Reasoning and Tool-use Compete in Agentic RL: From Quantifying Interference to Disentangled Tuning

Anonymous ACL submission

Abstract

Agentic Reinforcement Learning (ARL) trains large language models to interleave reasoning with external tool execution to solve complex tasks. Most existing ARL methods train a single set of parameters to support both reasoning and tool-use behaviors, implicitly assuming that joint training leads to improved overall agent performance. Despite its widespread adoption, this assumption has rarely been examined empirically. In this paper, we systematically examine this assumption by introducing Capability Effect Attribution (CEA), which provides quantitative evidence of interference between reasoning and tool-use behaviors. Through an in-depth analysis, we show that these two capabilities often induce misaligned gradient directions, leading to training interference that undermines the effectiveness of joint optimization and challenges the prevailing ARL paradigm. To address this issue, we propose Disentangled Action–Reasoning Tuning (DART), a simple and efficient framework that explicitly decouples parameter updates for reasoning and tool use via separate low-rank adaptation modules. With this simple change alone, DART outperforms all joint-optimization baselines and approaches the 2-Agent upper bound across thirteen benchmarks on retrieval-augmented QA and NL2SQL, further supporting our finding of capability interference under shared optimization.

1 Introduction

Recent advances in Agentic Reinforcement Learning (ARL) for post-training (Ouyang et al., 2022; Bai et al., 2022; Li et al., 2026a) have substantially extended the capabilities of large language models (LLMs). Beyond text generation, modern LLMs can perform complex reasoning and interact with external tools to solve tasks such as information retrieval (Jin et al., 2025), computation (Mai et al., 2025), data analysis (Zhang et al., 2025a), and research workflows (Qiao et al., 2025).

The goal of ARL is to train models that reliably execute external tools while exhibiting strong reasoning abilities (Wu et al., 2024). Most existing ARL paradigms (Schick et al., 2023; Shao et al., 2024; Zeng et al., 2024; Zhang et al., 2025b) jointly optimize these two *heterogeneous capabilities* based on a *single ARL objective with shared model parameters.* This design implicitly assumes that tool execution and logical reasoning can be effectively accommodated within the same parameter space. Prior work has shown that optimization interference arises when training across distinct domains (Ye et al., 2026; Yuan et al., 2026; Wu et al., 2025a); however, whether heterogeneous capabilities *within* a single agentic domain also interfere with each other remains largely unexplored.

In this work, we directly test the shared parameter assumption through a controlled empirical analysis of the interaction between tool-use and reasoning. Specifically, we introduce **Capability Effect Attribution** (CEA), a diagnostic framework that decomposes an agent's performance into individual capability effects and pairwise interaction terms. By constructing six controlled model variants via gradient masking and hybrid inference, we solve for per-question interaction coefficients and reveal a significant **negative interaction** between reasoning and tool-use under joint optimization, indicating that shared-parameter training induces implicit competition.

To explain the root cause of this interference, we examine the optimization dynamics (Ren and Sutherland, 2025; Li et al., 2026b) by analyzing gradients from reasoning and tool-use tokens. We identify a clear gradient misalignment: the two types of gradients are nearly orthogonal, causing joint optimization over a shared backbone to update parameters in a **compromise direction** that is suboptimal for both capabilities.

Motivated by this finding, we propose **Disentangled Action-Reasoning Tuning (DART)**, a simple

yet effective framework designed to test whether eliminating gradient misalignment improves performance. DART freezes the pretrained backbone and routes reasoning and tool-use tokens to separate LoRA (Hu et al., 2022) adapters, so that each capability updates only its own parameters without affecting the other.

Experiments on seven QA and six NL2SQL benchmarks show that DART consistently outperforms joint-optimization baselines, confirming the generality of capability interference and the effectiveness of gradient disentanglement. Two experimental observations further support this conclusion: **(1)** DART surpasses all joint-optimization baselines, including multi-LoRA methods, showing that resolving gradient misalignment via hard token-level routing is essential. **(2)** Ablation studies attribute DART's gains to misalignment elimination rather than extra parameter capacity.

Our contributions are summarized as follows:

- For ARL training, we empirically identify negative interaction between tool-use and reasoning using Capability Effect Attribution (CEA), and trace this interference to gradient misalignments under joint optimization.
- To validate this finding, we propose DART, a simple yet effective framework that disentangles gradients for reasoning and tool-use via separate LoRA adapters.
- Extensive experiments on two tasks across thirteen benchmarks show that DART surpasses joint-optimization baselines, confirming that capability interference is a non-negligible bottleneck in ARL and that gradient disentanglement is an effective remedy.

2 Related Work

ARL with Tool-use. ARL research focuses on fine-tuning LLMs as autonomous agents that learn to invoke external tools through environment feedback, bridging the gap between reasoning and action without dense step-level supervision tuning (Schick et al., 2023). Recent advancements have optimized various components of this pipeline, including reward formulation to induce emergent behaviors (Qian et al., 2025; Peiyuan et al., 2024; Mai et al., 2025), policy refinement for precise action interleaving (Feng et al., 2025; Singh et al., 2025; Wei et al., 2025) , and large-scale trajectory synthesis (Dong et al., 2025; Li et al., 2025) for scalable training (Jiang et al., 2025). However, none of these works examine whether reasoning and tool-use interfere with each other under joint optimization, which is the central question of our study.

Multi-LoRA. Existing Multi-LoRA methods follow the MoE paradigm (Shazeer et al., 2017), either using a soft router to mix multiple adapters for greater capacity (Li et al., 2024; Luo et al., 2024; Zhu et al., 2023; Wu et al., 2025b; Luo et al., 2025), or composing adapters to generalize across domains (Huang et al., 2024; Wang et al., 2024; Ma et al., 2024). These methods address a fundamentally different problem from ours: they aim to increase capacity or enable multi-domain transfer, not to prevent optimization interference between heterogeneous capabilities in a single domain setting. Moreover, their *soft mixing* routes each token's gradient to *all* adapters, resulting in an interaction between them. In contrast, DART uses a deterministic hard router so each token updates exactly one adapter, preventing reasoning and tool-use from competing over shared trainable parameters.

3 Preliminaries

This section presents the Agentic Reinforcement Learning and describes low-rank adaptation.

3.1 Agentic Reinforcement Learning (ARL)

An LLM agent $\pi_\theta(c_t \mid c_{<t})$ generates a trajectory τ under query q, interleaving reasoning and tool-use tokens.

$$\tau = (c_1, \ldots, c_t, \ldots, c_T), \tag{1}$$

To distinguish the roles of tokens within a trajectory, we define a *role-based* router function $\ell : \{1, \ldots, T\} \to \{r, a\}$. Here, $\ell(t) = r$ indicates that c_t is a **reasoning token**, while $\ell(t) = a$ indicates a **tool-use token**. Concretely, the router is defined as:

$$\ell(t) = \begin{cases} a, & \text{if } c_{<t} \text{ in tool-call span,} \\ r, & \text{otherwise.} \end{cases} \tag{2}$$

This assignment is fully deterministic and triggered by special tokens (e.g., <search> marks the start of a tool-call span as illustrated in Fig. 8(B)). The agent is optimized to maximize the expected reward $\mathcal{J}(\theta) = \mathbb{E}[R(\tau)]$. We estimate the policy gradient:

$$\nabla_\theta \mathcal{J}(\theta) \approx \mathbb{E}_{\tau \sim \pi_\theta} \left[\mathcal{A}(\tau) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(c_t \mid c_{<t}) \right], \tag{3}$$

where $\mathcal{A}(\tau)$ is the advantage derived by reward $R(\tau)$. In standard ARL, a single set of shared parameters θ is updated using gradients from both reasoning and tool-use tokens, without considering the distinction specified by $\ell(t)$.

3.2 Low-Rank Adaptation (LoRA)

To reduce fine-tuning overhead, Low-Rank Adaptation (LoRA) freezes the pre-trained weights $W \in \mathbb{R}^{d\times h}$ and introduces trainable low-rank decomposition matrices. For a given layer, let $\mathbf{h}_t \in \mathbb{R}^h$ denote the hidden state corresponding to token c_t. The forward pass is modified as:

$$\mathbf{h}'_t = W\mathbf{h}_t + \Delta W\mathbf{h}_t = W\mathbf{h}_t + BA\mathbf{h}_t, \quad (4)$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times h}$ are low-rank matrices with $r \ll \min(d, h)$. Only A and B are trained while W is frozen, and the same ΔW is applied to all tokens in τ.

4 Do Reasoning and Tool-Use Conflict?

In this section, we investigate whether jointly optimizing reasoning and tool-use in ARL leads to interference. We first introduce *Capability Effect Attribution* (CEA), a diagnostic framework that selectively activates each capability via gradient masking and hybrid inference (§ 4.1–4.2). CEA reveals a clear negative interaction between reasoning and tool-use (§ 4.3), which we trace to gradient misalignment between the two token types (§ 4.4).

4.1 Formalizing Capability Effect Attribution

We formalize CEA by attributing a model's expected correctness to individual capability effects and pairwise interaction terms.

Definition 4.1. The capabilities of agent are indicated by three binary indicators: base x_1, tool-use x_2, reasoning x_3. Here $x_i = 1$ indicates the capability exists and $x_i = 0$ otherwise.

Definition 4.2. Pairwise interaction indicators x_{ij} $(i, j \in \{1, 2, 3\})$ satisfy $x_{ij} = 1$ when capabilities i and j are jointly optimized, and 0 otherwise.

Based on the above definition, each model $\mathcal{M}_k$ associates a binary *capability indicator vector*

$$\mathbf{x}_{\mathcal{M}_k} = [x_1, x_2, x_3, x_{12}, x_{13}, x_{23}] \in \{0, 1\}^6. \quad (5)$$

For example, a model jointly trained for tool-use and reasoning has $\mathbf{x}_{\mathcal{M}} = [1, 1, 1, 1, 1, 1]$, indicating that all individual capabilities and their pairwise interactions are active. By corresponding different model with a fixed binary capability vector, we can perform controlled comparisons by comparing different models on the same question.

Proposition 3. *Let $s^q_{\mathcal{M}} \in (0, 1)$ be the expected correctness of model $\mathcal{M}$ on question q. Then $s^q_{\mathcal{M}}$ can be represented by*

$$s^q_{\mathcal{M}} = \sigma(\mathbf{x}^\top_{\mathcal{M}}\boldsymbol{\lambda}^q), \quad \boldsymbol{\lambda}^q = [\lambda^q_1, \lambda^q_2, \lambda^q_3, \lambda^q_{12}, \lambda^q_{13}, \lambda^q_{23}], \quad (6)$$

where $\sigma(\cdot)$ is sigmoid function, and λ^q_i and λ^q_{ij} are the main and interaction effects for question q.

The proposition indicates that the correctness $s^q_{\mathcal{M}}$ can be represented as a function of the composition $\mathbf{x}^\top_{\mathcal{M}}\boldsymbol{\lambda}^q$, so that the interaction coefficient directly serves our diagnostic purpose: $\lambda^q_{ij} > 0$ indicates a synergy between capability x_i and x_j, while $\lambda^q_{ij} < 0$ indicates interference between them. Therefore, diagnosing whether reasoning and tool-use interference reduces to examining the sign of λ^q_{23}.

Unlike predictive regression (e.g., logistic models (Hosmer et al., 2013)) that fits shared coefficients across samples, CEA performs *per-question attribution*: for each q, we solve an independent system to obtain $\boldsymbol{\lambda}^q$. This is feasible because the capability indicators are discrete and finite, so $\mathbf{x}^\top_{\mathcal{M}}\boldsymbol{\lambda}^q$ can exactly represent any probability assignment over these configurations (see Appendix A). Concretely, since $\boldsymbol{\lambda}^q \in \mathbb{R}^6$, it can be uniquely determined from six model variants with linearly independent $\mathbf{x}_{\mathcal{M}}$ and their corresponding $s^q_{\mathcal{M}}$.

Solving $\boldsymbol{\lambda}^q$. As clarified, specifying the interference requires to obtain $\boldsymbol{\lambda}^q$. To do so, applying the logit transform to both sides of Eq. 6 yields an equivalent system:

$$z^q_{\mathcal{M}_k} = \log \frac{s^q_{\mathcal{M}_k}}{1 - s^q_{\mathcal{M}_k}} = \mathbf{x}^\top_{\mathcal{M}_k}\boldsymbol{\lambda}^q. \quad (7)$$

Since $\lambda^q \in \mathbb{R}^6$, stacking the equations from 6 different model variants gives the linear system

$$\mathbf{z}^q = \mathbf{X}\boldsymbol{\lambda}^q, \quad (8)$$

which is enough to specify the value of $\boldsymbol{\lambda}^q$. Here the k-th row of the design matrix $\mathbf{X}$ is an indicator vector of $\mathbf{x}_{\mathcal{M}_k}$ model $\mathcal{M}$, and the k-th component of $s^q_{\mathcal{M}}$ is the corresponding expected correctness.

4.2 Constructing Controlled Model Variants for CEA

As clarified, design matrix $\mathbf{X}$ should be full-rank (Fig. 1B). The key challenge is to construct model

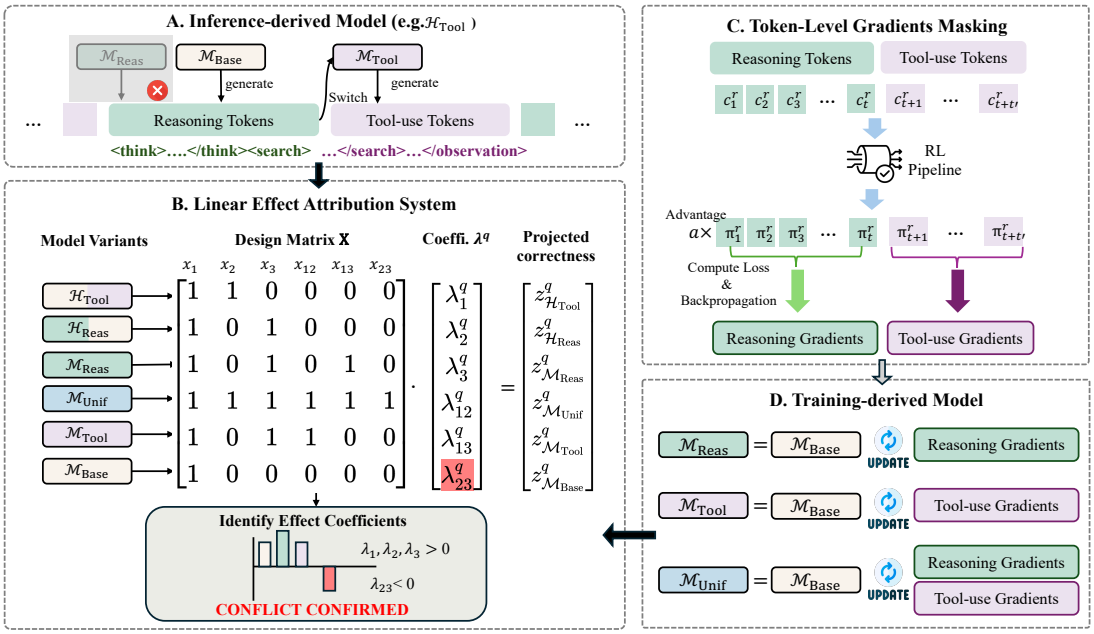


Figure 1: Overview of Capability Effect Attribution (CEA). **(A).Inference-derived Model:** Different token types are routed to separately trained models at inference time, composing capabilities without parameter-level interaction. **(B).Design Matrix and Attribution:** Six model variants populate the design matrix $\mathbf{X}$; solving the system yields per-question coefficients $\boldsymbol{\lambda}^q$, where $\lambda_{23} < 0$ signals interference. **(C).Token-Level Gradient Masking:** Binary masks gate per-token gradient contributions, isolating capability-specific parameter updates during training. **(D).Training-derived Models:** Gradient masking produces specialized variants from a shared backbone, enabling controlled comparisons across capability configurations.

variants that realize different capability configurations $\mathbf{x}_{\mathcal{M}_k}$, while keeping data, architecture, and hyperparameters unchanged. Our insight is that different capabilities are captured by gradients on disjoint token subsets (e.g., reasoning or tool-using) during training. Thus, by **gradient masking**(§ 4.2.1), we produce 4 models with varied indicator vectors from a shared base model. However, training-time control alone cannot yield every configuration we need (4 v.s. 6). To fill this gap, we introduce **hybrid inference**, an inference-time routing scheme that composes separately trained models without parameter-level interaction (§ 4.2.2), which resulting the other 2 indicator vectors we require.

4.2.1 Training-derived Models

As shown in Fig. 1D, we start from a common pretrained model and apply gradient masking to derive three specialized variants.

1.Base Model $\mathcal{M}_{\text{Base}}$. The off-the-shelf pretrained LLM, providing only base capabilities: $\mathbf{x}_{\text{Base}} = [1, 0, 0, 0, 0, 0]$. Starting from this model, we derive three variants via gradient masking. All variants share identical training data, architecture, and hyperparameters.

Using the token type $\ell(t)$ from § 3, we define a binary mask $\mathbf{m} = [m_1, \ldots, m_T]$ that gates per-token gradient contributions in Eq.(3):

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(c_t \mid c_{<t})\, \mathcal{A}(\tau) m_t \right]. \quad (9)$$

where $m_t \in \{0, 1\}$ gates whether token c_t contributes to the update. We partition indices into $\mathcal{T}_{\text{reas}} = \{t \mid \ell(t) = r\}$ and $\mathcal{T}_{\text{tool}} = \{t \mid \ell(t) = a\}$, and define three masking schemes:

2.Reasoning-specialized model $\mathcal{M}_{\text{Reas}}$. It retains only reasoning gradients:

$$m_t^{(\text{Reas})} = \mathbb{I}(t \in \mathcal{T}_{\text{reas}}), \qquad \mathbf{x}_{\text{Reas}} = [1, 0, 1, 0, 1, 0].$$

3.Tool-specialized model $\mathcal{M}_{\text{Tool}}$. It retains only tool-use gradients:

$$m_t^{(\text{Tool})} = \mathbb{I}(t \in \mathcal{T}_{\text{tool}}), \qquad \mathbf{x}_{\text{Tool}} = [1, 1, 0, 1, 0, 0].$$

4.Unified model $\mathcal{M}_{\text{Unified}}$. It uses the standard ARL objective:

$$m_t^{(\text{Uni})} = 1 \text{ for all } t, \qquad \mathbf{x}_{\text{Unified}} = [1, 1, 1, 1, 1, 1].$$

Note that even in single-capability models (e.g., $\mathcal{M}_{\text{Reas}}$), the retained capability's gradients still update the shared base parameters, so base and that capability are jointly optimized, giving $x_{1j} = 1$.

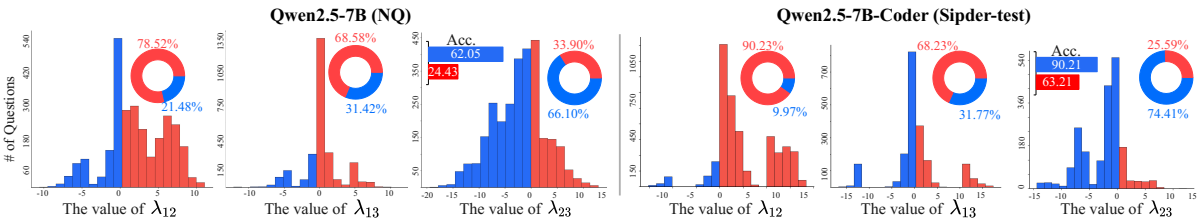


Figure 2: **Interference is specific to the reasoning–tool-use interaction.** Blue/red indicates negative/positive interaction coefficients. λ_{12}^q (base–reasoning) and λ_{13}^q (base–tool) are predominantly positive, showing each capability individually synergizes with the base. In contrast, λ_{23}^q (reasoning–tool) is dominated by negative values, revealing systematic interference under joint optimization. The **Acc.** columns show that interference concentrates on high-accuracy questions solvable by each capability alone, while synergy emerges on questions requiring both capabilities but degraded by their interference. Additional results are in Fig. 7.

4.2.2 Inference-derived Models

To complete the design matrix with 6 linearly independent rows, we add 2 hybrid-inference variants (Fig. 1A) that route different token types to separately trained models, eliminating parameter-level interaction.

5. Tool-hybrid model $\mathcal{H}_{\text{Tool}}$. We use the base model $\mathcal{M}_{\text{Base}}$ generates reasoning tokens; $\mathcal{M}_{\text{Tool}}$ is invoked for tool-action tokens. Because parameters are never jointly optimized, all interaction indicators remain zero: $\mathbf{x}_{\mathcal{H}_{\text{Tool}}} = [1, 1, 0, 0, 0, 0]$.

6. Reasoning-hybrid model $\mathcal{H}_{\text{Reas}}$. Similarly, we use the reasoning specialized model $\mathcal{M}_{\text{Reas}}$ for reasoning tokens, and the base model $\mathcal{M}_{\text{Base}}$ to handle tool-action tokens. No joint optimization occurs, yielding $\mathbf{x}_{\mathcal{H}_{\text{Reas}}} = [1, 0, 1, 0, 0, 0]$.

By doing all of these, we obtain six rows of $\mathbf{X} \in \mathbb{R}^{6\times 6}$, and they linearly independent (Fig. 1B), guaranteeing identifiability of $\boldsymbol{\lambda}^q$.

4.3 Diagnosing Interference via CEA

We instantiate CEA to quantify all three pairwise interaction coefficients. For each question q, we solve for $\boldsymbol{\lambda}^q$ using the design matrix $\mathbf{X}$ induced by the six model variants defined in § 4.2. All training-derived models are trained under identical hyperparameters and to convergence. The correctness s_v^q of question q is estimated via 50 independent answers per model–question pair. Additional implementation details are provided in Appendix D.

Fig. 2 shows the per-question distributions of λ_{12}^q, λ_{13}^q, and λ_{23}^q. As expected, λ_{12}^q (base–reasoning) and λ_{13}^q (base–tool) are predominantly positive: adding either capability alone improves the base model. In contrast, λ_{23}^q (reasoning–tool) is predominantly negative, suggesting that **joint optimization of the two capabilities introduces negative interaction**. The Acc. columns further reveal that interference concentrates on high-accuracy questions solvable by each capability alone, while synergy concentrates on questions that demand both capabilities yet remain low-accuracy due to their interference. However, CEA only reveals *where* and *how much* interference exists, not *why it arises*. To uncover the underlying mechanism, we turn to gradient analysis.

4.4 Explaining Interference via Gradient

A natural candidate is the gradient: if reasoning and tool-use tokens require misaligned parameter updates, joint optimization cannot satisfy both simultaneously. To test this, we measure the angular alignment between gradients from different token types. Specifically, we compute the gradient $\mathbf{g}_\tau^{(b)}$ for token type $b \in \{r, a\}$ in trajectory τ across $N = 16$ rollouts. We calculate the average angle between gradients of different types within the same trajectory $\mathbb{E}_i[\angle(\mathbf{g}_{\tau_i}^{(r)}, \mathbf{g}_{\tau_i}^{(a)})]$. As a baseline, we calculate the average angle between gradients of the same type from different trajectories $\mathbb{E}_{i\neq j}[\angle(\mathbf{g}_{\tau_i}^{(b)}, \mathbf{g}_{\tau_j}^{(b)})]$. Additional implementation details are provided in Appendix E.

As shown in Fig. 3, the angles between same-type gradients are small, while the gradients in different types (reasoning and tool-use) are nearly orthogonal. This orthogonality indicates that *reasoning and tool-use each require a distinct update direction*. Consequently, averaging these gradients forces the update toward a **compromise direction** sub-optimal for both. This finding points to a clear design principle: an effective solution must route reasoning and tool-use gradients into separate parameter subspaces.

5 Disentangled Action-Reasoning Tuning

Our finding (§ 4) shows that reasoning and tool-use negatively interact (Fig. 2) because their gradients

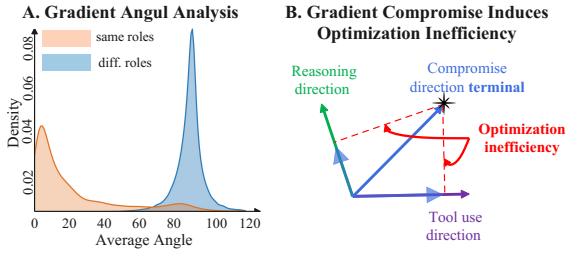


Figure 3: **Gradient misalignment leads to optimization inefficiency. (A).** Gradient angle distributions on NQ under Qwen2.5-3B, where same-capability gradients are aligned, while reasoning and tool-use gradients are nearly orthogonal. **(B).** Averaged orthogonal gradients yield a compromise update direction, leading to optimization inefficiency.

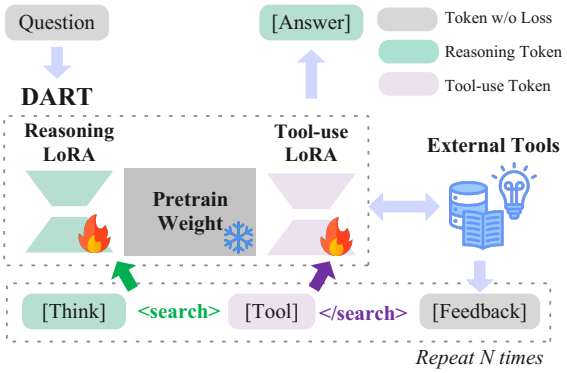


Figure 4: **Illustration of DART.** A frozen backbone augmented with two disjoint LoRA adapters for reasoning and tool-use, both attached to all linear layers, where a token-level router directs gradients into separate parameter subspaces to avoid interference.

misalignment in shared parameter space (Fig. 3). This points to a clear principle: the two capabilities should update **separate parameter subspaces**. A naive approach, training two independent models as a 2-Agent system, achieves this isolation but introduces additional system complexity (see Appendix F).

To avoid this overhead while still validating our finding, we design Disentangled Action-Reasoning Tuning (DART), a **simple yet effective** framework that achieves gradient isolation within a single model. Concretely, DART keeps the pretrained backbone weights W frozen and attaches two disjoint LoRA adapters: $\theta^r = \{B_r, A_r\}$ for reasoning and $\theta^a = \{B_a, A_a\}$ for tool-use. Freezing W is necessary because if W is also trained, both adapters' gradients would flow into the same backbone, bringing back the interference we aim to eliminate.

With this architecture, at each decoding step t, the model activates an adapter $u_t \in \{r, a\}$ determined by the token router $\ell(t)$ defined in Eq.(2). As illustrated in Fig. 4, token roles are assigned by special sentinel tokens (e.g., <search> triggers the tool-use LoRA). To ensure robust routing, we extend the vocabulary so that each sentinel token is a single atomic entry, allowing the model to predict the routing signal as an indivisible unit rather than assembling it character by character. Keeping the router rule-based removes additional training signals from the router itself, so that any improvement observed in subsequent experiments can be attributed purely to gradient disentanglement.

The forward pass for the hidden state $\mathbf{h}_t$ is then computed as:

$$\mathbf{h}'_t = W\mathbf{h}_t + B_{u_t}A_{u_t}\mathbf{h}_t. \tag{10}$$

As each token activates only the adapter associated with its capability type, the two parameter sets θ^r and θ^a are updated independently.

In § 4, we identified the root cause of the negative λ^q_{23}: reasoning and tool-use gradients compete for the same shared parameters. DART directly addresses this cause: since $\theta^r \cap \theta^a = \emptyset$ and W is frozen, no trainable parameter receives gradients from both token types. As we show in § 6, even this minimal design already brings clear gains, confirming that gradient misaligned between reasoning and tool-use is a non-negligible limitation in ARL.

Remark 5.1. One might worry that freezing W limits DART's representational capacity relative to full-parameter ARL. However, recent studies show that RL-based tuning primarily updates a sparse subset of parameters (Mukherjee et al., 2025); moreover, the outcome-based reward used in ARL carries at most one bit of information per rollout, making the effective per-step learning signal extremely sparse. This further reduces the parameter capacity needed for each update. Indeed, LoRA adapters have been shown to match full-parameter RL tuning under this regime (Schulman and Lab, 2025).

6 Experiments

To verify whether the gradient misalignment between reasoning and tool-use is a non-negligible bottleneck in ARL, we evaluate our DART on seven retrieval-augmented QA benchmarks and six NL2SQL benchmarks. Following Search-R1 (Jin et al., 2025) and SkyRL-SQL (Liu et al., 2025), all methods share the same backbone, data, RL algorithm, and hyperparameters; only gradient disentanglement differs (details in Appendix B). We

organize experiments around three questions: **Q1**: Does resolving the identified interference by our DART yields consistent improvements across tasks and scales? (§ 6.1) **Q2**: Does interference harm both reasoning and tool-use separately, and can it be resolved by disentangling gradient as our DART did? (§ 6.2) **Q3**: Do the gains come from gradient disentanglement or simply from increased parameter capacity? (§ 6.3)

Datasets. The benchmarks span two cases. **General QA**: NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022) for factual single-step QA. **Multi-Hop QA**: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), are multi-document reasoning. We train on merged NQ and HotpotQA splits, evaluate on all seven benchmarks, and report Exact Match (EM) (Yu et al., 2024). NL2SQL datasets are in Appendix B.4.

6.1 Main Results

We first address **Q1** by comparing DART against joint-optimization and multi-LoRA baselines.

Baselines. (1) Joint-optimization ARL: Search-R1-GRPO (Jin et al., 2025), the standard agentic RL baseline where reasoning and tool-use share all trainable parameters. **(2) Soft-routing multi-LoRA**: MixLoRA (Li et al., 2024), which uses multiple LoRA experts with learned routing, partially mixing gradients across capabilities. **(3) Other baselines**: Direct Inference, CoT (Wei et al., 2022), Rejection Sampling (Ahn et al., 2024), IR-CoT (Trivedi et al., 2023), RAG (Lewis et al., 2020), SFT (Chung et al., 2024), and R1 variants (Guo et al., 2025).

As shown in Tab. 1, disentangling gradients by our DART yields consistent gains across all benchmarks. **(1)** DART surpasses all joint-optimization baselines, confirming that *the interference identified in § 4 is indeed harmful and can be effectively mitigated*. **(2)** MixLoRA's soft routing consistently underperforms DART, indicating that *partial gradient disentanglement is insufficient*; full isolation via hard token-level routing is necessary. The same trends hold on the 7B backbone, Llama3.1-8B, and NL2SQL task (Appendix. C).

6.2 Mechanism Analysis

We next explore **Q2** by two experiments: **(1).** isolating each capability to verify that disentanglement improves both reasoning and tool-use separately; **(2).** testing whether inference-time model composition can replace training-time disentanglement.

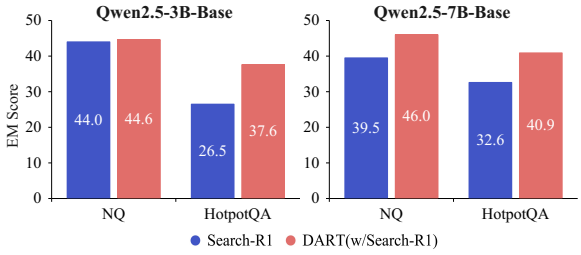


Figure 5: **Reasoning under Fixed Retrieval.** DART achieves higher EM than Search-R1 on NQ and HotpotQA when both use identical retrieval contexts, demonstrating improved reasoning capability independent of retrieval quality.

1. Interference Harms Each Capability Individually. To determine whether joint optimization degrades reasoning, we feed both DART and Search-R1 with **identical retrieval contexts** (collected from Search-R1's rollouts), so that any performance differences are attributed to reasoning ability. As in Fig. 5, DART consistently achieves higher EM than Search-R1 given the same retrieved evidence, confirming that joint optimization impairs reasoning learning and our DART mitigates this impairment. We further compare retrieval accuracy between DART and Search-R1 with fixed reasoning traces, confirming that joint optimization also impairs tool-use learning (Appendix H).

2. Training-Time Isolation vs. Inference-Time Composition. We test whether the damage from interference can be recovered at inference time by combining separately trained specialized models. Concretely, we extract each adapter from DART individually: $\text{DART}_{\text{Reas}}$: reasoning adapter only, $\text{DART}_{\text{Tool}}$: tool-use adapter only, which actives the adapter only if reasoning or tool-using respectively. Then we compare them against the hybrid schemes ($\mathcal{H}_{\text{Reas}}$, $\mathcal{H}_{\text{Tool}}$) from § 4.2 (Fig. 1A), which compose separately trained specialized models at inference time. As shown in Tab. 2, each DART adapter substantially outperforms its hybrid counterpart, confirming that joint training degrades each capability and that inference-time composition *cannot* recover this degradation.

6.3 Ablation Study

Finally, we address **Q3** with two ablations: one compares DART with a matched-rank single LoRA and a 2-Agent upper bound to rule out capacity effects; the other sweeps LoRA rank to verify insensitivity to parameter budget.

Table 1: General and Multi-Hop QA results for **Qwen2.5-3b-Base/Instruct**. Best results are in bold. ◊ denotes results from (Jin et al., 2025); † and ⋆ denote in- and out-domain datasets.

Methods	General QA			Gen-Avg	Multi-Hop QA				MH-Avg	Avg
	NQ†	TriviaQA⋆	PopQA⋆		HotpotQA†	2Wiki⋆	Musique⋆	Bamboogle⋆		
Direct Inference◊	0.106	0.288	0.108	0.167	0.149	0.244	0.020	0.024	0.109	0.134
CoT◊	0.023	0.032	0.005	0.020	0.021	0.021	0.002	0.000	0.011	0.015
IRCoT◊	0.111	0.312	0.200	0.208	0.164	0.171	0.067	0.240	0.161	0.181
RAG◊	0.348	0.544	0.387	0.426	0.255	0.226	0.047	0.080	0.152	0.270
SFT◊	0.249	0.292	0.104	0.215	0.186	0.248	0.044	0.112	0.147	0.176
R1-base◊	0.226	0.455	0.173	0.285	0.201	0.268	0.055	0.224	0.187	0.229
R1-instruct◊	0.210	0.449	0.171	0.277	0.208	0.275	0.060	0.192	0.184	0.224
Rejection Sampling◊	0.294	0.488	0.332	0.371	0.240	0.233	0.059	0.210	0.186	0.265
Qwen2.5-3B-Instruct										
Search-R1	0.397	0.565	0.391	0.451	0.331	0.310	0.124	0.232	0.249	0.336
MixLoRA	0.431	0.578	0.419	0.476	0.346	0.348	0.125	0.288	0.277	0.362
DART	**0.451**	**0.602**	**0.476**	**0.510**	**0.392**	**0.376**	**0.143**	**0.352**	**0.316**	**0.399**
Qwen2.5-3B-Base										
Search-R1	0.440	0.582	0.413	0.478	0.265	0.244	0.061	0.113	0.171	0.303
MixLoRA	0.427	0.595	0.443	0.488	0.292	0.282	0.063	0.176	0.203	0.325
DART	**0.457**	**0.605**	**0.478**	**0.513**	**0.399**	**0.389**	**0.155**	**0.352**	**0.324**	**0.405**

Table 2: DART adapters vs. hybrids.

Methods	Qwen2.5-3B		Qwen2.5-7B	
	NQ	HotpotQA	NQ	HotpotQA
$\mathcal{H}_{\text{Reas}}$	0.435	0.324	0.438	0.327
$\text{DART}_{\text{Reas}}$	0.448	0.359	0.449	0.412
$\mathcal{H}_{\text{Tool}}$	0.248	0.212	0.305	0.255
$\text{DART}_{\text{Tool}}$	0.372	0.283	0.378	0.332

Ablation 1: Disentanglement vs. More Parameters. We verify that the gains stem from gradient isolation, not from additional parameters.

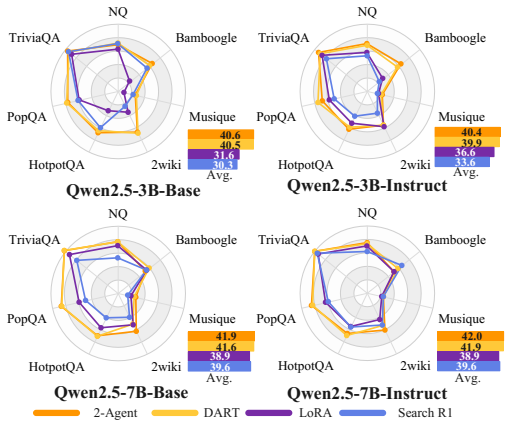


Figure 6: **Ablation on parameter capacity.** A matched-rank single LoRA performs similarly to Search-R1, while DART approaches the 2-Agent upper bound (two models each dedicated to one capability) across scales and benchmarks.

We compare three baselines: **Search-R1** (shared parameters, full fine-tuning), **LoRA** (single adapter r=16, same total rank as DART's r=8×2), and **2-Agent** (two fully independent models for tool-use and reasoning, Fig. 9), which serves as the parameter-disentangled upper bound.

As shown in Fig. 6, a single LoRA with matched total rank performs nearly identically to Search-R1, indicating that extra parameters alone provide limited benefit. In contrast, DART closely matches the 2-Agent upper bound, confirming that the gains come from isolating the two capabilities' gradients rather than from increased capacity.

Ablation 2: Effect of LoRA Rank. We find that DART's performance is largely insensitive to the LoRA rank; detailed results are in Appendix G.

7 Conclusion

This work identifies a fundamental yet previously overlooked problem in ARL: reasoning and tool-use negatively interact under joint optimization. CEA first exposes this phenomenon empirically, and gradient analysis traces it to misaligned parameter updates that push the two capabilities toward compromise directions. To validate this finding, we propose DART, which simply freezes the backbone and routes reasoning and tool-use tokens to disjoint LoRA adapters, ensuring that the two capabilities should update in separate parameter subspaces. This minimal change alone consistently surpasses all joint-optimization baselines and nearly recovers the 2-Agent upper bound across thirteen benchmarks, providing strong evidence that capability interference is a non-negligible bottleneck in current ARL systems. As agents grow more capable, one-size-fits-all optimization may become increasingly insufficient, and capability-aware training offers a promising alternative.

Limitations

Although resolving gradient misalignment yields consistent gains, it is not a free lunch. Because DART only replaces the single shared LoRA with two disjoint adapters, it is fully compatible with existing RL training pipelines. However, its token-level adapter switching introduces extra scheduling complexity at high-concurrency serving. Mitigating this overhead through batched adapter dispatch or fused multi-LoRA kernels is an engineering optimization orthogonal to ours contribution, which we leave to future work.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *European Chapter of the Association for Computational Linguistics*.

Leona S Aiken. 1991. *Multiple regression: Testing and interpreting interactions*. sage Newbury Park, CA.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. Preprint arXiv:2212.08073.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025. Toolstar: Empowering llm-brained multi-tool reasoner via reinforcement learning. Preprint arXiv:2505.16410.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. Preprint arXiv:2504.11536.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 2505–2515.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Preprint arXiv:2501.12948.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*.

David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied Logistic Regression*, 3rd edition. John Wiley & Sons.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *First Conference on Language Modeling*.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, and 1 others. 2025. Verltool: Towards holistic agentic reinforcement learning with tool use. Preprint arXiv:2509.01055.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Empirical Methods in Natural Language Processing*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Conference on Neural Information Processing Systems*.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and 1 others. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. Preprint arXiv:2404.15159.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Dingchu Zhang, Xixi Wu, Jialong Wu, and 1 others. 2025. Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. Preprint arXiv:2509.13305.

Xiuyu Li, Jinkai Zhang, Mingyang Yi, Yu Li, Longqiang Wang, Yue Wang, and Ju Fan. 2026a. Ets: Energy-guided test-time scaling for training-free rl alignment. In *International Conference of Machine Learning*.

Zhaochun Li, Mingyang Yi, Yue Wang, Shisheng Cui, and Yong Liu. 2026b. Towards a theoretical understanding to the generalization of rlhf. Preprint arXiv:2601.16403.

Shu Liu, Alan Zhu, Sumanth Hegde, Shiyi Cao, Shuo Yuan, Samion Suwito, Tyler Griggs, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. SkyRL-SQL: Multi-turn SQL data agents via RL. In *First Workshop on Multi-Turn Interactions in Large Language Models*.

Siqi Luo, Haoran Yang, Yi Xin, Mingyang Yi, Guangyang Wu, Guangtao Zhai, and Xiaohong Liu. 2025. Tr-pts: Task-relevant parameter and token selection for efficient tuning. In *International Conference on Computer Vision*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. Preprint arXiv:2402.12851.

Yufei Ma, Zihan Liang, Huangyu Dai, Ben Chen, Dehong Gao, Zhuoran Ran, Wang Zihan, Linbo Jin, Wen Jiang, Guannan Zhang, and 1 others. 2024. Modula: Mixture of domain-specific and universal lora for multi-task learning. In *Empirical Methods in Natural Language Processing*.

Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. 2025. Agentic RL scaling law: Spontaneous code execution for mathematical problem solving. In *Conference on Neural Information Processing Systems*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. Preprint arXiv:2212.10511.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tür, and Hao Peng. 2025. Reinforcement learning finetunes small subnetworks in large language models. In *Conference on Neural Information Processing Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Conference on Neural Information Processing Systems*.

Feng Peiyuan, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. Agile: A novel reinforcement learning framework of llm agents. *Conference on Neural Information Processing Systems*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. Preprint arXiv:2504.13958.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, and 1 others. 2025. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. Preprint arXiv:2509.13309.

Yi Ren and Danica J Sutherland. 2025. Learning dynamics of llm finetuning. In *International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Conference on Neural Information Processing Systems*.

John Schulman and Thinking Machines Lab. 2025. Lora without regret. *Thinking Machines Lab: Connectionism*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. Preprint arXiv:2402.03300.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *European Conference on Computer Systems*.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. Preprint arXiv:2505.01441.

Qwen Team. 2024. Qwen2 technical report. Preprint arXiv:2407.10671.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics*.

Haowen Wang, Tao Sun, Congyun Jin, Yingbo Wang, Yibo Fan, Yunqi Xu, Yuliang Du, and Cong Fan. 2024. Customizable combination of parameter-efficient modules for multi-task learning. In *International Conference on Learning Representations*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. Preprint arXiv:2212.03533.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Conference on Neural Information Processing Systems*.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. 2025. Autotir: Autonomous tools integrated reasoning via reinforcement learning. Preprint arXiv:2507.21836.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. 2024. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Conference on Neural Information Processing Systems*.

Weiqi Wu, Xin Guan, Shen Huang, Yong Jiang, Pengjun Xie, Fei Huang, Jiuxin Cao, Hai Zhao, and Jingren Zhou. 2025a. Masksearch: A universal pre-training framework to enhance agentic search capability. Preprint arXiv:2505.20285.

Xun Wu, Shaohan Huang, and Furu Wei. 2025b. Mixture of lora experts. In *International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Hua Ye, Siyuan Chen, Haoliang Zhang, Weihao Luo, Yanbin Li, and Xuan Zhang. 2026. Synergy over discrepancy: A partition-based approach to multi-domain LLM fine-tuning. In *Conference on Neural Information Processing Systems*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Conference on Neural Information Processing Systems*.

Xiaosong Yuan, Chen Shen, Shaotian Yan, kaiyuan liu, Xiaofeng Zhang, Sinan Fan, Liang Xie, Wenxiao Wang, Renchu Guan, Ying Wang, and Jieping Ye. 2026. Differential fine-tuning large language models towards better diverse reasoning abilities. In *International Conference on Learning Representations*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics*.

Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. 2025a. Deepanalyze: Agentic large language models for autonomous data science. Preprint arXiv:2510.16872.

Yuxin Zhang, Meihao Fan, Ju Fan, Mingyang Yi, Yuyu Luo, Jian Tan, and Guoliang Li. 2025b. Reward-sql: Boosting text-to-sql via stepwise reasoning and process-supervised rewards. Preprint arXiv:2505.04671.

Yun Zhu, Nevan Wichers, Chu-Cheng Lin, Xinyi Wang, Tianlong Chen, Lei Shu, Han Lu, Canoee Liu, Liangchen Luo, Jindong Chen, and 1 others. 2023. Sira: Sparse mixture of low rank adaptation. Preprint arXiv:2311.09179.

A Theoretical Justification for the Capability Formulation

In this section, we demonstrate that the formulation $s^q_{\mathcal{M}} = \sigma(\mathbf{x}^\top_{\mathcal{M}} \boldsymbol{\lambda}^q)$ presented in the main text exactly characterizes the capability distribution without loss of generality.

Suppose the true success rate of question q is governed by an arbitrary function $f(\cdot)$ such that $s^q_{\mathcal{M}} = \sigma(f((x_1, x_2, x_3)))$, the $s^q_{\mathcal{M}}$ can be represented as Sigmoid function is because it is a probability in $[0, 1]$. Besides that, it is natural that the probability is only related to the abilities of base x_1, reasoning x_2, tool-using x_3.

Because the indicator vector (x_1, x_2, x_3) consists of discrete binary variables representing specific model variants, $f(x_1, x_2, x_3)$ only takes a **finite number** of possible values. Consequently, $f(x_1, x_2, x_3)$ can naturally and exactly be parameterized by the inner product $\mathbf{x}^\top_{\mathcal{M}} \boldsymbol{\lambda}^q$, which is a standard result in statistics (Aiken, 1991).

To illustrate this, suppose that $f(0, 0, 0) = 0$ without loss of generality (model without any ability can not give correct answer), then:

$$
\begin{aligned}
& f(x_1, x_2, x_3) \\
& = f(1,0,0)x_1 + f(0,1,0)x_2 + f(0,0,1)x_3 \\
& + (f(1,1,0) - f(1,0,0) - f(0,1,0))x_1x_2 \\
& + (f(1,0,1) - f(1,0,0) - f(0,0,1))x_1x_3 \\
& + (f(0,1,1) - f(0,0,1) - f(0,1,0))x_2x_3 \\
& + (f(1,1,1) - f(1,1,0) - f(1,0,1) - f(0,1,1))x_1x_2x_3.
\end{aligned}
$$

By modeling $x_i x_j$ as x_{ij} and following the observation in (Aiken, 1991) that the effect of $x_1x_2x_3 \approx 0$, we get the result that $s^q_{\mathcal{M}} = \sigma(f(x_1, x_2, x_3)) = \sigma(\mathbf{x}^\top_{\mathcal{M}} \boldsymbol{\lambda}^q)$, by properly taking $\boldsymbol{\lambda}^q$, e.g., $\lambda^q_1 = f(1, 0, 0)$. According to the analysis in above, evaluating the diagnostic coefficients $\boldsymbol{\lambda}^q$ comprehensively captures the underlying interaction dynamics (e.g., synergy or interference) without introducing any restrictive structural assumptions.

B Experimental Settings

This section details the experimental settings used in our RL training, including the training algorithm, rollout configuration, prompt templates, reward formulations, and system-level optimizations. Unless otherwise stated, these settings are shared across all experiments.

B.1 RL Training Setup

For GRPO training, we follow the implementation in Verl (Sheng et al., 2025). The backbone model is Qwen2.5 (Team, 2024) series. For retrieval-augmented QA, we integrate an E5 retriever (Wang et al., 2022) and the 2018 Wikipedia dump (Karpukhin et al., 2020) as the corpus. For Natural language-to-SQL (NL2SQL) task, we use the SkyRL-SQL training set (Liu et al., 2025) with Qwen2.5-7B-Coder as the base model, and the agent invokes a SQLite execution engine as the tool. All experiments are conducted on a cluster of 8× NVIDIA A800 GPUs. Training is conducted for 100 optimization steps with a learning-rate warm-up ratio of 0.1. All GRPO experiments use a fixed configuration with rollout batch size 256, gradient batch size 64, temperature 1.0, top-p 1.0, and learning rate 1×10^{-6}. The KL-divergence coefficient β and clipping ratio ϵ are set to 0.001 and 0.2, respectively.

For all variants involving LoRA adaptation, we scale the learning rate by 10× following prior guidance (Schulman and Lab, 2025). To enable precise token-level routing in DART, we extend the tokenizer vocabulary with a small set of special tokens that explicitly mark reasoning and tool-use segments.

To improve training efficiency, we enable gradient checkpointing, FSDP offloading, and vLLM-based rollouts. Model checkpoints are saved every 20 training steps. If training diverges, we evaluate the most recent stable checkpoint according to the reward curve; otherwise, the final checkpoint is used for evaluation.

For both tasks, all compared methods share exactly the same training data, base model, prompt template, tool interface, and RL hyperparameters; the *only* difference is the degree and manner of capability decoupling. This unified setup ensures that *observed differences are attributable to the parameterization and routing design*, rather than changes in data, tools, or optimization settings.

B.2 Prompt Templates

We adopt task-specific prompt templates that enforce a minimal structural format while avoiding content-specific biases. Importantly, we intentionally restrict the constraints to the high-level structure (reasoning, tool invocation, final output), without enforcing reflective reasoning styles or problem-solving heuristics. This design choice ensures that the model's learning dynamics during RL remain observable and unbiased, allowing behaviors to emerge naturally from optimization rather than prompt engineering.

Retrieval-Augmented QA. Following Search-R1 (Jin et al., 2025), the template structures the model output into three iterative stages: (1) a reasoning phase, (2) a search engine invocation phase, and (3) a final answer. The maximum action budget B is set to 4, and the top 3 retrieved passages are used by default. The full template is shown in Tab. 3.

Multi-Turn SQL Query Generation. Following SkyRL-SQL (Liu et al., 2025), the template structures the model output into iterative stages: (1) a reasoning phase inside `<think>` blocks, (2) a SQL tool invocation phase inside `<sql>` blocks with execution feedback returned in `<observation>` blocks, and (3) a final SQL solution. The maximum action budget B is set to 4. The full template is shown in Tab. 4.

B.3 Reward Functions

The reward function serves as the sole training signal in our RL framework. We adopt rule-based outcome rewards that evaluate the correctness of the model's final output, without incorporating intermediate or format-based rewards.

Retrieval-Augmented QA. For factual reasoning tasks, the reward is computed using exact match (EM):

$$r_{\text{qa}}(x, y) = \text{EM}(a_{\text{pred}}, a_{\text{gold}}), \tag{11}$$

where a_{pred} is the extracted final answer from the model response y, and a_{gold} denotes the ground-truth answer.

Multi-Turn SQL Query Generation. For NL2SQL, the reward focuses solely on execution accuracy:

$$R_{\text{sql}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if match}(\mathbf{y}, \mathbf{y}_g) \\ -1 & \text{otherwise} \end{cases} \tag{12}$$

where $\mathbf{y}$ denotes the execution result of the predicted SQL query and $\mathbf{y}_g$ denotes the ground-truth execution result. A match is determined by comparing the execution outputs (i.e., result sets) rather than the SQL strings themselves.

B.4 Evaluation Benchmarks for NL2SQL

Our training set is identical to that of SkyRL-SQL (Liu et al., 2025). Following standard conventions, we evaluate execution accuracy (EX) on BIRD-Dev (Li et al., 2023), SPIDER-1.0 (Yu et al., 2018), SPIDER-DK (Gan et al., 2021b), SPIDER-Realistic (Deng et al., 2021), and SPIDER-Syn (Gan et al., 2021a).

C Additional Main Results

Retrieval-Augmented QA with Qwen2.5-7B and Llama3.1-8B. Tab. 5 reports the full results on the Qwen2.5-7B and Llama3.1-8B backbone. The trends are consistent with those observed for the 3B model in Tab. 1: DART outperforms Search-R1 and MixLoRA on nearly all benchmarks and all aggregate metrics. These results confirm that reasoning–tool-use interference is a general phenomenon independent of model scale and architecture, and that DART's disentanglement remains effective across different model families.

Multi-Turn SQL Query Generation. We further evaluate DART on the NL2SQL task to test whether its disentanglement benefit transfers to a different tool-use scenario. The experimental setup and evaluation protocol are detailed in Appendix B. As shown in Tab. 6, DART achieves the best single-model performance, surpassing the Sky-SQL baseline across all six benchmarks while closely matching the more resource-intensive 2-Agent system. MixLoRA's soft routing again underperforms, reinforcing the finding that partial gradient disentanglement is insufficient for effective capability isolation. We further note that MixLoRA's degradation is more severe on NL2SQL than on QA. We attribute this to the larger modality gap between reasoning tokens (natural language) and tool-use tokens (formal SQL): soft routing forces every adapter to absorb gradients from both modalities, amplifying interference when the two token distributions diverge significantly. These results confirm that DART's disentanglement benefit generalizes from retrieval-augmented QA to the NL2SQL setting, where the tool interface, action space, and reward signal are fundamentally different.

D Experimental Details and Extended Results for CEA

This section provides additional experimental details for the CEA presented in § 4.3 and extends the analysis to both tasks to verify the generality of our findings

D.1 Shared Experimental Protocol

For each question q, we consider the six models defined in § 4.1, which correspond to different combi-

Table 3: Prompt template for retrieval-augmented QA.

Retrieval-Augmented QA Prompt Template
Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search>, and it will return the top searched results between <information> and </information>. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer> xxx </answer>. Question: question.

Table 4: Prompt template for multi-turn SQL query generation.

NL2SQL Prompt Template
Task Overview: You are a data science expert. Below, you are provided with a database schema and a natural language question. Your task is to understand the schema and generate a valid SQL query to answer the question within limited turns. You should breakdown the problem, draft your reasoning process, and generate the solution. **Database Engine:** SQLite **Database Schema:** {db_details} This schema describes the database's structure, including tables, columns, primary keys, foreign keys, and any relevant relationships or constraints. **External Knowledge:** {external_knowledge} **Question:** {question} **Instructions:** - Make sure you only output the information that is asked in the question. - The generated query should return all of the information asked in the question without any missing or extra information. - Before generating the final SQL query, please think through the steps of how to write the query. **Format:** - Conduct thinking inside <think> ... </think> blocks every time you get new observation or information. - You can use SQL tool written within a single <sql> ... </sql> block to explore or verify. SQL tool output will be shown as dataframe inside <information> ... </information>. Based on this observation, you can think again and refine. - If you find no further exploration is needed or reaches max turns, you MUST directly provide the final SQL query solution inside <solution> ... </solution>.

nations of base, tool-use, and reasoning capabilities and induce a fixed design matrix $\mathbf{X}$. All models are trained under identical hyper-parameters and to convergence, *differing only in capability activation*, which ensures controlled and fair comparisons across models. Given the six empirical correctness estimates $\{\hat{s}^q_{\mathcal{M}}\}$, we solve a linear system in logit space to obtain the question-level effect vector $\boldsymbol{\lambda}^q$. The interaction coefficient λ^q_{23} captures the deviation of the jointly optimized reasoning–tool-use configuration from the linear additive expectation of the two individual capabilities, where negative values indicate interference and positive values indicate synergy. Similarly, λ^q_{12} and λ^q_{13} capture the base–reasoning and base–tool interactions, respectively.

For numerical stability, we discard any question q for which all six model variants yield $\hat{s}^q_{\mathcal{M}_k} = 0$, i.e., no model ever produces a correct answer across all N samples. Including such questions would cause the smoothed logit values to cluster near $-\infty$ for every variant, making the resulting $\boldsymbol{\lambda}^q$ dominated by boundary artifacts rather than genuine capability differences. After filtering, the additive smoothing defined in the main text ensures that all remaining $\bar{s}^q_{\mathcal{M}_k}$ lie strictly in $(0, 1)$ and the logit transform is well-behaved. We then aggregate all three pairwise coefficients across retained questions and report the proportion of negative and positive values. In all experiments, we set $N = 50$ and adopt the same stochastic decoding strategy as Appendix B, with fixed temperature and top-p sampling. Averaging over multiple samples reduces decoding noise and yields a more stable estimate of model correctness. Except for the task-specific details described below, all inference and evaluation hyper-parameters follow the settings of the main experiments.

D.2 Retrieval-Augmented QA

Token Role Definition. In the retrieval-augmented QA trajectory, we define two token roles: (1) *reasoning tokens* correspond to content within <think> blocks, where the model analyzes

Table 5: General and Multi-Hop QA results for **Qwen2.5-7b-Base/Instruct** and **Llama3.1-8B**. Best results are in bold. ◊ denotes results from (Jin et al., 2025); † and ⋆ denote in- and out-domain datasets.

Methods	General QA			Gen-Avg	Multi-Hop QA				MH-Avg	Avg
	NQ†	TriviaQA⋆	PopQA⋆		HotpotQA†	2Wiki⋆	Musique⋆	Bamboogle⋆		
Direct Inference◊	0.134	0.408	0.140	0.227	0.183	0.250	0.031	0.120	0.146	0.181
CoT◊	0.048	0.185	0.054	0.096	0.092	0.111	0.022	0.232	0.114	0.106
IRCoT◊	0.224	0.478	0.301	0.334	0.133	0.149	0.072	0.224	0.145	0.239
RAG◊	0.349	0.585	0.392	0.442	0.299	0.235	0.058	0.208	0.200	0.304
SFT◊	0.318	0.354	0.121	0.264	0.217	0.259	0.066	0.112	0.164	0.207
R1-base◊	0.297	0.539	0.199	0.345	0.242	0.273	0.083	0.203	0.200	0.262
R1-instruct◊	0.270	0.537	0.199	0.335	0.237	0.292	0.072	0.293	0.224	0.271
Rejection Sampling◊	0.360	0.592	0.380	0.444	0.331	0.296	0.123	0.355	0.276	0.348
Qwen2.5-7B-Instruct										
Search-R1	0.429	0.623	0.427	0.493	0.386	0.346	0.162	**0.400**	0.324	0.396
MixLoRA	0.446	0.625	0.432	0.501	0.398	0.342	0.152	0.368	0.315	0.395
DART	**0.467**	**0.642**	**0.505**	**0.538**	**0.431**	**0.349**	**0.163**	0.397	**0.330**	**0.420**
Qwen2.5-7B-Base										
Search-R1	0.395	0.560	0.388	0.448	0.326	0.297	0.125	0.360	0.277	0.350
MixLoRA	0.458	0.626	0.443	0.509	0.408	0.318	0.156	0.368	0.323	0.402
DART	**0.472**	**0.639**	**0.507**	**0.539**	**0.425**	**0.338**	**0.155**	**0.376**	**0.323**	**0.416**
Llama3.1-8B-Instruct										
Search-R1	0.481	0.659	0.489	0.543	0.438	0.387	0.201	0.448	0.368	0.443
MixLoRA	0.477	0.659	0.475	0.537	0.424	0.411	0.192	0.472	0.375	0.444
DART	**0.501**	**0.665**	**0.516**	**0.561**	**0.464**	**0.411**	**0.217**	**0.476**	**0.392**	**0.464**

Table 6: **Execution accuracy (EX) on NL2SQL benchmarks.** All methods use the same SkyRL-SQL training data and Qwen2.5-7B-Instruct backbone; only the decoupling strategy differs.

Method	BIRD-Dev	Spider-Dev	Spider-DK	Spider-Realistic	Spider-Syn	Spider-Test	Avg.
Sky-SQL	0.4912	0.8085	0.7121	0.7611	0.6925	0.8281	0.7156
LoRA	0.4844	0.8075	0.7159	0.7598	0.7021	0.8277	0.7162
MixLoRA	0.4735	0.7821	0.6820	0.7521	0.6822	0.7823	0.6924
2-Agent	**0.5271**	0.8162	0.7084	**0.7894**	0.7108	**0.8314**	0.7306
DART	0.5215	**0.8251**	**0.7144**	0.7815	**0.7137**	0.8291	**0.7309**

retrieved information and performs multi-step reasoning; (2) *tool-use tokens* correspond to content within <search> blocks, where the model formulates search queries to invoke the retrieval engine. These two roles are mutually exclusive and jointly cover all non-prompt tokens in each trajectory, enabling the six-variant design matrix $\mathbf{X}$ and gradient masking procedure described in § 4.2.

Correctness Metric. We adopt exact match (EM) as the correctness measure:

$$\hat{s}^q_{\mathcal{M}} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{EM}\big(a_q^{(n)}, a_{\text{gold}}\big),$$

where $a_q^{(n)}$ is the extracted final answer from the n-th rollout, a_{gold} is the ground-truth answer, and $\mathrm{EM}(\cdot) \in \{0, 1\}$.

D.3 Multi-Turn SQL Query Generation (NL2SQL)

In this setting, the tool is a SQLite execution engine rather than a search engine, and the action space consists of SQL queries rather than search queries. Following the training setup in Appendix B, we use the SkyRL-SQL training set (Liu et al., 2025) with Qwen2.5-7B-Coder as the base model.

Token Role Definition. In the NL2SQL trajectory, we define two token roles analogous to the QA setting: (1) *reasoning tokens* correspond to content within <think> blocks, where the model plans query strategies, interprets execution feedback, and reasons about schema relationships; (2) *tool-use tokens* correspond to content within <sql> blocks, where the model generates SQL queries for execution. These two roles are mutually exclusive and jointly cover all non-prompt tokens in each trajectory, enabling the same six-variant design matrix

$\mathbf{X}$ and gradient masking procedure.

Correctness Metric. Instead of exact match, we adopt execution accuracy (EX) as the correctness measure:

$$\hat{s}^q_{\mathcal{M}} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{match}\Big(\mathrm{exec}(y_q^{(n)}),\ \mathrm{exec}(y_{\text{gold}})\Big)\,,$$

where $\mathrm{exec}(\cdot)$ denotes the result set returned by executing the SQL query, and $\mathrm{match}(\cdot) \in \{0, 1\}$. A prediction is correct if its execution output matches the ground-truth execution output, regardless of syntactic SQL differences.

D.4 Extended Results

We extend the CEA analysis of Fig. 2 to additional datasets (PopQA, TriviaQA) and model architectures (Qwen2.5-3B, Qwen2.5-7B, Llama3.1-8B). For each dataset, we evaluate on the first 1,000 samples from the test set. As shown in Fig. 7, λ^q_{23} is consistently dominated by negative values across all dataset–model combinations. The Acc. columns confirm that interference concentrates on high-accuracy questions where both capabilities are well-learned. Additionally, the synergy group (positive λ^q_{23}) in 7B models achieves higher accuracy than that in 3B models, reflecting the stronger base capacity. This cross-task and cross-architecture consistency provides evidence that the interference identified by CEA reflects a general property of ARL, rather than an artifact of a specific model or dataset.

E Implementation Details of Gradient Misalignment

This section provides implementation details for the gradient angle analysis described in the main text. Following the training and sampling protocol of (Jin et al., 2025), for each input query we sample $N = 16$ rollouts $\{\tau_i\}_{i=1}^{N}$ from the current policy. All analyses are conducted with fixed base model parameters: we perform forward and backward passes solely to extract gradients and do not update the model.

Based on the token-level masked update in Eq. 9 and the hyperparameter settings described in Appendix B, we compute policy gradients for different token roles within each trajectory. Specifically, for each rollout τ_i, we compute gradients $\mathbf{g}^{(b)}_{\tau_i}$ for token role $b \in \{r, a\}$, where r denotes reasoning tokens and a denotes tool-use tokens. Gradients for different roles are obtained via separate backward passes, with gradients explicitly zeroed between passes to avoid accumulation effects.

Gradient angles are computed from the cosine similarity between two gradient vectors. Given two gradients $\mathbf{g}_1$ and $\mathbf{g}_2$, we first compute their cosine similarity as

$$\cos(\mathbf{g}_1, \mathbf{g}_2) = \frac{\mathbf{g}_1^\top \mathbf{g}_2}{\|\mathbf{g}_1\|_2\,\|\mathbf{g}_2\|_2},$$

where gradients are flattened over all model parameters. The corresponding angle is then obtained by

$$\angle(\mathbf{g}_1, \mathbf{g}_2) = \arccos(\cos(\mathbf{g}_1, \mathbf{g}_2))\,,$$

which yields values in $[0, \pi]$. This conversion allows us to interpret gradient alignment geometrically, with smaller angles indicating stronger alignment and angles approaching $\pi/2$ or larger indicating increasing degrees of misalignment.

All experiments use the same numerical and system settings as training. We enable FlashAttention-2 and gradient checkpointing to support long-sequence computation, and perform all forward and backward passes in bfloat16 precision. In memory-constrained environments, parameters are managed with CPU offloading. The maximum lengths of both prompts and responses are set to 4096 tokens.

Notably, all gradients are computed over the full sequence, but only tokens selected by the corresponding role mask contribute to the policy loss and backpropagation. Gradient clipping is disabled by default to avoid altering the geometry of gradients. We additionally observe qualitatively similar gradient angle patterns when repeating the analysis at other training steps, suggesting that the observed gradient interference is not specific to a single checkpoint.

Fig. 8(A) shows that across all settings, reasoning–tool gradients are close to orthogonal, while same-capability gradients exhibit stronger alignment, indicating clear directional separation. Compared to the 3B model, the 7B model shows a more dispersed distribution of same-role gradients, which we attribute to its larger capacity: with more parameters, the model admits a wider range of gradient directions for the same capability across different samples.

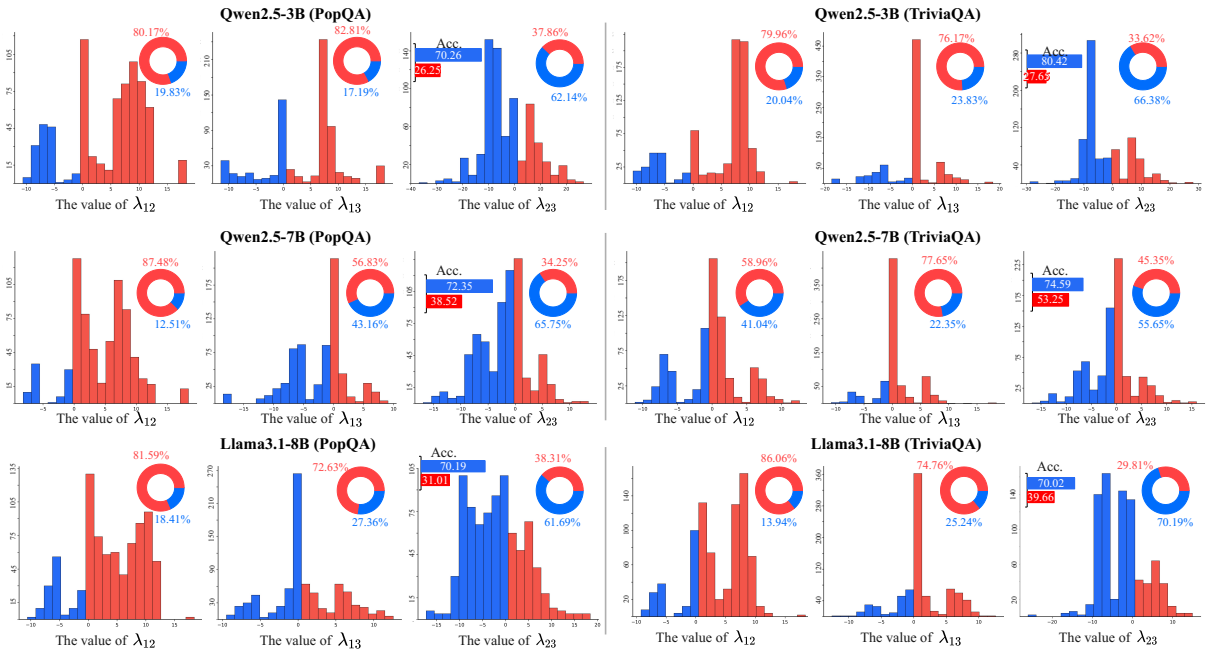


Figure 7: **The reasoning–tool-use interference generalizes across datasets, model scales, and architectures.** We extend the CEA analysis of Fig. 2 to PopQA and TriviaQA with Qwen2.5-3B, Qwen2.5-7B, and Llama3.1-8B, evaluating on the first 1,000 test samples per dataset. Blue/red indicates negative/positive interaction coefficients. Across all combinations, λ^q_{23} is consistently dominated by negative values. The **Acc.** columns confirm that interference concentrates on high-accuracy questions. The synergy group in 7B models shows higher accuracy than in 3B, reflecting the stronger base capacity.

F Theoretical Efficiency: DART vs. 2-Agent System

A common alternative to a unified model is a disentangled 2-agent system, where a specialized reasoning model $\mathcal{M}_{\text{Reas}}$ and a tool-use model $\mathcal{M}_{\text{Tool}}$ collaborate. While this modularity seems intuitive, it introduces significant overhead in resource consumption and latency. Below, we provide a theoretical analysis of why the DART framework is more efficient.

Training Memory: The Shared-Backbone Advantage We analyze the training-time GPU memory complexity of DART in comparison with 2-agent system. Let P denote the number of parameters in the backbone model, and let p denote the number of parameters introduced by a LoRA adapter, where $p \ll P$ (typically below 0.5% of P). Model parameters and gradients are stored in BF16 precision, while optimizer states are stored in FP32 precision.

Under a disentangled multi-agent GRPO setup, two trainable policy backbone models must be resident on GPU. For each model, training stores parameters, gradients, and Adam-style optimizer states, contributing approximately parameters. As a result, the dominant static memory cost scales as $\mathcal{O}(P_{\text{2-agent}} \approx 2 \times 4P = 8P$. In contrast, DART trains both capabilities within a single shared backbone and confines all trainable parameters to lightweight LoRA adapters. The backbone is frozen, and gradients as well as optimizer states are stored only for the adapter parameters. As a result, the dominant static memory cost scales as $\mathcal{O}(P_{\text{DART}}) \approx P + \mathcal{O}(p)$, where the contribution of p is negligible.

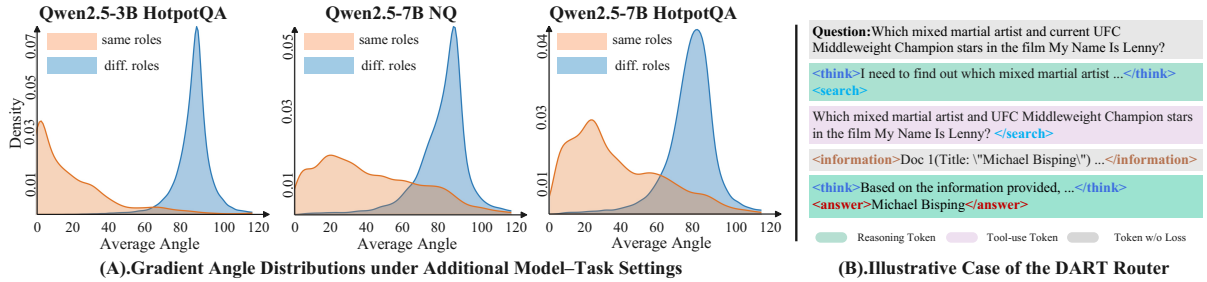
According to our empirical observation, the resulting memory ratio can be approximated as

$$\frac{\mathcal{O}(P_{\text{2-agent}})}{\mathcal{O}(P_{\text{DART}})} \approx \mathcal{O}(8).$$

DART reduces the training-time static memory footprint by roughly $8\times$ while maintaining performance comparable to 2-agent.

Inference Latency: The KV-Cache Advantage. The most critical bottleneck in multi-turn interactions is computing the prefill during context switching.

- **2-Agent Latency:** When $\mathcal{M}_{\text{Reas}}$ generates a thought and hands it to $\mathcal{M}_{\text{Tool}}$, the latter must re-encode the entire conversation history H



(A).Gradient Angle Distributions under Additional Model–Task Settings (B).Illustrative Case of the DART Router

Figure 8: **Gradient Misalignment and Router Behavior in DART. (A).** Gradient angle distributions under additional model–task settings, showing that gradients from the same capability are well aligned, while gradients between reasoning and tool-use tokens are largely orthogonal. **(B).** An illustrative example of the DART router, highlighting **rule-based** token-level routing decisions that distinguish reasoning, tool-use, and loss-free tokens during a tool-augmented QA process.

Metric	disentangled 2-Agent (LoRA)	DART (Multi-LoRA)
Backbone Instances	2	**1**
VRAM (Weight-dominant)	$\approx 2P$	$\approx$ **1P**
Context Switching Cost	High (Re-encoding $\mathcal{O}(L^2)$)	**Zero (KV-Cache Reuse)**

Table 7: Theoretical comparison between a disentangled 2-agent system and the DART framework. P denotes the backbone parameter count; L denotes sequence length.

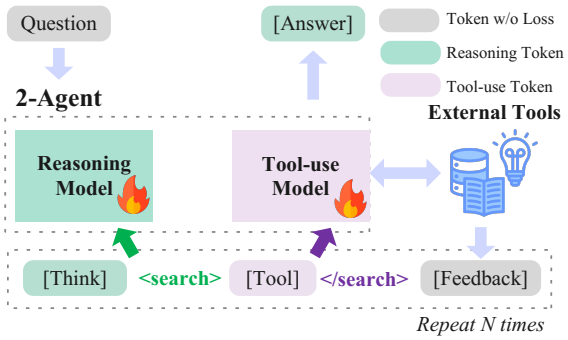


Figure 9: **2-Agent System Architecture.** A reasoning model and a tool-use model operate as separate models and interact through explicit handoffs. The reasoning model decides when to invoke tools, while the tool-use model executes tool calls and returns feedback.

of length L to build its own Key-Value (KV) cache. This re-computation has a complexity of $\mathcal{O}(L^2)$.

- **DART Latency:** Since DART operates on a single backbone, the KV-cache remains valid across capability switches. Moving from reasoning to tool-invocation only requires a negligible $\mathcal{O}(1)$ switch of the active LoRA ranks. The historical context is never re-processed, drastically reducing the Time-To-First-Token (TTFT) for subsequent turns.

As summarized in Tab. 7, DART simplifies the deployment stack. A 2-agent system requires an external orchestrator to synchronize states and format prompts between models, whereas DART internalizes this logic within a single inference pipeline.

G Effect of LoRA Rank in DART

We study the effect of the LoRA rank in DART by varying the adapter rank on the **Qwen2.5-3B-Base** model. Figure. 10(A) reports DART's EM performance on NQ and HotpotQA under different LoRA ranks (8/16/32) for both Qwen2.5-3B and Qwen2.5-7B backbones, with the 2-agent system shown as a reference. Overall, **DART is not strongly sensitive to the rank choice**: varying the rank changes EM only marginally, and the relative ordering across datasets and model scales remains consistent. Across all settings, DART stays close to the 2-agent baseline, indicating that *its improvements are not driven by simply increasing adapter capacity*. This is an interesting observation, which indicates that under the disentangled learning paradigm, a slight parameter capacity is enough to make the model completes the task well in practice.

H Retrieval Accuracy Evaluation

In section 6.2, we show that the single ability of DART is also improved, compared to the hybrid model. Next, we directly verify the search accuracy of DART model is improved, compared with

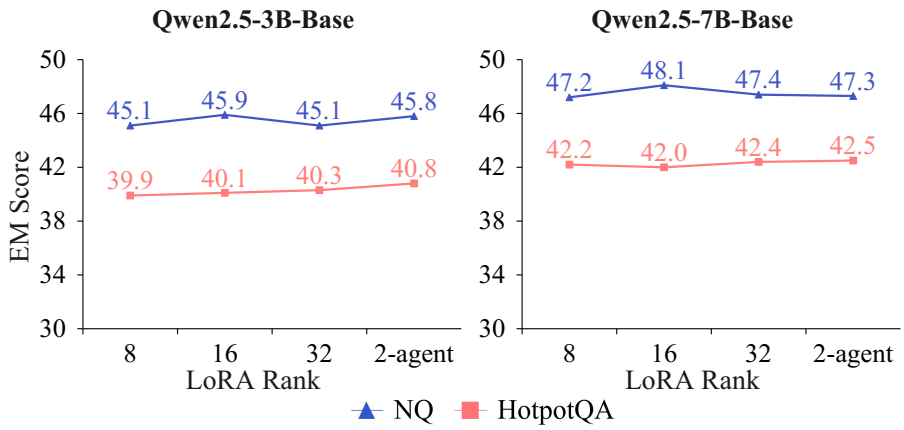


Figure 10: **LoRA Rank Sensitivity.** DART exhibits stable EM performance across LoRA ranks and remains close to the 2-agent baseline.

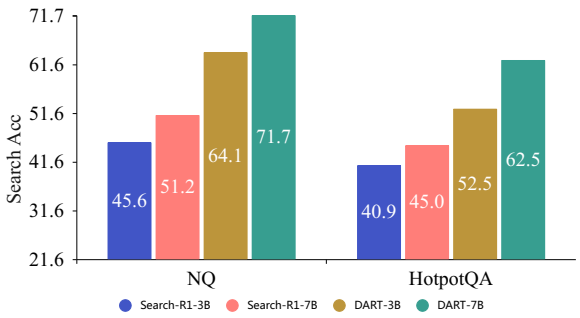


Figure 11: **Search Accuracy of DART.** DART consistently achieves higher search accuracy than Search-R1 across both datasets and model scales, and retrieval accuracy scales monotonically with model size.

baseline model. Concretely, we report the retrieval accuracy results and the corresponding evaluation protocol, which are presented exclusively here to analyze tool-use behavior under different training paradigms. We compare the jointly trained Search-R1 baseline with DART on the NQ and HotpotQA benchmarks, focusing on the model's ability to retrieve task-relevant information during inference.

We evaluate retrieval performance using *retrieval accuracy*. Let $\mathcal{S}$ denote the evaluation set. For each example $j \in \mathcal{S}$, the model retrieves a set of information documents or passages denoted by $\mathcal{D}_j$, and the ground-truth answer set is given by G_j. We define a retrieval correctness indicator $\mathrm{RetCorrect}(\mathcal{D}_j, G_j)$, which equals 1 if there exists at least one retrieved document in $\mathcal{D}_j$ that matches any element in G_j, and 0 otherwise. The overall retrieval accuracy is then defined as

$$\mathrm{Acc} = \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} \mathrm{RetCorrect}(\mathcal{D}_j, G_j).$$

We report retrieval accuracy for both Qwen2.5-3B and Qwen2.5-7B backbones under identical data splits and inference settings. Search-R1 optimizes reasoning and tool use jointly, whereas DART isolates their parameter updates during training. All methods share the same retrieval format and correctness criterion.

As shown in Figure 11, DART consistently achieves higher retrieval accuracy than Search-R1 across both datasets and model scales. This indicates that DART retrieves task-relevant information more reliably, particularly on multi-hop and fact-intensive tasks, highlighting the effectiveness of training-time capability disentanglement for tool use. As expected, retrieval accuracy scales consistently with model size: the 7B backbone outperforms the 3B backbone for both Search-R1 and DART.