M2A: Synergizing Mathematical and Agentic Reasoning in Large Language Models

Junjian Wang[1,2] **Xin Zhou**[3*] **Qiran Xu**[3] **Kun Zhan**[3]

[1]Foundation Model Research Center, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Li Auto Inc.

wangjunjian2025@ia.ac.cn {zhouxin12,xuqiran,zhankun}@lixiang.com

Abstract

While reasoning has become a central capability of large language models (LLMs), the reasoning patterns required for different scenarios are often misaligned. Mathematical reasoning typically relies on intrinsic logic to solve closed-world problems in a single response, whereas agentic reasoning requires not only internal reasoning but also multi-turn interaction with external environments, interleaving thought and action. This misalignment prevents mathematical and agentic reasoning from effectively benefiting from each other, often yielding unstable reasoning behavior and only limited performance gains under multi-task learning. In this paper, we propose M2A, a novel paradigm that synergizes mathematical and agentic reasoning via model merging. To avoid overfitting to superficial reasoning patterns under joint training, M2A operates directly in parameter space: it identifies the feature subspace critical for agent behavior, and merges the mathematical reasoning task vector only along its null space, thereby injecting reasoning capability along directions that do not perturb agent behavior. Unlike SFT or RL, M2A requires no additional gradient-update and exposes the merging coefficient as a simple knob for controlling reasoning length. Experiments in a challenging real-world coding agent setting show that our method effectively extends agentic reasoning depth and delivers substantial performance improvements. Applied to a fine-tuned Qwen3-8B, M2A improves its SWE-Bench Verified resolved rate from 44.0% to 51.2% without retraining the model. Further experiments demonstrate that M2A generalizes across model families, agent scaffolds, and task domains.

1 Introduction

Reasoning serves as the core of human intelligence, driving critical processes from problem-solving to strategic planning. In the realm of Large Language Models (LLMs), chain-of-thought prompting [40, 18] shows that eliciting reasoning can unlock substantial gains on complex tasks, and recent large reasoning models [15, 10, 4] further advanced *mathematical reasoning* capability through reinforcement learning with verifiable rewards (RLVR), achieving strong performance in mathematics, competitive programming, and broader STEM domains. More recently, the frontier of reasoning has shifted beyond closed-world problem solving toward *agentic reasoning*, where models must reason while interacting with external environments over multiple turns [41]. Prior work such as ReAct [51] highlights that tightly coupling reasoning with action is essential for complex interactive tasks, making the improvement of agentic reasoning increasingly important for LLM-based agents.

*Corresponding author.
Code is available at https://github.com/Laplucky/M2A.

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

Although reasoning is crucial across many tasks, the behavior patterns required by different reasoning scenarios are often misaligned, particularly between mathematical reasoning and agentic reasoning. Mathematical reasoning is primarily a process of *internal thinking*: the model works in a relatively closed-world setting and aims to produce a complete solution in a single response [28, 3, 12]. In contrast, agentic reasoning is better characterized as a process of *coordinating internal thinking with external action*: the model must keep interacting with an open-world environment over multiple turns and decide when to think and when to act [51, 32, 41]. Such contrasts across task domains, interaction formats, and reasoning styles prevent multi-task learning from improving both capabilities together, and can even let mathematical reasoning patterns override agentic behaviors. Recent studies [12, 20, 36] and our experiments (Table 1) show a consistent pattern: multi-task SFT [12] or RL [36] boosts mathematical reasoning but barely helps on agent tasks.

In this paper, we present M2A, a novel paradigm that synergizes mathematical and agentic reasoning in a training-free manner. To avoid fitting mismatched data distribution during multi-task training, **M2A couples the two capabilities directly in parameter space through model merging**. While this mitigates the training-stage mismatch, finer-grained conflicts still remain in parameter space. Naively merging [14, 43, 7] a reasoning model into an agent model can overwrite the agent's interaction behavior: the merged model reasons for substantially longer but interacts with the environment far less frequently, reducing external actions and ultimately degrading agent performance. This observation motivates us to reformulate our goal as a **behavior-preserving capability integration problem**—how to combine mathematical and agentic reasoning without disrupting the agent's interaction pattern. Taking inspiration from [39, 6, 13], M2A identifies the parameter subspace critical for agent behavior and applies null-space projection to the mathematical-reasoning update before merging it into the agent model, thereby integrating the two capabilities while minimally perturbing agentic behavior. To further mitigate capability conflicts at a finer granularity, **we introduce an adaptive layer-wise merging coefficient mechanism**, which additionally serves as a natural knob for adjusting reasoning length on top of existing models. Compared with costly SFT [37, 35] or RL [34, 21], M2A requires no further training to the existing model, and offers a simple yet effective mechanism for controlling reasoning length.

We evaluate M2A and baselines on SWE-Bench Verified [17], a challenging coding-agent benchmark that requires resolving real-world GitHub issues. Our experiments show that multi-task SFT, while able to transfer superficial reasoning traces, fails to translate them into agent capability and can actively hurt agent performance. In contrast, M2A improves the Qwen3-8B baseline from 44.0% to 51.2% on SWE-Bench Verified, while growing the average per-step reasoning length from 253.3 to 327.4 tokens. More importantly, M2A makes this transfer controllable: we observe that the average reasoning length grows monotonically and nearly linear with the merging coefficient, turning the coefficient into a predictable knob for modulating agent reasoning behavior. The ablation study demonstrates the efficacy of each module within the M2A framework. Our analysis further reveals that M2A not only enhances per-step reasoning depth but also shifts the agent's trajectory-level behavior toward more frequent external actions. Additional experiments on Llama-3.1-8B, Mini-SWE-Agent [47], R2E [16], and GAIA [25] further demonstrate the generalizability of M2A across model families, agent scaffolds, and task domains.

Our contributions are as follows: (1) We propose M2A, a novel paradigm that synergizes mathematical reasoning and agentic reasoning in LLMs in a training-free manner. M2A preserves the agent's external action behavior while injecting stronger internal reasoning through behavior-preserving null-space model merging. (2) On SWE-Bench Verified, M2A improves a fine-tuned Qwen3-8B from 44.0% to 51.2% resolved rate, consistently outperforming multi-task SFT and representative model-merging baselines. (3) We conduct systematic ablations and analysis to demonstrate the effectiveness of M2A, and show that merge strength serves as an effective control knob over reasoning length, enabling explicit control of the reasoning behavior.

2 Related Work

2.1 Generalization of LLM Reasoning

LLM reasoning has become a central mechanism for improving complex problem solving [40, 15, 10]. However, reasoning gains do not always generalize beyond the training domain [2, 12]. Although mathematical reasoning is widely viewed as a general source of reasoning capability,

math-specialized models often bring limited benefits to broader tasks such as scientific QA, coding, and agent planning [12, 2]. Recent work therefore explores broader reasoning data and multi-domain RL objectives [33, 24, 28, 30, 36], but the resulting generalization remains sensitive to the target behavior. This limitation is especially pronounced for agentic reasoning. Prior studies show that math reasoning does not consistently improve broader LLM capabilities [12]; multi-task RL can improve several reasoning-intensive domains but does not reliably enhance agent performance [36]; and reasoning and tool-use behaviors may interfere under joint optimization [20]. These findings suggest that mathematical and agentic reasoning cannot be reliably combined by simply mixing their training data. Effective synergy requires strengthening internal thinking while preserving the external action behavior required by agents. This motivates our parameter-level approach, which integrates mathematical reasoning updates through behavior-preserving model merging rather than joint training.

2.2 Model Merging

Model merging aims to combine multiple fine-tuned models into a single model without additional training, reducing storage and computational costs [5, 23, 29, 46]. Early methods such as Model Soups [42] and Task Arithmetic [14] showed that weight averaging or task-vector composition can be effective when source models are sufficiently aligned. Later approaches, TIES-Merging [43] alleviates parameter interference via pruning, symbol election, and merging, DARE [52] enhances merged model robustness and base capabilities by randomly dropping and rescaling fine-tuning weights. Recent studies [49, 53, 55] have explored geometry and subspace-aware formulations to better preserve model structure during composition. However, most model merging methods are primarily designed for relatively aligned capabilities. RAIN-Merging [13] introduces null-space projected merging to preserve the thinking format of reasoning models while enhancing instruction-following ability. But it focuses on preserving the structured thinking format within a single response, without considering the multi-turn think–act–observe behavior required by agents. Moreover, its instruction-attention-guided merge control is designed for instruction following and is less applicable to agent tasks. In contrast, M2A extends null-space projected merging to synergize mathematical and agentic reasoning by explicitly calibrating agent-critical behavioral features and adaptively controlling both merge layers and merge strength.

3 Methodology

In this section, we first define the problem setting for synergizing mathematical and agentic reasoning, and clarify why traditional multi-task learning and model merging are insufficient. We then introduce M2A, a novel paradigm that leverages null-space model merging to integrate mathematical reasoning into an agent model while maintaining its agent behavior pattern.

3.1 Problem Statement

We consider a multi-turn ReAct-style agent dataset $\mathcal{D}_{\text{agent}}$ and a single-turn mathematical reasoning dataset $\mathcal{D}_{\text{math}}$. Our goal is to obtain a single model that synergizes both capabilities, achieving strong performance on agent tasks while benefiting from mathematical reasoning.

Multi-task SFT A standard approach to combine the two capabilities is joint fine-tuning on $\mathcal{D} = \mathcal{D}_{\text{agent}} \cup \mathcal{D}_{\text{math}}$:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}) \right], \tag{1}$$

with the hope that the long chain-of-thought behavior elicited by the math data can generalize to agentic reasoning [40, 10]. However, the two data distributions diverge across task domains, interaction formats, and reasoning styles, causing joint optimization to fit surface-level patterns rather than transferable reasoning capabilities [24, 12]. As a result, even if multi-task SFT successfully transfers the surface-level "long reasoning" style, this increased length provides little benefit to—and sometimes even degrades—actual agent performance.

Naive Model Merging A training-free alternative is model merging [42, 14, 43], which combines capabilities directly in parameter space and thereby avoids the data-level conflicts of joint optimization. Given the shared base model θ_0, the task vectors $\Delta\theta_{\text{agent}} = \theta_{\text{agent}} - \theta_0$ and $\Delta\theta_{\text{reason}} = \theta_{\text{reason}} - \theta_0$, a naive merge:

$$\theta_{\text{merge}} = \theta_{\text{agent}} + \alpha\Delta\theta_{\text{reason}}, \tag{2}$$

where α is a scalar controlling the strength of the injected reasoning signal. However, operating in parameter space alone is not sufficient [13]: naive merging treats all parameter directions equally, so $\Delta\theta_{\text{reason}}$ may perturb the directions that govern the agent's think–act–observe loop, causing the merged model θ_{merge} to overthink instead of act, or to continue generating when it should seek environmental feedback.

3.2 M2A: Behavior-Preserving Reasoning Synergy

M2A formulates the synergy between mathematical and agentic reasoning as a behavior-preserving capability integration problem: how to combine mathematical and agentic reasoning without disrupting agent's interaction pattern. M2A consists of three stages: (i) calibrating the agent-critical subspace from behavior markers, (ii) projecting the reasoning task vector onto the null space of agent-critical subspace to prevent overriding agentic behavior, and (iii) calibrating layer-wise merge coefficients and selecting compatible layers for merging.

3.2.1 Agent-Critical Behavior Calibration

The first stage of M2A is to identify the subspace that encodes agent-critical behaviors—the interaction patterns that a successful agent must preserve. If these behaviors are perturbed, the merged model may continue reasoning when it should act, or fail to respond appropriately to new observations.

Behavior Markers An agent's think–act–observe behavior is most clearly reflected in when it switches between internal thinking and external action: starting or stopping reasoning, making tool calls, or pausing for external feedback [51, 41]. These transitions appear in the trajectory as a few special tokens, and the activations around them carry the strongest signals of the agent's interaction behavior. We therefore identify these transitions through a set of markers:

$$\mathcal{M}_{\text{native}} = \{\text{<think>}, \text{</think>}, \text{<tool_call>}, \text{</tool_call>}\}, \tag{3}$$

These markers follow our agent's chat template; other formats can be supported by redefining $\mathcal{M}$.

Agent-critical Subspace For each marker, we extract activations from its local neighborhood of r tokens, chosen to capture the transition from both sides: for start markers, the activations at positions $[t_{\text{marker}}, t_{\text{marker}} + r]$ reflect the behavior after entering a new state; for end markers, the activations at positions $[t_{\text{marker}} - r, t_{\text{marker}}]$ reflect the decision to exit the current state. For each layer l, we collect the corresponding input hidden states from the agent model into a calibration matrix:

$$\mathbf{C}^{(l)} = [\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_{n_l}^{(l)}] \in \mathbb{R}^{d \times n_l}. \tag{4}$$

where d is the hidden dimension and n_l is the number of collected features. The subspace spanned by these calibration features:

$$S_{\text{agent}}^{(l)} = \text{span}(\mathbf{C}^{(l)}), \tag{5}$$

captures the representation directions most relevant to the agent's think–act–observe behavior pattern. Details are provided in Appendix B.

3.2.2 Null-space Projected Merging

The second stage is to merge the reasoning task vector into the agent model without perturbing the agent-critical subspace $S_{\text{agent}}^{(l)}$. Since model merging operates on every weight matrix of the LLM, for notational simplicity we use $\Delta\mathbf{W}_{\text{reason}}^{(l)}$ to denote the reasoning task vector at layer l for any such matrix. For any protected feature $\mathbf{h}_{\text{agent}} \in S_{\text{agent}}^{(l)}$, naive merging induces an undesired perturbation:

$$\Delta\mathbf{h} = \Delta\mathbf{W}_{\text{reason}}^{(l)}\mathbf{h}_{\text{agent}}, \tag{6}$$

where $\Delta\mathbf{h} \neq 0$ shifts the hidden representation away from its calibrated directions and can disrupt the agent's think–act–observe behavior. To eliminate this perturbation, we seek a refined task vector

$\widehat{\Delta\mathbf{W}}^{(l)}_{\text{reason}}$ that eliminates interference within the protected subspace $S^{(l)}_{\text{agent}}$ while minimizing the deviation from the original reasoning behavior. This is formulated as a constrained optimization problem:

$$\widehat{\Delta\mathbf{W}}^{(l)}_{\text{reason}} = \arg\min_{\mathbf{W}} \|\mathbf{W} - \Delta\mathbf{W}^{(l)}_{\text{reason}}\|_F^2 \quad \text{s.t.} \quad \mathbf{W}\mathbf{h} = \mathbf{0}, \ \ \forall \mathbf{h} \in S^{(l)}_{\text{agent}}. \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. To solve this problem, we resort to null-space projection [39, 13]. Let $\mathbf{Q}^{(l)} \in \mathbb{R}^{d\times r_l}$ be an orthonormal basis of $S^{(l)}_{\text{agent}}$ (e.g., obtained via SVD of $\mathbf{C}^{(l)}$), where r_l is the rank of $S^{(l)}_{\text{agent}}$. The projector onto the orthogonal complement of $S^{(l)}_{\text{agent}}$ is:

$$\mathbf{P}^{(l)}_{\text{null}} = \mathbf{I} - \mathbf{Q}^{(l)}\mathbf{Q}^{(l)\top}, \tag{8}$$

and the refined task vector is obtained by projecting $\Delta\mathbf{W}^{(l)}_{\text{reason}}$ along its input dimension:

$$\widehat{\Delta\mathbf{W}}^{(l)}_{\text{reason}} = \Delta\mathbf{W}^{(l)}_{\text{reason}}\,\mathbf{P}^{(l)}_{\text{null}}. \tag{9}$$

For any $\mathbf{h} \in S^{(l)}_{\text{agent}}$, we can write $\mathbf{h} = \mathbf{Q}^{(l)}\mathbf{a}$ for coefficient vector $\mathbf{a}$. By the orthonormality of $\mathbf{Q}^{(l)}$:

$$\widehat{\Delta\mathbf{W}}^{(l)}_{\text{reason}}\mathbf{h} = \Delta\mathbf{W}^{(l)}_{\text{reason}}\mathbf{P}^{(l)}_{\text{null}}\mathbf{Q}^{(l)}\mathbf{a} = \Delta\mathbf{W}^{(l)}_{\text{reason}}\left(\mathbf{Q}^{(l)} - \mathbf{Q}^{(l)}\mathbf{Q}^{(l)\top}\mathbf{Q}^{(l)}\right)\mathbf{a} = \mathbf{0}, \tag{10}$$

which means the projected task vector introduces no change to agent-critical direction. Moreover, $\widehat{\Delta\mathbf{W}}^{(l)}_{\text{reason}}$ is the Frobenius-optimal solution among all matrices satisfying this constraint. In this way, null-space projection selectively removes the components of $\Delta\mathbf{W}^{(l)}_{\text{reason}}$ that conflict with agent behavior, while keeping the rest intact for reasoning integration. In practice, we implement this projection with an equivalent block-wise matrix-free solver to avoid explicitly materializing the full projector; see Appendix D.

3.2.3 Adaptive Layer-Wise Merging

In practice, null-space projection alone is insufficient. We find that the norms of $\Delta\theta^{(l)}_{\text{agent}}$ and $\Delta\theta^{(l)}_{\text{reason}}$ vary substantially across layers (see Figure 9). Consequently, a uniform merge coefficient can cause the reasoning task vector to dominate the agent's parameters, degrading its agentic behavior. We therefore explore the synergistic balance between internal thinking and external action by introducing two complementary mechanisms that control *how much* and *where* math reasoning is merged.

Merge Coefficient Calibration To address the scale imbalance, we introduce a layer-wise calibration mechanism. We normalize the reasoning task vector to the same scale as the agentic baseline, ensuring the integration is balanced across all layers:

$$\alpha_l = \beta \cdot \frac{\|\Delta\theta^{(l)}_{\text{agent}}\|_F}{\|\Delta\theta^{(l)}_{\text{reason}}\|_F + \epsilon}. \tag{11}$$

where β is a global scaling factor and ϵ is a small constant for numerical stability. This calibration removes the need to hand-tune the merge coefficient [5, 43, 45] layer by layer and turns β into a single interpretable hyperparameter: smaller β preserves the agent's original interaction behavior, while larger β injects stronger mathematical reasoning.

Similarity-aware Layer Mask Even after scale calibration, merging layers with conflicting updates can still destabilize the agent. We therefore select only those layers where the two task vectors are directionally aligned. Specifically, we compute the cosine similarity $\text{sim}^{(l)}$ between $\Delta\theta^{(l)}_{\text{agent}}$ and $\Delta\theta^{(l)}_{\text{reason}}$, and retain layers whose similarity exceeds the global mean $\bar{\text{sim}}$:

$$M_l = \mathbb{I}\left[\text{sim}^{(l)} > \bar{\text{sim}}\right]. \tag{12}$$

where $M_l \in \{0, 1\}$ indicates whether layer l participates in the merge. This filters out layers where the two task vectors are less aligned, focusing the merge on regions with natural synergy and improving overall merging efficiency.

Final Merge Rule Combining null-space projection, layer-wise coefficient calibration, and similarity-aware layer mask, our final merge rule for each mergeable weight matrix at layer l is:

$$\mathbf{W}_{\text{merge}}^{(l)} = \mathbf{W}_{\text{agent}}^{(l)} + M_l \, \alpha_l \, \widehat{\Delta \mathbf{W}}_{\text{reason}}^{(l)}. \tag{13}$$

In this way, M2A yields a merged model that synergizes mathematical and agentic reasoning: it strengthens internal reasoning while preserving the behaviorally important structure of multi-turn interaction. The layer-wise calibration also reduces coefficient sensitivity and exposes β as an explicit control knob for shaping merged model behavior. Implementation details for computing α_l and M_l from layer-wise mergeable blocks are provided in Appendix D.1.

4 Experiments

4.1 Experimental Setup

Baselines. We build our experiments on Qwen3-8B-Base [44] and compare our approach with two baseline families: SFT and model merging. Regarding SFT, we employ DeepSeek-R1-0528-Qwen3-8B as the Reasoning-8B model. We also train two additional models using Step-Fun's SFT dataset[2]: **Agent-8B**, trained on 30k curated coding-agent samples, and **Multi-Task-8B**, trained on a 60k mixture of coding-agent and math-reasoning data. For model merging, we adopt five representative baselines: **Task Arithmetic** [14], **TIES-Merging** [43], **DARE** [52], **SLERP** [7], and **RAIN-Merging** [13]. We apply M2A on top of Agent-8B, and all model-merging baselines use Agent-8B and Reasoning-8B as their source models. Further details are provided in Appendix A.

Evaluation. We evaluate M2A and baselines on the challenging coding-agent benchmark SWE-Bench Verified [17] under the OpenHands scaffold [27], where each task requires the model to reason and interact with the environment to resolve real-world software engineering issues. We report the Resolved Rate as the metric of agent performance; unless otherwise specified, all reported results are averaged over three independent runs (Avg@3). We also evaluate mathematical reasoning on AIME2024 [3], AIME2025 [3], and MATH500 [11]. Details are shown in Appendix A.

4.2 Main Results

Table 1 shows the results of the proposed method and baselines. To provide a more comprehensive view of our method's effectiveness, we additionally report the performance of various models and methods of different scales. From the results, we can find that:

Multi-task SFT struggles to transfer mathematical reasoning to agentic reasoning under data-level mismatch. Multi-task SFT increases the average reasoning length from 253.3 to 347.4, yet the average number of interaction steps (Avg. Step) drops from 175.3 to 150.1 and the resolved rate decreases from 44.0% to 41.1%. In other words, the longer reasoning traces do not translate into better agentic problem solving; instead, the model interacts less with the environment and solves fewer tasks. This suggests that jointly optimizing over mathematical and agentic data only transfers the surface pattern of long reasoning; simply mixing math reasoning and agent data is therefore insufficient for reasoning synergy.

Naive model merging suffers from behavior pattern mismatch. The reasoning model and the agent model follow different behavior patterns: the former solves problems through single-turn internal thinking, while the latter must maintain a multi-turn think–act–observe loop with the environment. If merging methods operate directly in parameter space without distinguishing these behavior patterns, the merged model can be dominated by the single-turn reasoning behavior, empirically exhibit substantially longer reasoning length together with far fewer interaction steps. Task Arithmetic [14] and SLERP [7] do achieve modest gains over the base agent (47.6% and 47.2% vs. 44.0%), but this comes at the cost of nearly halving the interaction frequency (Avg. Step 87.6 vs. 175.3) and tripling the reasoning length. The merged models compensate for shorter interaction with longer single-turn thinking, rather than genuinely synergizing the two behaviors, and the gain is still noticeably below M2A. Less stable methods degrade further: TIES-Merging [43] drops to 39.0% (Avg. Step 112.1), and DARE [52] collapses to 22.0% with Avg. Step drops sharply to 68.6. RAIN-Merging [13] instead

[2]https://huggingface.co/datasets/stepfun-ai/Step-3.5-Flash-SFT

Table 1: Main results on SWE-Bench Verified. Avg. Reasoning Length is the number of reasoning tokens per step, and Avg. Step represents the number of turns the model interacts with the environment. Results are averaged over 3 runs, subscripts denote standard deviation.

Model	Params	Training	Score (%) ↑	Avg. Reasoning Length	Avg. Step
Proprietary Models					
OpenAI-GPT-5.2-Thinking [26]	–	–	80.0	–	–
Gemini-3.1-Pro [8]	–	–	80.6	–	–
Claude-Opus-4.6 [1]	–	–	80.8	–	–
Open-Source Models					
SWE-agent-LM-7B [48]	7B	SFT	15.2	–	–
SWE-Mirror-LM-7B [38]	7B	SFT	22.8	–	–
SWE-Dev-7B [37]	7B	SFT + RL	23.4	–	–
Klear-Agent-8B-RL [19]	8B	SFT + RL	40.4	–	–
SWE-AGILE [21]	8B	SFT + RL	24.1	–	–
SWE-Lego-Qwen3-8B [35]	8B	SFT	42.2	–	–
SERA-8B [31]	8B	SFT	37.1	–	–
DeepSWE-32B-Preview [22]	32B	SFT + RL	42.2	–	–
SWE-Lego-Qwen3-32B [35]	32B	SFT	52.6	–	–
SWE-World-32B-RL [34]	32B	SFT + RL	55.0	–	–
Kimi-Dev-72B [50]	72B	SFT + RL	48.6	–	–
Our Implementations					
Agent-8B	8B	SFT	$44.0_{\pm 0.9}$	253.3	175.3
Reasoning-8B	8B	SFT	$0.2_{\pm 0.1}$	**1800.7**	9.2
Multi-Task-8B	8B	SFT	$41.1_{\pm 1.4}$	347.4	150.1
Task Arithmetic [14]	8B	Training-free	$47.6_{\pm 0.9}$	836.9	87.1
TIES-Merging [43]	8B	Training-free	$39.0_{\pm 0.9}$	494.6	112.1
DARE [52]	8B	Training-free	$22.0_{\pm 0.8}$	1219.1	68.6
SLERP [7]	8B	Training-free	$47.2_{\pm 0.7}$	747.4	87.6
RAIN-Merging [13]	8B	Training-free	$43.2_{\pm 1.1}$	262.7	170.4
M2A (Ours)	8B	Training-free	$\mathbf{51.2}_{\pm \mathbf{0.6}}(\uparrow 7.2)$	327.4	**178.0**

preserves the agent's behavior, but for the same reason it barely injects any additional reasoning into the model, and its resolved rate (43.2%) shows no improvement over the base agent (44.0%). This indicates that preserving single-turn long reasoning pattern is insufficient for agents. Effective reasoning synergy requires jointly strengthening single-turn reasoning and preserving the multi-turn interaction behavior that drives external action. We carefully tuned the hyperparameters for the baselines, and additional experimental results are presented in the Appendix A.3.

M2A establishes a novel paradigm for synergizing mathematical and agentic reasoning. By addressing the mismatches identified above, M2A improves `Agent-8B` **from 44.0% to 51.2%** on SWE-Bench Verified, yielding a substantial +7.2 point gain. This result is even better than several larger models trained with SFT or RL. Compared with Multi-task SFT, M2A avoids joint optimization over the mismatched mathematical and agentic data. Compared with naive model merging, M2A further addresses the behavior pattern mismatch between single-turn mathematical reasoning and multi-turn agentic interaction by protecting agent-critical behavior directions through null-space projection. As a result, M2A strengthens internal thinking and preserves agentic interaction: Avg. reasoning length increases from 253.3 to 327.4, while Avg. Step remains high at 178.0. Consequently, M2A achieves the best agentic performance among all compared methods. These results suggest that mathematical and agentic reasoning are most effectively synergized when the integration mechanism enhances internal reasoning while preserving external action behavior. Beyond this, M2A requires no costly training, synergistically improves mathematical reasoning as well, and further offers a controllable mechanism for tuning reasoning length, as we demonstrate in later sections.

4.3 Synergistic Effect and Ablation Study

Synergistic Improvement on Mathematical and Agentic Reasoning M2A is designed to synergize mathematical and agentic reasoning. This design suggests that the merged model should not only

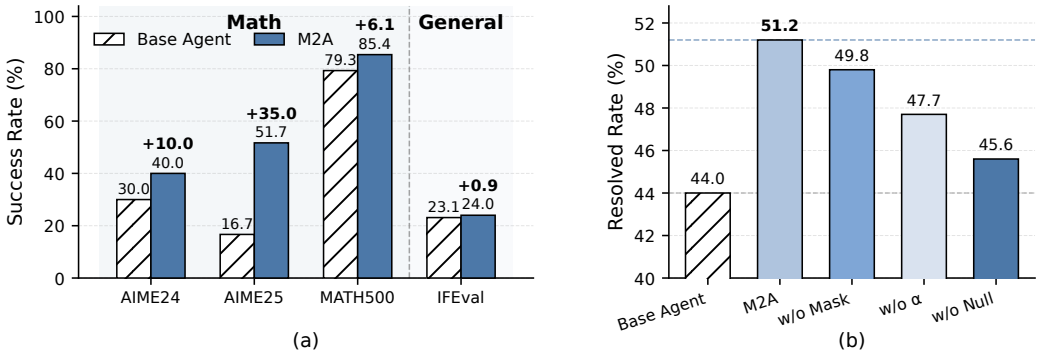

Figure 1: (a) The performance of M2A on mathematical and general capability benchmarks; (b) Ablation study of M2A on SWE-Bench Verified.

improve agentic tasks, but also strengthen the mathematical reasoning capability. Our experiments indicate that M2A produces a **synergistic effect**: the agent becomes better at interactive problem solving while also gaining stronger intrinsic reasoning ability. As shown in Figure 1 (a), M2A consistently improves mathematical reasoning while also boosting SWE-Bench Verified performance. These gains indicate that the injected reasoning component remains effective for closed-world mathematical reasoning, while the null-space projection prevents it from overwriting the agent's behavior pattern. In addition, IFEval remains nearly unchanged, suggesting that M2A does not noticeably degrade general instruction-following ability. Overall, these results show that M2A enables a synergistic enhancement of mathematical and agentic reasoning.

Ablation Study of M2A We ablate the three core components of M2A on SWE-Bench Verified. As shown in Figure 1 (b), the full method achieves the best performance, while removing any component weakens the gain. Without Merge Coefficient Calibration (w/o α), the merged model is no longer properly calibrated to the agent model, making the merge less effective. Without the Similarity-aware Layer Mask (w/o Mask), M2A cannot filter layers where the agent and reasoning task vectors are less compatible, which also reduces performance. The most significant degradation comes from removing null-space projection (w/o Null). In this case, the model retains only a small improvement over the Base Agent, indicating that naive reasoning injection can interfere with the agent-critical behavior subspace. This validates the central mechanism of M2A: effective reasoning transfer requires preserving the agent-critical behavior, rather than simply increasing reasoning intensity.

4.4 Merge Strength as a Control Knob for Reasoning Behavior

Merge strength β controls the intensity of the projected reasoning update injected into the agent model, and provides M2A with a **control knob** for **shaping merged model's behavior pattern**. As shown in Figure 2 (b), the average reasoning length per step grows approximately monotonically with β, and becomes nearly linear in the high-performing region. This indicates that a simple parameter-space coefficient can be translated into predictable behavior-space control over reasoning intensity. Meanwhile, the stable high-performing region around medium merge strength reduces the burden of manual hyperparameter search. We use $\beta = 0.8$ and $\beta = 1.2$ as empirical transition points, dividing the sweep into three behavior regimes. This controllability distinguishes M2A from standard merging baselines. In Appendix A.3, we show that other model merging baselines do not provide a stable behavior-control interface: their reasoning length changes nonlinearly and can collapse abruptly.

In the *Low Reasoning* regime, the injected reasoning signal is weak. The model largely preserves the behavior pattern of the original agent, but its reasoning remains weak. As a result, it tends to act with limited analysis and has insufficient capacity to build a reliable understanding of complex software issues before intervention.

As β enters the *Medium Reasoning* regime, the model exhibits **the most effective behavior pattern**. It gains stronger internal thinking while preserving the agent's ability to interact with the environment. This regime does not simply increase reasoning length; it shifts the agent toward more evidence-aware decision making. The model can spend more computation on understanding the issue, collecting

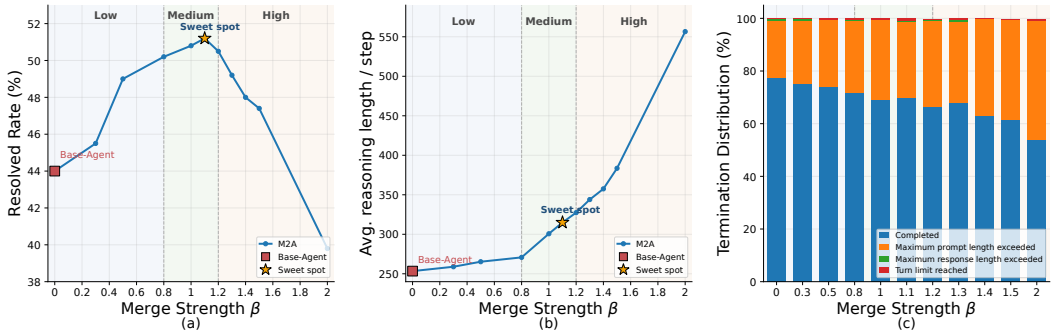

Figure 2: Effect of merge strength β on merged model behavior. (a) SWE-Bench Verified performance, (b) Avg. reasoning length per step, and (c) Termination Reason Distribution.

relevant context, and planning edits, without drifting away from the multi-turn behavior pattern. We provide a detailed analysis of this phenomenon in the next subsection. Importantly, performance remains **consistently high** across this interval, showing that M2A exposes **a robust behavior-control** region rather than relying on a fragile coefficient.

In the *High Reasoning* regime, further increasing β continues to lengthen reasoning traces but no longer improves performance. The model becomes overly dominated by internal thinking: it delays interaction, consumes more context, and is more likely to exceed its effective context-handling capacity. This is consistent with Figure 2(c), where high-β models show more failures caused by maximum prompt length exceeded.

4.5 Trajectory-Level Analysis: From Trial-and-Error Editing to Evidence-Grounded Action

To understand how M2A improves agent performance, we analyze trajectory-level behavior in Table 2. We observe two consistent shifts: (i) the agent reasons more at decision-critical steps (edit and late-stage synthesis), and (ii) it performs more search and inspection before editing, while issuing fewer and more concentrated edits overall. Together, these shifts suggest that M2A changes when and how the agent commits to an action — gathering more evidence before editing rather than probing through repeated edits. We refer to this as a shift from trial-and-error editing (frequent, exploratory edits with limited prior inspection) to evidence-grounded action (fewer edits preceded by more search and reasoning).

Table 2: Trajectory-level changes in reasoning and action behavior after applying M2A. Relative change is computed as $(\text{M2A} - \text{Base})/\text{Base}$.

Metric	Definition	Base Agent	M2A	Rel. Change
Resolved Rate ↑	Fraction of issues resolved.	44.0%	**51.2%**	**+16.4%**
Reasoning at edit ↑	Reasoning length tokens at edit steps.	305.3	**401.4**	**+31.5%**
Late-stage synthesis ↑	Mean reasoning tokens near trajectory end.	189.6	**230.7**	**+21.7%**
Search before first edit ↑	Search actions before the first edit.	19.5	**25.3**	**+30.0%**
Inspect before first edit ↑	File inspections before the first edit.	10.4	**12.3**	**+18.2%**
Edit count ↓	Edit operations per trajectory.	20.1	**16.7**	**-17.0%**
Unique edit files ↓	Distinct files touched by edits.	6.3	**6.0**	**-5.0%**

Regarding internal thinking, M2A strengthens reasoning at decision-critical stages. As shown in Table 2, the merged model reasons more at edit steps and shows stronger late-stage synthesis before finalizing a solution. This indicates that the additional reasoning is not merely front-loaded at the beginning of a trajectory, but remains active when the agent decides what to modify and how to complete the task.

From the perspective of external action, M2A yields actions that are both highly selective and precisely targeted. Before the first edit, the agent performs more search and inspection, suggesting that it gathers more evidence before intervening. Meanwhile, it uses fewer edit operations and touches

slightly fewer files. This suggests that M2A effectively mitigates blind trial-and-error behavior and encourages the agent to localize the problem before acting.

4.6 Generalization Across Models, Scaffolds, and Tasks

To evaluate the generalizability of M2A, we extend our experiments beyond the original Qwen3-8B and OpenHands setting and compare M2A against two representative model-merging baselines. The results are summarized in Table 3. Complete results with all baselines are provided in Appendix A.5.

Generalization across models and scaffolds. On Llama-3.1-8B [9], M2A improves the SWE-Bench Verified score from 17.6% to 23.0%, outperforming baselines, while maintaining a comparable number of interaction steps. Without scaffold-specific retraining or recalibration, M2A consistently achieves the highest score 46.4% and 49.8% on Mini-SWE-Agent [47] and R2E [16], respectively. These results demonstrate that M2A effectively generalizes across different model families and scaffolds.

Generalization beyond coding agents. On GAIA [25], M2A improves the score from 73.33% to 78.79%, substantially outperforming baselines. Notably, M2A achieves this improvement with a moderate increase in reasoning length, compared with the substantially longer reasoning traces produced by the merging baselines. This suggests that M2A's benefits extend beyond coding agents to general agent domain tasks.

Table 3: Generalization across model families, agent scaffolds, and tasks. SWE denotes SWE-Bench Verified.

Method	Score (%) ↑	Avg. Reasoning Length	Avg. Step
Llama-3.1-8B / OpenHands / SWE			
Agent-8B	17.6	207.79	67.66
Task Arithmetic [14]	21.0	483.38	43.73
RAIN-Merging [13]	20.4	242.45	70.00
M2A	**23.0**	228.76	70.70
Qwen3-8B / Mini-SWE-Agent / SWE			
Agent-8B	42.6	179.08	168.00
Task Arithmetic [14]	43.5	866.42	80.79
RAIN-Merging [13]	35.8	179.40	189.56
M2A	**46.4**	294.97	169.99
Qwen3-8B / R2E / SWE			
Agent-8B	46.6	208.93	149.27
Task Arithmetic [14]	48.7	847.47	70.70
RAIN-Merging [13]	46.8	209.30	151.70
M2A	**49.8**	277.76	166.33
Qwen3-8B / GAIA			
Agent-8B	73.33	850	13.65
Task Arithmetic [14]	66.1	2759.1	11.04
RAIN-Merging [13]	69.1	2007.1	14.45
M2A	**78.79**	959	19.22

5 Conclusion

We introduced M2A, a novel paradigm for synergizing mathematical and agentic reasoning through behavior-preserving null-space model merging. By protecting agent-critical behavior while injecting mathematical reasoning, M2A avoids the data-level mismatch of Multi-task SFT and the behavior pattern mismatch of naive merging. Experiments on SWE-Bench Verified show that M2A improves Qwen3-8B from 44.0% to 51.2% without additional training, while also improving mathematical reasoning and enabling controllable reasoning behavior. And experiments demonstrate that M2A generalizes across model families, agent scaffolds, and task domains. These findings open a new direction for synergizing reasoning capabilities across fundamentally different behavior patterns.

Acknowledgments and Disclosure of Funding

Funding Statement. This work was conducted during Junjian Wang's internship with the Foundation Model Team at Li Auto Inc. Financial support and computational resources for this research were provided by Li Auto Inc. This work was supported in part by the National Natural Science Foundation of China (No. 62276260).

Competing Interests. The authors declare no competing interests.

References

[1] Anthropic. Introducing claude opus 4.6. `https://www.anthropic.com/news/claude-opus-4-6`, February 2026. Accessed: 2026-04-25.

[2] Haoyue Bai, Yiyou Sun, Wenjie Hu, Shi Qiu, Maggie Ziyu Huan, Peiyang Song, Robert Nowak, and Dawn Song. How and why llms generalize: A fine-grained analysis of llm reasoning from cognitive behaviors to low-level patterns. *arXiv preprint arXiv:2512.24063*, 2025.

[3] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

[4] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[5] Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim K Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. *Advances in Neural Information Processing Systems*, 37:84746–84776, 2024.

[6] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

[7] Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, 2024.

[8] Google DeepMind. Gemini 3.1 pro. `https://deepmind.google/models/gemini/pro/`, 2026. Accessed: 2026-04-25.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[12] Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

[13] Zhehao Huang, Yuhang Liu, Baijiong Lin, Yixin Lou, Zhengbao He, Hanling Tian, Tao Li, and Xiaolin Huang. Rain-merging: A gradient-free method to enhance instruction following in large reasoning models with preserved thinking format. *arXiv preprint arXiv:2602.22538*, 2026.

[14] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

[15] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[16] Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*, 2025.

[17] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[19] Kwai-Klear. mini-swe-agent-plus: The 100-line AI agent that solves GitHub issues with text-edit tool. `https://github.com/Kwai-Klear/mini-swe-agent-plus`, 2025. GitHub Repository.

[20] Yu Li, Mingyang Yi, Xiuyu Li, Ju Fan, Fuxin Jiang, Binbin Chen, Peng Li, Jie Song, and Tieying Zhang. Reasoning and tool-use compete in agentic rl: From quantifying interference to disentangled tuning. *arXiv preprint arXiv:2602.00994*, 2026.

[21] Shuquan Lian, Juncheng Liu, Yazhe Chen, Yuhong Chen, and Hui Li. Swe-agile: A software agent framework for efficiently managing dynamic reasoning context. *arXiv preprint arXiv:2604.11716*, 2026.

[22] M. Luo, N. Jain, J. Singh, S. Tan, A. Patel, Q. Wu, A. Ariyak, C. Cai, T. Venkat, S. Zhu, B. Athiwaratkun, M. Roongta, C. Zhang, L. E. Li, R. A. Popa, K. Sen, and I. Stoica. DeepSWE: Training a fully open-sourced, state-of-the-art coding agent by scaling RL. `https://www.together.ai/blog/deepswe`, July 2025. Together AI Blog post.

[23] Qianli Ma, Dongrui Liu, Qian Chen, Linfeng Zhang, and Jing Shao. Led-merging: Mitigating safety-utility conflicts in model merging with location-election-disjoint. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21749–21767, 2025.

[24] Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

[25] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pages 9025–9049, 2024.

[26] OpenAI. Introducing gpt-5.2. `https://openai.com/index/introducing-gpt-5-2/`, December 2025. Accessed: 2026-04-25.

[27] OpenHands Team. OpenHands critic 32b exp 20250417. `https://huggingface.co/OpenHands/openhands-critic-32b-exp-20250417`, 2025. Accessed: 2025-12-08.

[28] Bo Pang, Deqian Kong, Silvio Savarese, Caiming Xiong, and Yingbo Zhou. Reasoning curriculum: Bootstrapping broad llm reasoning from math. *arXiv preprint arXiv:2510.26143*, 2025.

[29] Biqing Qi, Fangyuan Li, Zhen Wang, Junqi Gao, Dong Li, Peng Ye, and Bowen Zhou. Less is more: Efficient model merging with binary task switch. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15265–15274, 2025.

[30] Shyam Sundhar Ramesh, Xiaotong Ji, Matthieu Zimmer, Sangwoong Yoon, Zhiyong Wang, Haitham Bou Ammar, Aurelien Lucchi, and Ilija Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks. *arXiv preprint arXiv:2602.05547*, 2026.

[31] Ethan Shen, Danny Tormoen, Saurabh Shah, Ali Farhadi, and Tim Dettmers. Sera: Soft-verified efficient repository agents. *arXiv preprint arXiv:2601.20789*, 2026.

[32] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[33] Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.

[34] Shuang Sun, Huatong Song, Lisheng Huang, Jinhao Jiang, Ran Le, Zhihao Lv, Zongchao Chen, Yiwen Hu, Wenyang Luo, Wayne Xin Zhao, et al. Swe-world: Building software engineering agents in docker-free environments. *arXiv preprint arXiv:2602.03419*, 2026.

[35] Chaofan Tao, Jierun Chen, Yuxin Jiang, Kaiqi Kou, Shaowei Wang, Ruoyu Wang, Xiaohui Li, Sidi Yang, Yiming Du, Jianbo Dai, et al. Swe-lego: Pushing the limits of supervised fine-tuning for software issue resolving. *arXiv preprint arXiv:2601.01426*, 2026.

[36] Haoqing Wang, Xiang Long, Ziheng Li, Yilong Xu, Tingguang Li, and Yehui Tang. To mix or to merge: Toward multi-domain reinforcement learning for large language models. *arXiv preprint arXiv:2602.12566*, 2026.

[37] Haoran Wang, Zhenyu Hou, Yao Wei, Jie Tang, and Yuxiao Dong. Swe-dev: Building software engineering agents with training and inference scaling. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3742–3761, 2025.

[38] Junhao Wang, Daoguang Zan, Shulin Xin, Siyao Liu, Yurong Wu, and Kai Shen. Swe-mirror: Scaling issue-resolving datasets by mirroring issues across repositories. *arXiv preprint arXiv:2509.08724*, 2025.

[39] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[41] Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

[42] 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*, pages 23965–23998. PMLR, 2022.

[43] 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.

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

[45] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

[46] 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.

[47] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[48] John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

[49] Sihan Yang, Kexuan Shi, and Weiyang Liu. Orthogonal model merging. *arXiv preprint arXiv:2602.05943*, 2026.

[50] Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, Yingwei Ma, et al. Kimi-dev: Agentless training as skill prior for swe-agents. *arXiv preprint arXiv:2509.23045*, 2025.

[51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[52] 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 *Forty-first International Conference on Machine Learning*, 2024.

[53] Haobo Zhang and Jiayu Zhou. Unraveling lora interference: Orthogonal subspaces for robust model merging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26459–26472, 2025.

[54] Jiale Zhao, Guoxin Chen, Fanzhe Meng, Minghao Li, Jie Chen, Hui Xu, Yongshuai Sun, Wayne Xin Zhao, Ruihua Song, Yuan Zhang, et al. Immersion in the github universe: Scaling coding agents to mastery. *arXiv preprint arXiv:2602.09892*, 2026.

[55] Shenghe Zheng, Hongzhi Wang, Chenyu Huang, Xiaohui Wang, Tao Chen, Jiayuan Fan, Shuyue Hu, and Peng Ye. Decouple and orthogonalize: A data-free framework for lora merging. *arXiv preprint arXiv:2505.15875*, 2025.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We elucidate our contributions in the Abstract and Introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix E discusses the main limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: Our study is mainly empirical exploration.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We report our detailed experiment settings in Appendix A.1.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The current submission cites the public datasets, models, and evaluation frameworks used in the experiments, and provides detailed experimental and implementation descriptions. We plan to open-source the code in the future.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We report our detailed experiment settings in Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report the average value and standard deviation in our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We discuss computation cost in Appendix C.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read this code.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer:[Yes]

    Justification: We properly credited the creators or original owners of assets (e.g., code, data, models), used in the paper and conformed the license and terms.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLM is used only for writing.

Guidelines:

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

Appendix

A Additional Experimental Details

A.1 SWE-Bench Verified Evaluation Setup

Evaluation Dataset. We primarily evaluate agentic reasoning on SWE-Bench Verified [17], a human-validated subset of SWE-Bench consisting of 500 solvable instances curated from real-world GitHub issues. The benchmark spans 12 open-source Python repositories and is designed to assess the end-to-end issue resolution capability of LLM-based agents. Each instance provides a repository state and an issue description, and the agent is required to inspect the codebase, interact with the environment, modify relevant files, and submit a patch. A task is considered successfully resolved only if the submitted patch passes the official evaluation tests.

Evaluation Metric. We report the resolved rate as Avg@3 on SWE-Bench Verified. For each model, we run three independent trajectories under the same evaluation configuration and report the average resolved rate across the three runs. We also report Avg. reasoning length, computed as the average number of tokens in the model-generated chain-of-thought content, i.e., the `<think>` $\cdots$ `</think>` block, over all steps.

Evaluation Environment. We use a self-hosted environment based on OpenHands. Specifically, all models are evaluated with the openhands scaffold and the Docker backend. **To prevent reward hacking** through repository history, we remove historical git logs from the evaluation repositories before running the agent. This ensures that the model cannot exploit commands such as inspecting previous commits to obtain solution-relevant information, and must instead rely on code understanding, environment interaction, and patch generation.

During SWE-Bench Verified evaluation, we use 8 $\times$ H200 GPUs. At inference time, YaRN RoPE scaling is set to 8.0, and the maximum context length is set to 262k tokens. The detailed configuration is shown in Table 4.

Table 4: **SWE-Bench Verified evaluation configuration.** All models are evaluated with the same self-hosted OpenHands scaffold and Docker backend.

Item	Configuration
Benchmark	SWE-Bench Verified
Metric	resolved rate (Avg@3)
Scaffold	`openhands`
Environment	Self-hosted environment
Backend	`docker`
Git history	Removed before evaluation
Evaluation GPUs	8 $\times$ H200
RoPE scaling at inference	YaRN, factor = 8.0
Maximum context length	262144
Maximum steps	400
Step timeout	600 s
Trajectory timeout	14400 s
Reward timeout	1200 s
Maximum prompt length	262144
Maximum response length	81920
Temperature	0.7
Top-p	1.0
Tool-call format	`native`

A.2 SFT Training Details

Table 5 reports the supervised fine-tuning settings for Agent-8B and Multi-Task-8B. Both models are trained with `ms-swift` from the same base model, `Qwen3-8B-Base`. To better align the training format with the ReAct-style think–act pattern, we process code-agent trajectories into an explicit

reasoning-and-action format: <think> reasoning and response </think> + <tool_call> ··· </tool_call>. For long-context training, we use YaRN RoPE scaling with factor=4.0 and type=yarn.

Table 5: **SFT training settings.** Agent-8B is trained only on code-agent trajectories, while Multi-Task-8B is trained on a mixture of code-agent and mathematical reasoning trajectories.

Setting	Agent-8B	Multi-Task-8B
Base model	Qwen3-8B-Base	Qwen3-8B-Base
Training data	30k code-agent trajectories from Step-3.5-Flash-SFT[1]	30k code-agent trajectories and 30k mathematical reasoning trajectories from Step-3.5-Flash-SFT[1]
Training framework	ms-swift[2]	ms-swift[2]
Optimizer	Adam	Adam
Learning rate	5×10^{-5}	5×10^{-5}
Epochs	2	3
Global batch size	64	64
Cutoff length	131072	131072
RoPE scaling	YaRN, factor = 4.0, type = yarn	YaRN, factor = 4.0, type = yarn

[1] https://huggingface.co/datasets/stepfun-ai/Step-3.5-Flash-SFT

[2] https://github.com/modelscope/ms-swift.git

We verify that all training data used in this work have no instance-level overlap with SWE-Bench Verified. Specifically, the code-agent trajectories used to train Agent-8B and Multi-Task-8B do not contain any of the 500 SWE-Bench Verified evaluation instances. Therefore, the reported SWE-Bench Verified results are not affected by training-data leakage from the evaluation set.

A.2.1 Epoch Sweep for Multi-Task SFT

To rule out the possibility that the weak performance of Multi-Task-8B is simply caused by an under-tuned training schedule, we conduct an epoch sweep while keeping the training data mixture, base model, training framework, and other optimization settings unchanged. As shown in Table 6, increasing the number of epochs does not lead to a consistent improvement on SWE-Bench Verified. Multi-Task-8B obtains a resolved rate of 40.0 at 2 epochs, 41.1 at 3 epochs, 40.1 at 4 epochs, and 39.2 at 5 epochs. These results indicate that the limitation of multi-task SFT is not insufficient training, but rather the data-level mismatch introduced by jointly optimizing heterogeneous mathematical reasoning and multi-turn agent trajectories.

Table 6: **Epoch sweep for Multi-Task SFT on SWE-Bench Verified.** We keep the data mixture and optimization settings fixed, and only vary the number of training epochs. Increasing training epochs does not effectively improve agent performance.

Training Epochs	Resolved Rate
2	40.0
3	41.1
4	40.1
5	39.2

A.3 Merging Hyperparameters

For all model-merging baselines, the source models are fixed to Agent-8B and Reasoning-8B. To ensure a fair comparison, we conduct a merge-coefficient sweep for each baseline method and report the best-performing sweet spot in the main results. Thus, the baseline numbers in Table 1 are not obtained from arbitrary default coefficients, but from the strongest setting found under the same SWE-Bench Verified evaluation protocol.

Figures 3–6 show the merge-coefficient sweeps for Task Arithmetic, SLERP, TIES-Merging, and DARE. A consistent pattern is that these baselines are highly sensitive to the merge coefficient α. When α is small, the injected reasoning signal is weak and the model remains close to the original

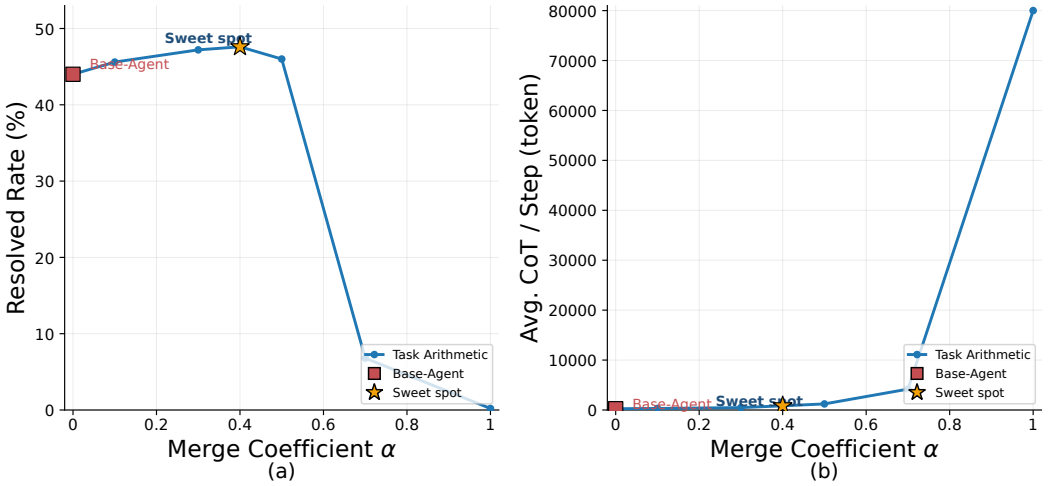

Figure 3: The effect of merge coefficient for Task Arithmetic. (a) SWE-Bench Verified performance, (b) Avg.CoT length per step

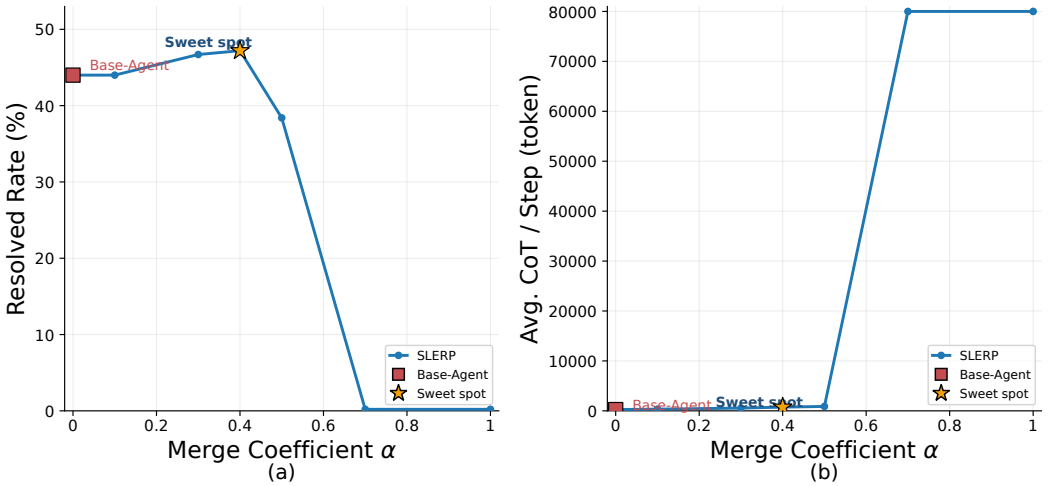

Figure 4: The effect of merge coefficient for SLERP. (a) SWE-Bench Verified performance, (b) Avg.CoT length per step

Agent-8B. As α increases to a narrow sweet spot, Task Arithmetic and SLERP can obtain moderate improvements. However, further increasing α quickly destabilizes the agent: the model becomes dominated by the single-turn mathematical reasoning pattern, produces extremely long reasoning traces, and often reaches the maximum response-length limit. This causes the agent to stop interacting effectively with the environment, leading to a sharp collapse in SWE-Bench Verified performance.

This phenomenon highlights a key limitation of standard model merging. Although the merge coefficient α nominally controls the amount of reasoning-model update being added, it does not provide reliable control over the resulting behavior pattern. The average CoT length of the merged model does not scale smoothly or linearly with α; instead, it often stays close to the base agent in the low-coefficient region and then abruptly jumps to near the response-length limit. Such discontinuous behavior makes baseline merging difficult to tune and easy to collapse, especially in agentic settings where the model must repeatedly alternate between thinking, acting, and observing.

In contrast, M2A exposes the global merge strength β as an effective behavior-control knob. As shown in Figure 2, increasing β leads to an approximately monotonic, and nearly linear in the high-performing region, growth of Avg. CoT per step. This means that M2A can translate a

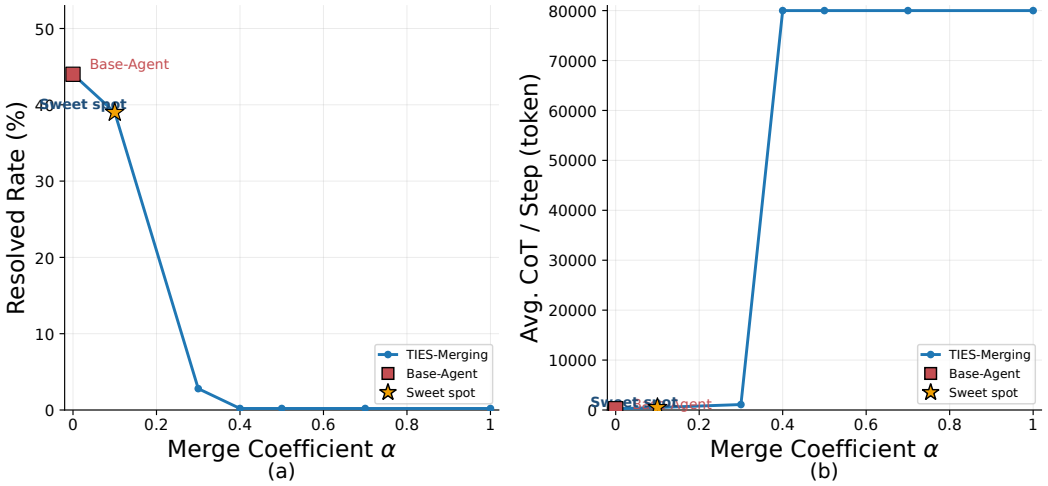

Figure 5: The effect of merge coefficient for TIES-Merging. (a) SWE-Bench Verified performance, (b) Avg.CoT length per step

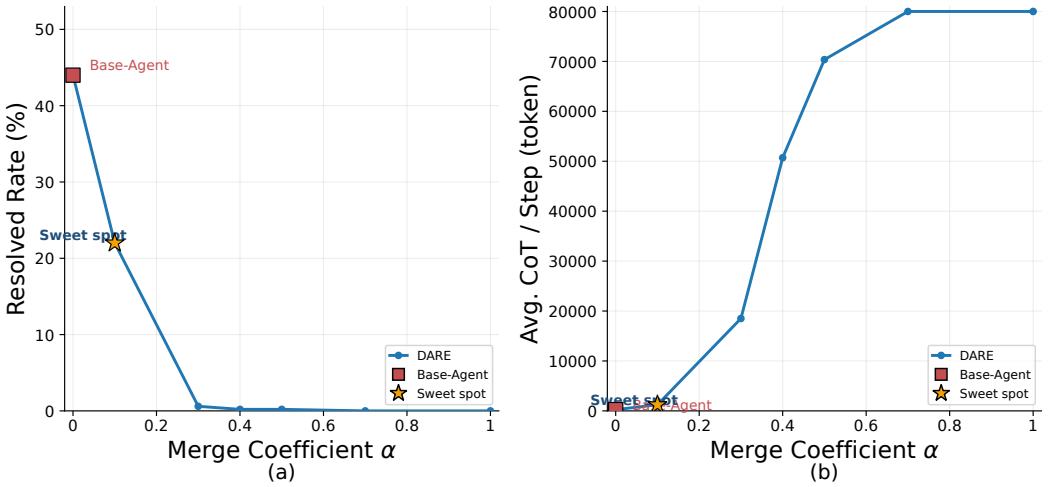

Figure 6: The effect of merge coefficient for DARE. (a) SWE-Bench Verified performance, (b) Avg.CoT length per step

simple parameter-space coefficient into predictable behavior-space control over reasoning intensity. Moreover, M2A has a stable medium-reasoning region where performance remains consistently strong, reducing the burden of manual hyperparameter tuning. The comparison between Figures 2 and 3–6 therefore further demonstrates the advantage of M2A: it does not merely find a better merge coefficient, but changes the merge coefficient from a fragile tuning variable into a controllable interface for shaping post-merge agent behavior.

For RAIN-Merging, we follow the parameter setting of the original method. Specifically, RAIN-Merging does not use the same manually swept scalar coefficient as Task Arithmetic, SLERP, TIES-Merging, or DARE. Instead, its merge coefficients are **adaptively selected** by the instruction-attention guided merging coefficient mechanism proposed in the paper [13]. We keep this design unchanged to faithfully evaluate RAIN-Merging under its intended configuration.

A.4 Average interaction steps under different merge strengths.

We further analyze how merge strength β affects the number of agent–environment interaction steps. As shown in Figure 7, the average number of steps per trajectory remains **relatively stable** across the

sweep, especially in the Low and Medium Reasoning regimes. This indicates that increasing β mainly controls the depth of internal reasoning, rather than suppressing the agent's external interaction behavior. In the Medium Reasoning regime, where M2A achieves the best performance, the agent maintains a step count close to the Base Agent while producing longer and more useful reasoning traces. Therefore, the performance gain does not come from reducing interaction or prematurely terminating the trajectory, but from better coordinating reasoning with action.

When β enters the High Reasoning regime, the average step count begins to decrease mildly due to exceeding maximum prompt length. Nevertheless, the step-count variation is much smoother than the abrupt behavior collapse observed in other merging baselines, further showing that M2A preserves the think–act–observe loop while exposing β as a control knob for reasoning behavior.

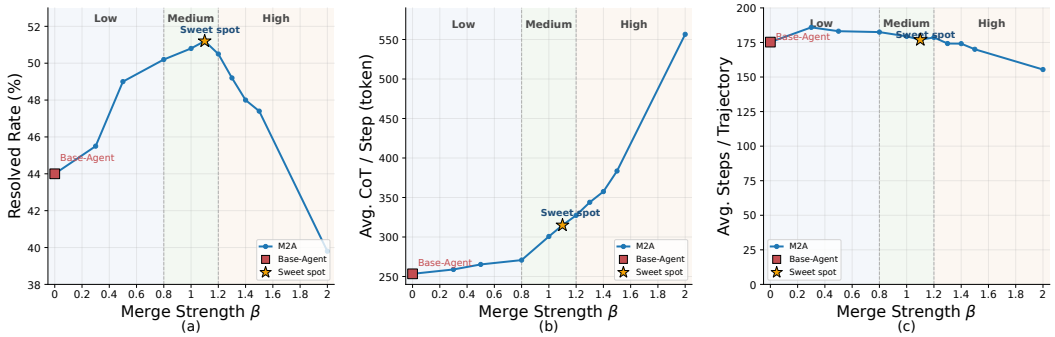

Figure 7: Effect of merge strength β on merged model behavior. (a) SWE-Bench Verified performance, (b) average reasoning length per step, and (c) average interaction steps per trajectory.

A.5 Additional Generalization Results

We provide comprehensive comparisons with additional model-merging baselines, including SLERP, DARE, and TIES-Merging, across different model families, agent scaffolds, and benchmarks. The complete results are reported in Table 7. M2A consistently achieves the highest performance across all evaluated settings, outperforming all five model-merging baselines. Most conventional merging methods substantially increase reasoning length while reducing interaction steps, often leading to degraded agent performance. In contrast, M2A achieves stronger reasoning capabilities while maintaining effective multi-turn interaction, further demonstrating its robustness across model families, agent scaffolds, and task domains.

B Calibration Set Construction

B.1 Agent Calibration Set

We construct the calibration set from on-policy agent trajectories on Scale-SWE [54]. We randomly sampled around 300 distinct tasks from Scale-SWE [54] and used our SFT-trained Agent-8B model to solve them, resulting in 300 successful or failed task trajectories. Each trajectory consists of multiple rounds of think–action–observation interactions. We sampled 100 individual interaction steps from these 300 trajectories and used them as calibration data. Thus, the calibration samples cover different tasks, both successful and failed trajectories, and different interaction step within each trajectory, thereby reducing potential sampling bias. The calibration set is used only to extract behavior-critical hidden states for M2A, not for fine-tuning or benchmark evaluation.

We verify that these 100 calibration trajectories have no instance-level overlap with SWE-Bench Verified. In particular, none of the calibration trajectories corresponds to any of the 500 SWE-Bench Verified evaluation instances. This separation ensures that the calibration stage does not expose the model to evaluation issues or solution trajectories from the test benchmark.

Each calibration example is organized into three fields: `prompt`, `reasoning`, and `response`. The `prompt` contains the system prompt and the multi-turn conversation history before the current action. The `reasoning` field contains the model's current-step internal thinking process, enclosed

Table 7: Complete generalization results across model families, agent scaffolds, and tasks. SWE denotes SWE-Bench Verified. The best score in each setting is highlighted in bold.

Method	Score (%) ↑	Avg. Reasoning Length	Avg. Step
Llama-3.1-8B / OpenHands / SWE			
Agent-8B	17.6	207.79	67.66
Task Arithmetic	21.0	483.38	43.73
TIES-Merging	20.6	257.36	53.10
DARE	12.6	815.49	30.70
SLERP	21.8	250.68	61.70
RAIN-Merging	20.4	242.45	70.00
M2A	**23.0**	228.76	70.70
Qwen3-8B / Mini-SWE-Agent / SWE			
Agent-8B	42.6	179.08	168.00
Task Arithmetic	43.5	866.42	80.79
TIES-Merging	38.6	420.12	92.25
DARE	23.8	1332.48	67.00
SLERP	33.8	707.38	83.25
RAIN-Merging	35.8	179.40	189.56
M2A	**46.4**	294.97	169.99
Qwen3-8B / R2E / SWE			
Agent-8B	46.6	208.93	149.27
Task Arithmetic	48.7	847.47	70.70
TIES-Merging	47.0	513.47	87.10
DARE	34.6	1130.82	58.00
SLERP	46.8	715.84	76.00
RAIN-Merging	46.8	209.30	151.70
M2A	**49.8**	277.76	166.33
Qwen3-8B / GAIA			
Agent-8B	73.33	850.00	13.65
Task Arithmetic	66.10	2759.10	11.04
TIES-Merging	66.67	2535.51	11.99
DARE	40.61	2571.79	9.51
SLERP	26.67	1891.57	11.58
RAIN-Merging	69.10	2007.10	14.45
M2A	**78.79**	959.00	19.22

by `<think>` and `</think>`. The `response` field contains the model's current-step external action, usually a tool call. Figure 8 gives a simplified example of the trajectory format.

B.2 Behavior Marker Set

Each calibration example is first formatted with the model's chat template and then represented as the concatenation of the prompt, reasoning field, and response field. We locate behavior-transition markers in the tokenized sequence using character-to-token offset mapping. The marker set is not tied to a specific serialization format; instead, it is instantiated according to the tool-call format used by the agent trajectory. For XML-style tool calls, we use

$$\mathcal{M}_{\text{xml}} = \{\texttt{<think>}, \texttt{</think>}, \texttt{<function=}, \texttt{</function>}\}, \tag{14}$$

while for native tool-call formats, we use

$$\mathcal{M}_{\text{native}} = \{\texttt{<think>}, \texttt{</think>}, \texttt{<tool_call>}, \texttt{</tool_call>}\}. \tag{15}$$

When mixed formats are present, M2A uses the union of all supported markers, $\mathcal{M} = \mathcal{M}_{\text{xml}} \cup \mathcal{M}_{\text{native}}$.

For a start marker at token position t, we collect the local neighborhood $[t, t + r]$; for an end marker, we collect $[t - r, t]$. In the main experiments, we use $r = 5$. This neighborhood construction captures the local representation changes when the model enters reasoning, exits reasoning, starts a tool call, or finishes an external action block.

B.3 Calibration Set Size Ablation

We further study the effect of calibration set size on M2A. All experiments use the same model, merge configuration, evaluation scaffold, and SWE-Bench Verified evaluation protocol; we only vary the

```
{
  "prompt": "<|im_start|>system
You are OpenHands agent, a helpful AI assistant that can interact
with a computer to solve tasks.
...
<|im_start|>user
Now consider the following Github issue:
...
<|im_start|>assistant",

  "reasoning": "<think>
I need to understand the issue, inspect the repository, locate the
relevant implementation, and decide which file should be modified.
</think>",

  "response": "<tool_call>
{\"name\": \"execute_bash\",
 \"arguments\": {\"command\": \"find /workspace/repo -type f | head\"}}
</tool_call>"
}
```

Figure 8: **Simplified example of a code-agent calibration trajectory.** The prompt contains the system prompt and conversation history; the reasoning field contains the current-step internal thinking; and the response field contains the external tool call.

number of calibration trajectories used to construct the agent-critical behavior subspace. As shown in Table 8, using only 50 trajectories already improves over the base agent, but the performance is lower than the default setting. Increasing the calibration set size to 100 trajectories yields the best resolved rate of 51.2%. Further increasing the calibration set to 200 or 300 trajectories does not bring additional improvement, while requiring more calibration computation.

These results indicate that 100 trajectories are sufficient to capture the key behavior-critical directions for preserving the agent's think–act–observe pattern. Therefore, we use 100 calibration trajectories in the main experiments, which provides the best trade-off between performance and efficiency.

Table 8: Calibration set size ablation on SWE-Bench Verified.

Calibration Set Size	**Resolved Rate (%)**
50	49.1
100	**51.2**
200	51.0
300	50.8

B.4 Calibration Window Radius Ablation

We further ablate the calibration window radius r, which determines how many neighboring tokens are collected around each behavior transition marker when constructing the agent-critical behavior subspace. This experiment examines whether M2A is sensitive to the local window used for behavior calibration. A very small window may miss the transition context around reasoning and tool-call markers, while an overly large window may include task-specific semantic tokens that are less directly related to the agent's think–act–observe behavior pattern.

All experiments use the same model, calibration set size, merge configuration, and SWE-Bench Verified evaluation protocol; we only vary the radius r. As shown in Table 9, M2A is relatively robust to this hyperparameter. The resolved rate remains consistently high across different radius choices: $r = 3$ achieves 51.0%, $r = 5$ achieves 51.2%, and $r = 10$ still obtains 49.5%. All settings substantially outperform the base agent, indicating that the behavior-critical subspace can be reliably estimated from local marker neighborhoods and does not depend on a fragile radius choice.

At the same time, a moderate radius performs best. The slightly lower performance at $r = 3$ suggests that a smaller window may not fully capture the local transition context, while the drop at $r = 10$

suggests that larger windows may introduce more task-specific or irrelevant trajectory tokens into the protected subspace. Therefore, we use $r = 5$ in the main experiments, as it provides the best performance while preserving robustness across nearby radius choices.

Table 9: **Calibration window radius ablation on SWE-Bench Verified.**

Window Radius r	**Resolved Rate (%)**
3	51.0
5	**51.2**
10	49.5

C Compute Resources

Table 10 reports the compute resources used in our experiments. M2A itself is training-free and does not require gradient updates, optimizer states, or rollout-based policy optimization. Its additional cost comes from forward-pass calibration and parameter-space merging.

Table 10: **Compute resources.** We report the hardware, peak memory usage, and Wall-clock Time used for SFT and M2A merging.

Stage	**Hardware**	**Peak Memory / GPU**	**Time(hours)**
Agent-8B SFT	32 × H200	140 GB	≈ 30
Multi-Task-8B SFT	32 × H200	140 GB	≈ 48
M2A calibration and merging	8 × H200	50 GB	≈ 2

D Implementation Details of M2A

Orthonormal basis for the agent-critical subspace. The main text defines the agent-critical behavior subspace as

$$S^{(l)}_{\text{agent}} = \text{span}(C^{(l)}), \qquad C^{(l)} = [h^{(l)}_1, h^{(l)}_2, \ldots, h^{(l)}_{n_l}] \in \mathbb{R}^{d \times n_l}, \tag{16}$$

where each $h^{(l)}_i$ is a hidden feature collected from the local neighborhood of a behavior-transition marker. Conceptually, an orthonormal basis $Q^{(l)}$ of this subspace can be obtained by a thin SVD:

$$C^{(l)} = U^{(l)} \Sigma^{(l)} V^{(l)\top}. \tag{17}$$

We keep the left singular vectors corresponding to non-negligible singular values:

$$Q^{(l)} = U^{(l)}, \tag{18}$$

In this explicit form, the null-space projector is

$$P^{(l)}_{\text{null}} = I - Q^{(l)} Q^{(l)\top}, \tag{19}$$

and the behavior-preserving reasoning update for a weight matrix whose input dimension matches $S^{(l)}_{\text{agent}}$ is:

$$\widetilde{\Delta W}^{(l)}_{\text{reason}} = \Delta W^{(l)}_{\text{reason}} P^{(l)}_{\text{null}}. \tag{20}$$

This yields

$$\widetilde{\Delta W}^{(l)}_{\text{reason}} h = 0, \qquad \forall h \in S^{(l)}_{\text{agent}}, \tag{21}$$

which is the no-perturbation condition used in the main text.

Matrix-free implementation used in experiments. Although the above SVD form is useful for presenting the geometry of M2A, our implementation does not explicitly materialize $Q^{(l)}$ or $P^{(l)}_{\text{null}}$. This is mainly for efficiency: in a transformer layer, the protected constraints must be applied separately to multiple parameter blocks, including W_Q, W_K, W_V, W_O, W_{gate}, W_{up}, and W_{down}, and for attention projections the update is further processed head-wise. In the default setting, we merge all these components, denoted as `qkvof`.

Instead, we implement the same null-space projection through block-wise linear constraint operators. For a mergeable block B in layer l, let

$$\delta_B = \text{vec}(\Delta W_B) \tag{22}$$

be the vectorized reasoning task update. We construct a linear operator A_B from the same marker-neighborhood calibration features. The protected-subspace condition is then written as:

$$A_B \widetilde{\delta}_B = 0. \tag{23}$$

The projected update is obtained by solving:

$$\widetilde{\delta}_B = \arg\min_z \|z - \delta_B\|_2^2 \quad \text{s.t.} \quad A_B z = 0. \tag{24}$$

The closed-form solution is:

$$\widetilde{\delta}_B = \delta_B - A_B^\top (A_B A_B^\top)^{-1} A_B \delta_B. \tag{25}$$

For numerical stability, we use the ridge-stabilized form:

$$\widetilde{\delta}_B = \delta_B - A_B^\top (A_B A_B^\top + \lambda I)^{-1} A_B \delta_B, \tag{26}$$

where $\lambda = 10^{-4}$ by default.

This matrix-free formulation is equivalent to the explicit Q-based projection when A_B spans the same protected feature subspace. In practice, A_B is built from sampled marker-neighborhood constraints, so it provides an efficient sampled approximation to the ideal projection $\Delta W^{(l)}_{\text{reason}}(I - Q^{(l)}Q^{(l)\top})$. The implementation only requires products with A_B and $A_B^\top$, avoiding the memory cost of constructing the full projector for every layer, head, and parameter block.

Block-wise projection details. The implementation applies the above projection block-wise. For attention projections, W_Q, W_K, and W_V are sliced by head rows, while W_O is sliced by head columns. For models with grouped-query attention, the key/value head corresponding to query head h is selected by $h \bmod H_{\text{kv}}$. For FFN layers, the same projection principle is applied to W_{gate}, W_{up}, and W_{down}; the down-projection update is internally transposed to match the solver layout and then transposed back before being added to the model.

Small constraint systems are solved by dense linear solvers. When dense solving is unstable or memory-expensive, we fall back to conjugate gradient on $(A_B A_B^\top + \lambda I)$ with a maximum of 100 iterations and tolerance 10^{-5}. Thus, the implementation follows the null-space projected merging objective in the main text, but realizes it through an efficient matrix-free solver rather than explicitly computing $Q^{(l)}$ for every block.

D.1 Implementation Details of Adaptive Layer-Wise Merging

Layer-wise task-vector aggregation. In the main text, $\Delta\theta^{(l)}_{\text{agent}}$ and $\Delta\theta^{(l)}_{\text{reason}}$ denote the layer-wise agent and reasoning task vectors. In implementation, each of them is represented as a collection of mergeable parameter blocks:

$$\Delta\theta^{(l)}_x = \left\{\Delta W^{(l)}_{x,p}\right\}_{p\in\mathcal{P}^{(l)}}, \qquad x \in \{\text{agent}, \text{reason}\}, \tag{27}$$

where

$$\Delta W^{(l)}_{\text{agent},p} = W^{(l)}_{\text{agent},p} - W^{(l)}_{\text{base},p}, \qquad \Delta W^{(l)}_{\text{reason},p} = W^{(l)}_{\text{reason},p} - W^{(l)}_{\text{base},p}. \tag{28}$$

Here $\mathcal{P}^{(l)}$ contains all enabled mergeable blocks in layer l, including the Q/K/V/O attention projections and the FFN gate/up/down projections. For attention modules, Q/K/V/O are computed head-wise; for FFN modules, the full projection matrices are used.

Merge coefficient calibration. The layer-wise Frobenius norm in Eq. (11) is computed by summing the squared norms over all enabled blocks:

$$\left\|\Delta\theta_x^{(l)}\right\|_F = \left(\sum_{p\in\mathcal{P}^{(l)}} \left\|\Delta W_{x,p}^{(l)}\right\|_F^2\right)^{1/2}, \qquad x \in \{\text{agent}, \text{reason}\}. \tag{29}$$

The calibrated coefficient is then computed as

$$\alpha_l = \beta \cdot \frac{\left\|\Delta\theta_{\text{agent}}^{(l)}\right\|_F}{\left\|\Delta\theta_{\text{reason}}^{(l)}\right\|_F + \epsilon}. \tag{30}$$

If the reasoning-vector norm is numerically close to zero, we set $\alpha_l = 0$. This is the quantity visualized in Figure 9(a): the reasoning task vector often has a larger magnitude than the agent task vector, and α_l rescales it to the layer-wise agent scale before merging.

Similarity-aware layer mask. In the main text, $\text{sim}^{(l)}$ denotes the cosine similarity between $\Delta\theta_{\text{agent}}^{(l)}$ and $\Delta\theta_{\text{reason}}^{(l)}$. In implementation, this similarity is computed by aggregating all enabled mergeable blocks in layer l:

$$\text{sim}^{(l)} = \frac{\sum_{p\in\mathcal{P}^{(l)}} \left\langle \Delta W_{\text{agent},p}^{(l)}, \Delta W_{\text{reason},p}^{(l)} \right\rangle_F}{\left\|\Delta\theta_{\text{agent}}^{(l)}\right\|_F \left\|\Delta\theta_{\text{reason}}^{(l)}\right\|_F + \epsilon}, \tag{31}$$

where $\mathcal{P}^{(l)}$ denotes the set of mergeable parameter blocks in layer l, and $\langle A, B\rangle_F = \text{Tr}(A^\top B)$ is the Frobenius inner product. The global similarity threshold is the mean similarity over all selected layers:

$$\bar{\text{sim}} = \frac{1}{|\mathcal{L}|}\sum_{l\in\mathcal{L}} \text{sim}^{(l)}, \tag{32}$$

where $\mathcal{L}$ is the set of selected layers. The binary layer mask is computed as

$$M_l = \mathbb{I}\left[\text{sim}^{(l)} > \bar{\text{sim}}\right]. \tag{33}$$

Layers with $M_l = 1$ are merged after null-space projection, while layers with $M_l = 0$ are skipped and kept identical to the agent model. As illustrated in Figure 9(b), this mask selects similarity-aligned layers for reasoning injection and protects low-similarity layers from potentially conflicting updates.

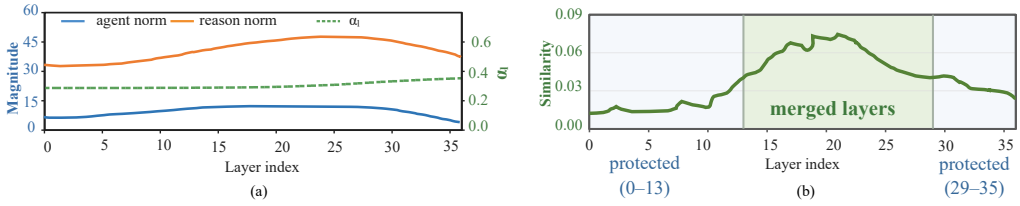

Figure 9: (a) α_l is computed from the ratio between the layer-wise agent and reasoning task-vector norms. (b) M_l is obtained by layer-wise similarities. Green regions are merged layers, while blue regions are protected layers.

D.2 Calibration Source Quality and Scaffold Distribution

For the mixed-scaffold setting, we constructed the calibration set using trajectories from Mini-SWE-Agent and R2E, performed merging with this calibration set, and evaluated the resulting model with OpenHands. Across all calibration sources, the performance differences are consistently small, indicating that M2A is not sensitive to the specific choice of trajectories, whether they are successful, failed, or from different scaffolds. Random sampling performs slightly better, because it provides greater trajectory diversity. Overall, these results support the robustness of M2A to both calibration-source quality and scaffold distribution.

Table 11: Calibration-source comparison on Qwen3-8B/OpenHands with SWE-Bench Verified. Mixed-scaffold calibration uses 50% Mini-SWE-Agent and 50% R2E trajectories; the resulting merged model is evaluated with OpenHands.

Calibration source	Score (%)	Avg.Reasoning Length	Avg.Step
Random sampling	**51.2**	327.40	178.00
Failed rollouts	50.6	299.29	165.77
Successful rollouts	50.2	291.53	162.59
Mixed scaffolds	50.0	302.90	166.67

D.3 Alternative Layer-Selection Rules

Table 12: Layer-selection ablation on Qwen3/OpenHands. Layer ranges use the reported indices and are not counts. The global-mean threshold is approximately 0.03.

Rule	Selected layers	Score (%)	Avg.Reasoning Length	Avg.Step
Global mean (≈ 0.03)	14–28	**51.2**	327.40	178.00
Top-5	18–22	46.2	247.49	159.49
Top-10	16–25	45.6	292.54	151.71
Fixed threshold (0.04)	15–25	49.6	283.15	165.34
Fixed threshold (0.05)	17–23	50.0	308.22	154.29

The global-mean similarity rule performs better than the tested top-(k) alternatives and does not require an additional hyperparameter. The global-mean rule outperforms Top-(5) and Top-(10) by 5.0 and 5.6 percentage points, respectively. It adapts automatically to the layer-similarity distribution and avoids introducing an additional (k) or manually tuned threshold.

Reproducibility and numerical details. All models are loaded in bfloat16, while projection computations are performed in float32 by default. The implementation supports assigning Q/K, V/O, and FFN projection computations to different devices. To improve robustness for long trajectories, calibration examples exceeding the maximum sequence length are skipped during constraint construction.

Chat template for multi-turn reasoning. Qwen3 models use a default chat template that compresses historical assistant reasoning to save context length. Specifically, for previous assistant turns before the latest user query, the default template removes the content inside `<think>...</think>` and only keeps the final assistant response. While this design is efficient for ordinary dialogue, it can be suboptimal for multi-turn agent evaluation, because the agent may need to condition on its previous reasoning traces when planning subsequent tool calls, interpreting observations, or revising earlier decisions.

To better evaluate the effect of reasoning integration in multi-turn agent settings, we modify the chat template to preserve historical reasoning traces. The key change is that, for every assistant message, if `reasoning_content` is available or can be parsed from `<think>...</think>`, we explicitly serialize it back into the conversation:

$$\texttt{assistant message} \Rightarrow \texttt{<think>}\,\texttt{reasoning_content}\,\texttt{</think>} + \texttt{final content}. \quad (34)$$

In other words, we remove the default logic that only preserves reasoning after the latest non-tool user query. The modified template therefore keeps the full multi-turn reasoning history, including previous thought traces, tool calls, and tool responses.

The tool-call format itself is unchanged. Function signatures are still provided inside `<tools>...</tools>`, tool invocations are serialized with `<tool_call>...</tool_call>`, and environment observations are returned inside `<tool_response>...</tool_response>`. Thus, the modification only affects whether historical reasoning traces are retained, not the tool-use protocol.

For fairness, all evaluated models use the same reasoning-preserving chat template. Therefore, the reported gains do not come from giving M2A a different conversation format. All models are evaluated under the same context-length limit, tool-call format, and trajectory construction protocol.

E Limitations

Our main controlled study focuses on Qwen3-8B with OpenHands and SWE-Bench Verified; additional experiments cover Llama-3.1-8B, Mini-SWE-Agent, R2E, and GAIA. These settings support transfer beyond the original configuration but do not establish universal preservation across arbitrary tool schemas or model scales. In particular, interaction counts may increase on longer-horizon tasks, and calibration still requires observable action/turn boundaries and access to model weights. Broader deep-research settings and sensitivity to implicit boundaries remain open directions.

F Broader Impact

This work studies a training-free method for synergizing mathematical and agentic reasoning in large language models. Its potential positive impact is to improve the capability and efficiency of software engineering agents without requiring additional costly SFT or RL, making stronger agentic reasoning more accessible to researchers and practitioners with limited computational resources. By preserving multi-turn interaction behavior while enhancing internal reasoning, M2A may help build coding agents that perform more reliable debugging, code repair, and repository-level understanding. At the same time, stronger coding agents may also introduce risks if deployed without proper safeguards. More capable software agents could be misused to automate the generation of insecure code, discover or exploit vulnerabilities, or make unintended modifications in real-world codebases. In addition, because agentic systems interact with external environments, failures may propagate beyond text generation and affect files, repositories, or execution environments. Therefore, practical deployment should include sandboxed execution, human oversight for consequential changes, access control, logging, and security-oriented evaluation before use in real-world software systems.

G Qualitative Trajectory Case Study

G.1 Case Study 1

To better illustrate the behavioral change induced by M2A, we present a concrete SWE-Bench Verified case study from the same issue instance, where Agent-8B fails while M2A-Agent-8B succeeds. Both agents are evaluated under the same OpenHands scaffold and are given the same repository-level issue. The issue requires fixing a Sphinx autodoc bug: when Google-style docstrings are parsed by Napoleon, the return description is represented as `:returns:`, but the autodoc type-hint merger only recognizes `:return:` when deciding whether to add `:rtype:` under `autodoc_typehints_description_target = "documented"`.

Trajectory comparison on `sphinx-doc/sphinx`: `autodoc_typehints_description_targ` with Napoleon

Agent-8B baseline: failed trajectory

Outcome: max prompt length exceeded

Pattern 1: broad exploration without a stable problem representation.

```
<think>
The issue is about autodoc_typehints_description_target
not working with Napoleon. Maybe the bug is in how
Napoleon processes docstrings, or how autodoc merges fields.
Let me inspect more files and tests.
</think>

<tool_call>
grep -n "documented" /testbed/tests/test_ext_autodoc.py
grep -rn "Returns:" /testbed/tests/roots/test-ext-autodoc/
grep -rn "def _process_docstring" /testbed/sphinx/ext/
</tool_call>
```

Pattern 2: repeated uncertainty and delayed action.

```
<think>
Maybe there is no test covering Napoleon.
Maybe the issue is that :rtype: is added but removed later.
Maybe the output differs because of configuration.
I am going in circles.
</think>
```

Pattern 3: context consumed before a reliable fix.

```
<tool_call>
create /testbed/reproduce_issue.py
run reproduction
inspect Napoleon processing
inspect autodoc typehints
search tests again
...
</tool_call>

termination_reason = max_prompt_length_exceeded
reward = 0
```

Interpretation. The baseline performs many exploratory actions but does not quickly compress the evidence into a precise root cause. Its trajectory is dominated by repeated search and hypothesis revision, causing the context to grow until the task fails.

M2A-Agent-8B: successful trajectory

Outcome: resolved

Pattern 1: evidence-grounded problem representation.

```
<think>
The bug is in documented-mode type-hint merging.
Napoleon converts Google-style "Returns:" into a
:returns: field, while autodoc checks only :return:
when deciding whether the return value is documented.
</think>

<tool_call>
grep -n "augment_descriptions_with_types" \
  /testbed/sphinx/ext/autodoc/typehints.py
</tool_call>
```

Pattern 2: fewer but more targeted external actions.

```
<edit>
--- a/sphinx/ext/autodoc/typehints.py
+++ b/sphinx/ext/autodoc/typehints.py

- elif parts[0] == 'return':
+ elif parts[0] in ('return', 'returns'):
      has_description.add('return')

- if name == 'return':
+ if name in ('return', 'returns'):
      continue
</edit>
```

Pattern 3: verification before finalization.

```
<tool_call>
python reproduce_issue.py
python -m pytest tests/test_ext_autodoc_configs.py \
  -k "description" -xvs
</tool_call>

Return type field FOUND.
1 passed.
termination_reason = env_done
reward = 1
```

Interpretation. M2A first builds a compact causal explanation of the failure, then converts it into a minimal code change and verifies the result. This trajectory shows stronger internal reasoning without suppressing the external think–act–observe loop.

Key behavioral difference

The baseline follows an **explore–revise–explore** pattern: it repeatedly searches related code paths and tests, but fails to stabilize the root cause before exhausting the prompt budget. In contrast, M2A follows an **understand–patch–verify** pattern: it identifies the semantic mismatch between `:return:` and `:returns:`, performs a localized patch, and validates the fix. This supports the trajectory-level finding that M2A shifts the agent from early trial-and-error editing toward evidence-grounded action.

Figure 10: Qualitative trajectory comparison. Agent-8B fails by repeatedly exploring related code paths without forming a stable root-cause representation. M2A-Agent-8B succeeds by first identifying the Napoleon–autodoc field mismatch, then applying a minimal patch and verifying it. The example illustrates how M2A strengthens internal reasoning while preserving effective external action.

G.2 Case Study 2

We provide another representative trajectory comparison on a `scikit-learn` issue. The issue concerns an inconsistency in `LogisticRegressionCV`: when probabilistic scorers such as `neg_log_loss` are used with `multi_class='multinomial'`, the helper estimator inside `_log_reg_scoring_path` should use the same multi-class mode as the outer `LogisticRegressionCV`. This case illustrates the behavioral difference between the base agent and

M2A. The base agent identifies the surface-level argument to add, but makes an incomplete edit that removes an existing constructor argument. In contrast, M2A first builds a more precise causal understanding of how scoring depends on `predict_proba`, and then translates this understanding into a minimal and semantics-preserving edit.

Agent-8B Baseline: Trial-and-Error Edit

Trajectory pattern. The base agent quickly identifies that `multi_class` is relevant, but does not fully preserve the semantics of the existing constructor call. Its external action is close to the suggested fix, yet it overwrites the existing `fit_intercept` argument.

Reasoning focus.

- Locates `_log_reg_scoring_path`.
- Recognizes that `multi_class` should be passed.
- Applies the edit without preserving all existing arguments.

Final patch.

```
- log_reg = LogisticRegression(fit_intercept=fit_intercept)
+ log_reg = LogisticRegression(multi_class=multi_class)
```

Key failure pattern. The agent performs a near-correct but incomplete edit: it adds the new argument by replacing the constructor call rather than extending it. As a result, the patch loses `fit_intercept=fit_intercept`, which changes the behavior of the helper estimator beyond the intended fix.

Outcome. Failed resolution.

M2A-Agent-8B: Evidence-Grounded Action

Trajectory pattern. M2A first reasons about the scoring path and the probability prediction mechanism before editing. It identifies that probabilistic scorers call `predict_proba`, whose behavior depends on the estimator's `multi_class` setting. This leads to a focused and semantics-preserving patch.

Reasoning focus.

- Traces `scoring(log_reg, X_test, y_test)`.
- Connects `neg_log_loss` to `predict_proba`.
- Infers that `multi_class` must be inherited by the helper estimator.
- Preserves the existing `fit_intercept` behavior.

Final patch.

```
- log_reg = LogisticRegression(fit_intercept=fit_intercept)
+ log_reg = LogisticRegression(fit_intercept=fit_intercept,
+                              multi_class=multi_class)
```

Key success pattern. The edit is not merely longer or more verbose reasoning. The additional reasoning is used to construct a better problem representation, which is then converted into a smaller and more faithful external action.

Outcome. Successful resolution.

Figure 11: **Trajectory comparison on a `scikit-learn` issue.** The base agent exhibits a trial-and-error pattern: it identifies the relevant parameter but applies an incomplete edit that removes the existing `fit_intercept` argument. M2A instead performs evidence-grounded action: it reasons through the scoring and probability-prediction path, then applies a minimal patch that preserves the original constructor semantics while adding `multi_class`. This case supports our observation that M2A shifts agent behavior from early, surface-level editing toward evidence-grounded action.