns on OfficeBench.

Error Type	Description (example)
JSON nesting	Action JSON is wrapped as a quoted string inside a field (e.g., `"action": "{...}"`).
Template confusion	The model outputs `<think>...</think><answer>...</answer>` as a literal string field.
Escape error	Invalid escape sequences cause JSON parsing failure (e.g., `Invalid \\escape`).
Crash action	The model emits an invalid fallback action (e.g., `got_stuck`) that violates the required schema.

ARM – Success

```
{"app":"excel","action":"read_file","file_path":".../
     score.xlsx"}
{"app":"calendar","action":"create_event","user":"Alice",
     ...}
{"app":"calendar","action":"create_event","user":"David",
     ...}
{"app":"calendar","action":"create_event","user":"Carol",
     ...}
{"app":"system","action":"finish_task","answer":"None"}
Result: Success
```

Figure 6: OfficeBench Task 2-14. The backbone violates the required structured format and crashes; ARM preserves valid actions and completes the multi-app workflow.

OfficeBench Task 3-49: pair students and notify via calendar + email

Backbone (Model Stock) – Fail

```
{"action":"<think>...</think><answer> {\"app\":\"system\",

\"action\":\"switch_app\",\"target_app\":\"excel\"} </
     answer>"}
[Error] Malformed action string. Expected <think>...</
     think><answer>[JSON]</answer>.
Result: Fail
```

ARM – Success

```
{"app":"excel","action":"read_file","file_path":".../
     schedule.xlsx"}
{"app":"calendar","action":"create_event","user":"Alice",
     ...}
{"app":"calendar","action":"create_event","user":"Carol",
     ...}
{"app":"email","action":"send_email","sender":"Alice",
     ...}
{"app":"email","action":"send_email","sender":"Carol",
     ...}
{"app":"system","action":"finish_task","answer":"None"}
Result: Success
```

Figure 7: OfficeBench Task 3-49. The backbone emits the full think-answer template as a literal string; ARM produces valid structured actions and finishes the task.

OfficeBench Task 3-7: invalid escape sequence

Backbone (Model Stock) – Fail

```
[Error] Invalid \\escape ... Malformed action!
{"action":"{\"app\":\"system\",\"action\":\"got_stuck
     \"}"}
Result: Fail
```

ARM – Success

```
{"app":"excel","action":"read_file","file_path":".../
    students.xlsx"}
{"app":"word","action":"write_to_file","file_path":".../
    admission.docx", ...}
{"app":"email","action":"send_email","recipient":"
    jennifer.gonzalez@example.com", ...}
...
Result: Success
```

Figure 8: OfficeBench Task 3-7. The backbone triggers a JSON parsing error (invalid escape); ARM maintains valid structured outputs and successfully completes multi-recipient execution.

B.2 τ-bench: Tool-Call Failure Cascades

Representative tool-call cascades. We present three cases where the backbone either repeats a failing tool call without correcting the underlying issue, makes redundant queries and acts on the wrong target, or omits a required critical tool action.

τ-bench Task 13: error loop after a tool failure

Backbone – Fail (loop)

```
update_reservation_passengers -> Error (passengers
    mismatch)
cancel_reservation -> OK
book_reservation -> Error (payment amount does not add up
    )
book_reservation -> Error (repeated many times)
Reward: 0.0
```

ARM – Success

```
get_reservation_details -> OK
search_direct_flight -> OK
...
Reward: 1.0
```

Figure 9: τ-bench Task 13. The backbone enters a repeated error loop after a tool failure; ARM resolves the issue and completes without looping.

τ-bench Task 31: redundant queries and wrong target

Backbone – Fail

```
get_reservation_details called repeatedly on unrelated
    reservations
cancel_reservation -> cancelled an incorrect booking
Reward: 0.0
```

ARM – Success

```
get_user_details -> OK
cancel_reservation(correct id) -> OK
Reward: 1.0
```

Figure 10: τ-bench Task 31. The backbone makes redundant queries and cancels the wrong booking; ARM cancels the intended reservation directly.

Table 8: Validation metrics on AgentBench. "Invalid Action" indicates malformed agent outputs rejected by the environment. "Task Limit" indicates failure to complete within the maximum step limit. All values are percentages (%).

Model	Benchmark	Invalid Action		Task Limit	
		Backbone	ARM	Backbone	ARM
Qwen3-8B	OS	2.08	1.39	12.50	8.33
	WebShop	0.00	0.00	0.50	3.00
Qwen2.5-7B	OS	0.69	0.00	15.97	18.75
	WebShop	0.00	0.00	5.00	0.50

OS: command choice affects execution correctness

Backbone (Average) – Failed

```
touch logfile.txt
echo "abc" >> logfile.txt   # includes newline
tr -cd '[:alnum:]' < logfile.txt | sort -u | wc -l
Answer: 1 (Wrong)
```

ARM – Success

```
echo -n "abc" > logfile.txt   # no newline
fold -w1 | sort | uniq | wc -l
Answer: 3 (Correct)
```

Figure 11: OS case study. The backbone fails due to incorrect file handling, while ARM succeeds.

τ-bench Task 46: missing a required critical action

Backbone – Fail

```
get_user_details -> OK
get_reservation_details -> OK
(terminated without send_certificate)
Reward: 0.0
```

ARM – Success

```
get_user_details -> OK
get_reservation_details -> OK
send_certificate(amount=50) executed
Reward: 1.0
```

Figure 12: τ-bench Task 46. The backbone omits a required tool action; ARM executes the critical step and completes the task.

B.3 OS and WebShop: Action Schema and Execution Errors

Validation signals and task outcomes. Tables 8 report validation signals (invalid action and task-limit timeouts) on OS and WebShop.

Qualitative example. Figure 11 shows a representative OS episode in which both models emit valid actions, but the backbone executes an imprecise command and returns an incorrect answer.

Summary. Across suites, these cases show that many merge failures originate from localized violations on role-critical spans or early tool/action mistakes that derail multi-turn trajectories. ARM frequently prevents such cascades by preserving re-

quired structured formats and executing key tool/action steps more reliably.

C Efficiency and Overhead Details

ARM is training-free and operates via forward-pass tracing on a lightweight calibration set. In our setup, AOS-based backbone selection traces activations for six candidate backbones across four in-domain benchmarks, costing $\sim$0.5 GPU-hour per backbone–benchmark pair on a single H20 (about 12 GPU-hours total); once the backbone is selected, the merge and neuron transplantation completes in under 20 minutes. At the benchmark level, each transplant set is typically small (roughly 2–3% for τ-bench, OfficeBench, and WebShop). Activation statistics are stored in compressed NPZ files, requiring less than 500MB total. Overall, these results indicate that ARM can produce a more robust generalist agent with modest one-time calibration cost and targeted neuron-level edits, without any additional training.