# COLLABORATIVE LEARNING AND REASONING OVER MODEL SLICES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language models (LLMs) excel across diverse tasks, benefiting from strong knowledge priors, flexible in-context reasoning, and scalable test-time computation. Despite these strengths, single-model reasoning remains vulnerable to error propagation, as later steps depend on the model's earlier outputs, leaving reasoning blind spots difficult to independently reassess. To address this limitation, we develop *collaborative learning*, encompassing the construction of heterogeneous collaborators and the learning of effective cross-model collaboration. Specifically, *model slicing* replaces a large model with smaller, heterogeneous models under a comparable parameter budget, enabling prior reasoning to be reassessed across distinct model parameterizations. *Pyramid supervision* combines protocol distillation with step-level supervision to refine reasoning and interaction decisions throughout collaboration. We evaluate our approach across four task categories: multi-hop question answering, mathematical reasoning, code generation, and open-ended constrained generation. The trained heterogeneous models achieve comparable or better task performance than the original large model and established multi-agent baselines, including reinforcement-learning-based methods, while also achieving higher inference efficiency than the multi-agent baselines and exhibiting differentiated collaboration patterns across tasks. These findings show that learned collaboration turns model heterogeneity into complementary reasoning capabilities. Code is available at `https://anonymous.4open.science/r/collaborativelearning`.

## 1 INTRODUCTION

Large language models (LLMs) have demonstrated capabilities across tasks (Wang et al., 2024b). Advances highlight the benefits of scaling test-time computation (Snell et al., 2025; Muennighoff et al., 2025; Wu et al., 2025). Many approaches seek to improve task performance by allowing a single model to structure its reasoning process (Besta et al., 2024) and generate, inspect, and revise intermediate steps (Zhou et al., 2024; Zhang et al., 2024; Kumar et al., 2025). However, when the same model both generates and evaluates a solution, it may exhibit self-bias, favoring its own outputs and limiting its ability to identify and correct reasoning errors (Huang et al., 2024; Xu et al., 2024). Consequently, errors in intermediate steps may remain undetected and influence subsequent reasoning (Zheng et al., 2025). This motivates *reasoning-state isolation* across models, allowing prior reasoning to be independently reassessed under distinct model parameterizations rather than repeatedly evaluated within the same model's reasoning space (Du et al., 2024; Liang et al., 2024).

Accordingly, we develop a *collaborative learning* framework for reasoning and learning across heterogeneous models, as illustrated in Figure 1. Model slicing replaces a single large model with several smaller, heterogeneous models under a comparable total parameter budget (Liu et al., 2026b). Through explicit handoffs of task state between models (Chen et al., 2024c), prior reasoning can be independently reassessed under distinct model parameterizations, thereby enabling model-specific reasoning-state isolation while producing interaction trajectories that capture reasoning and interaction decisions. Together, this isolation and the resulting trajectories provide the structural basis for training heterogeneous models to leverage complementary capabilities (Wang et al., 2025; Abdulaal et al., 2025; Chen et al., 2025) when questioning, extending, or revising prior conclusions.



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

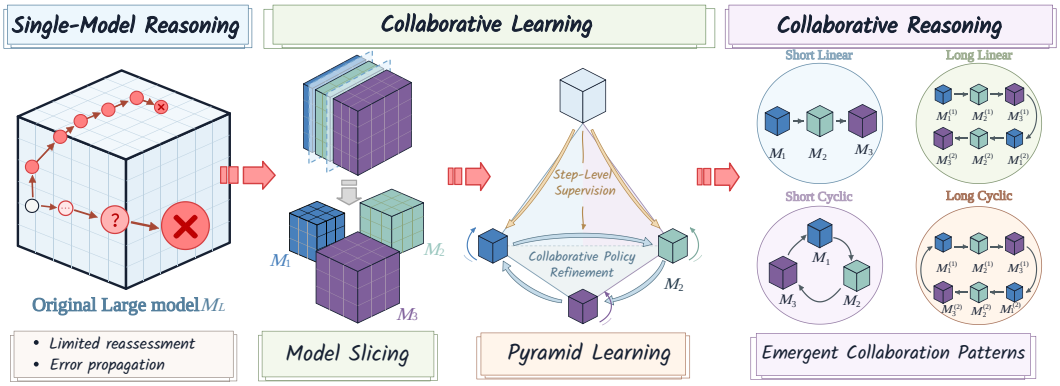


Figure 1: **Collaborative Learning and Reasoning over Model Slices.** Model slicing reallocates a single large model into multiple smaller, heterogeneous models under a comparable parameter budget, enabling reasoning to be reassessed across model parameterizations. Building on this, collaborative learning trains heterogeneous models through pyramid supervision, combining protocol distillation for interaction initialization with step-level supervision over collaboration trajectories[2].

Although this structure provides a foundation for cross-model collaboration, outcome-level supervision alone provides insufficient guidance for intermediate reasoning and interaction decisions throughout the collaborative process (Lightman et al., 2024; Wang et al., 2024a; Setlur et al., 2025). To address this limitation, we propose *pyramid supervision*, which uses supervisory signals from larger models to guide the heterogeneous models produced by model slicing toward effective cross-model collaboration. It combines interaction initialization with fine-grained supervision over reasoning and interaction decisions along model-generated collaboration trajectories, while integrating task-level rewards to jointly optimize intermediate decision quality and overall task performance. Through this training mechanism, heterogeneous models learn effective collaborative reasoning and develop differentiated interaction patterns across tasks.

We evaluate our approach across four task categories: multi-hop question answering, mathematical reasoning, code generation, and open-ended constrained generation. With a total parameter budget comparable to that of the original large model, the trained heterogeneous models collectively match or outperform the original model as well as established multi-agent system (MAS) and multi-agent reinforcement learning (MARL) baselines. Analysis of interaction trajectories further reveals reduced reasoning redundancy and differentiated collaboration patterns characterized by questioning, complementing, and revising prior reasoning. Together, these results suggest that collaborative reasoning across heterogeneous models can improve task performance while enabling complementary contributions and the reassessment of prior reasoning across distinct model parameterizations.

Our key contributions are summarized in three points:

- We propose a trainable paradigm for collaborative reasoning and learning across heterogeneous models, enabling independent reassessment across distinct model parameterizations and progressive learning of effective cross-model collaboration from interaction trajectories.
- Collaborative learning comprises model slicing and pyramid supervision. Model slicing constructs heterogeneous models under a comparable budget and enables reasoning-state isolation through task-state handoffs. Pyramid supervision combines interaction initialization with fine-grained supervision from the original large model over reasoning and interaction decisions.
- We validate collaborative learning across four task categories. The trained heterogeneous models achieve comparable or better performance than the original large model, established MAS baselines, and MARL methods, while demonstrating higher inference efficiency than MAS and MARL baselines and exhibiting emergent task-adaptive cross-model collaboration patterns.

## 2 METHOD

Collaborative reasoning characterizes an iterative reasoning process among heterogeneous models, where explicit task-state handoffs under a unified interaction protocol allow prior reasoning to be

[2]Short linear, long linear, short cyclic, and long cyclic trajectories are defined in Section 3.5.

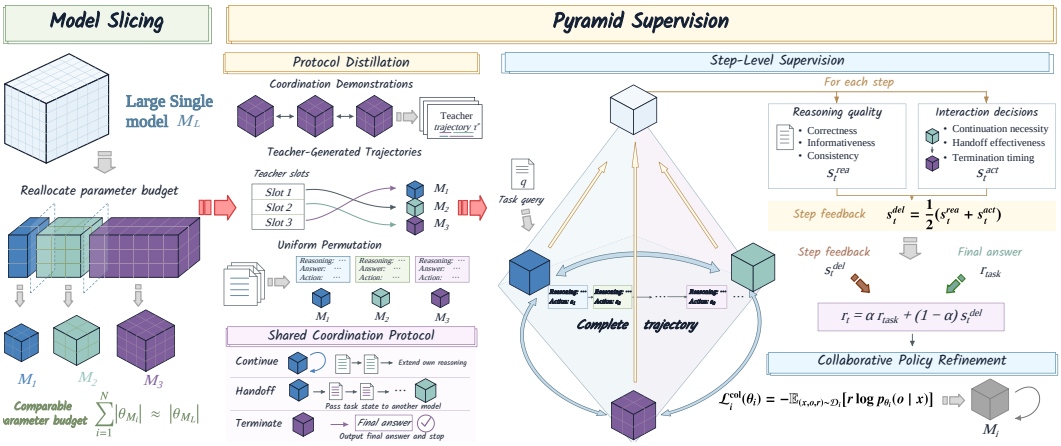


Figure 2: **Overview of collaborative learning.** (a) Model slicing constructs smaller, heterogeneous models under a comparable total parameter budget. (b) Pyramid supervision initializes collaboration through protocol distillation and refines it through step-level supervision. The original large model evaluates each step's reasoning and interaction decisions within the complete trajectory. Step-level scores and final task rewards jointly guide collaborative policy refinement.

continually reassessed by models with different parameters (▷ Section 2.1). To realize this process, collaborative learning constructs a heterogeneous model set through model slicing and learns effective cross-model collaboration through pyramid supervision (▷ Section 2.2).

### 2.1 Collaborative Reasoning

Collaborative reasoning refers to a process in which multiple models jointly construct a solution through successive contributions (Yin et al., 2023; Chen et al., 2024a). We define a unified interaction protocol that specifies how heterogeneous models generate reasoning and retain, transfer, or terminate control[3]. For a task input $q$, each interaction step $t$ is represented by

$$z_t = (i_t, x_t, y_t, a_t) \tag{1}$$

where $i_t$ denotes the active model $M_{i_t}$, $x_t$ contains the task input and preceding interaction history, $y_t$ is the current reasoning, and $a_t$ is the interaction action. The action space is defined as

$$a_t \in \{\texttt{continue}, \texttt{terminate}\} \cup \{\texttt{handoff}(j) \mid j \in \{1, \ldots, N\},\ j \neq i_t\} \tag{2}$$

where `continue` keeps control with the model, `handoff(j)` transfers the task state and control to $M_j$, and `terminate` ends collaborative reasoning. A complete trajectory is represented as

$$\tau = (z_1, z_2, \ldots, z_T) \tag{3}$$

where $T$ denotes the total number of interaction steps recorded in the complete trajectory.

Through `handoff`, models use distinct parameters to extend, question, or revise prior conclusions, enabling cross-model reassessment across the evolving reasoning process. The resulting trajectories record reasoning and interaction decisions for subsequent collaborative learning.

### 2.2 Collaborative Learning

Collaborative learning encompasses both the construction of heterogeneous collaborators and the learning of cross-model collaboration, with the pipeline illustrated in Figure 2. Model slicing constructs heterogeneous collaborators under a comparable parameter budget, providing the structural basis for cross-model reasoning. Pyramid supervision guides collaborative learning through protocol distillation and step-level supervision, progressively refining models' reasoning and interaction decisions while integrating task-level rewards throughout the collaborative reasoning process.

[3]Response fields and action-specific output requirements are detailed in Appendix C.

### 2.2.1 MODEL SLICING

Model slicing replaces a large model with smaller, independently parameterized models under a comparable parameter budget, each operating as a standalone model and interacting through explicit task-state handoffs. Let $M_L$ denote the large model with parameters $\theta_L$. We have

$$\mathcal{M} = \{M_1, \ldots, M_N\}, \qquad \sum_{i=1}^{N} |\theta_i| \approx |\theta_L| \tag{4}$$

where $\theta_i$ denotes the parameters of $M_i$, and $|\theta|$ denotes the parameter count.

These models differ in size and parameters and are not obtained by directly partitioning the weights of $M_L$. Together with task-state handoffs, this heterogeneous model set enables model-specific *reasoning-state isolation* across distinct model parameterizations during collaboration.

### 2.2.2 PYRAMID SUPERVISION

We introduce *pyramid supervision*, which refines collaboration through supervision across model scales. Protocol distillation uses the largest model slice to initialize cross-model interaction, while step-level supervision uses the original large model to evaluate collaboration trajectories.

**Protocol Distillation.** Although the interaction protocol specifies how models communicate and transfer control, heterogeneous models do not naturally learn to use it effectively as reasoning evolves. In particular, they must learn when to retain control, hand off the task state, or terminate based on the evolving reasoning context (Zeng et al., 2024). We therefore use protocol distillation to initialize cross-model interaction from teacher-generated trajectories (Chen et al., 2024b).

We select the largest model in $\mathcal{M}$ by parameter count as the teacher $M^\star$ and instantiate $N$ homogeneous copies to form the teacher system:

$$\mathcal{M}_{\text{teacher}} = \{M_1^\star, \ldots, M_N^\star\}, \qquad M_j^\star \equiv M^\star, \quad j = 1, \ldots, N \tag{5}$$

Following the same interaction protocol as the target models, the teacher system generates multi-step trajectories for distillation inputs $\mathcal{Q}_{\text{distill}} = \{q^{(m)}\}_{m=1}^{K}$:

$$\tau^{(m)} \sim P_{\mathcal{M}_{\text{teacher}}}\left(\tau \mid q^{(m)}\right), \qquad m = 1, \ldots, K \tag{6}$$

We retain qualified trajectories as demonstrations for subsequent distillation.

To avoid a fixed correspondence between teacher slots and target models, we sample a uniform permutation for each trajectory:

$$\pi^{(m)} \sim \text{Uniform}(S_N), \qquad \widetilde{i}_t^{(m)} = \pi^{(m)}\left(i_t^{(m)}\right), \quad t = 1, \ldots, T_m \tag{7}$$

where $S_N$ denotes the set of permutations over $\{1, \ldots, N\}$. Within each trajectory, the same permutation is applied consistently to all occurrences associated with each teacher slot, including model indices, handoff targets, and corresponding references in the interaction history. This operation changes only the correspondence between teacher slots and target models across trajectories, while preserving the reasoning content, step order, and handoff structure of the original trajectory.

Let $\widetilde{x}_t^{(m)}$ and $\widetilde{o}_t^{(m)}$ denote the reassigned context and response. Supervision covers the complete response, including reasoning and interaction decisions (Chen et al., 2023). For each model $M_i$, the reassigned context–response pairs satisfying $\widetilde{i}_t^{(m)} = i$ form its training set $\mathcal{D}_i^{\text{pro}}$. Following sequence-level knowledge distillation (Kim & Rush, 2016), each model minimizes

$$\mathcal{L}_i^{\text{pro}} = -\mathbb{E}_{(x,o)\sim\mathcal{D}_i^{\text{pro}}}\left[\log p_{\theta_i}(o \mid x)\right] \tag{8}$$

This objective initializes cross-model interaction capabilities in the heterogeneous models, providing a stable starting point for trajectory-level refinement of reasoning and interaction decisions.

**Step-Level Supervision.** After protocol distillation, the heterogeneous models perform collaborative reasoning under the shared interaction protocol, generating trajectories for subsequent optimization. For a given training task $q$, a collaborative trajectory is sampled as

$$\tau = (z_1, \ldots, z_T) \sim P_{\mathcal{M}}(\tau \mid q) \tag{9}$$

where $P_{\mathcal{M}}$ is induced by the current models and the shared interaction protocol.

Learning effective collaboration from these trajectories requires identifying the contribution of each reasoning and interaction decision to the overall process (Foerster et al., 2018). The value of an individual step depends not only on its local correctness but also on how it influences subsequent collaboration. We therefore retain the original large model $M_L$ as an evaluator $\mathcal{J}$, which assesses each step in the context of the complete trajectory. Given the task $q$ and trajectory $\tau$, the evaluator produces two complementary scores:

$$\left\{\left(s_t^{\mathrm{rea}}, s_t^{\mathrm{act}}\right)\right\}_{t=1}^{T} = \mathcal{J}(q, \tau), \qquad s_t^{\mathrm{rea}}, s_t^{\mathrm{act}} \in [-1, 1] \tag{10}$$

The reasoning score $s_t^{\mathrm{rea}}$ evaluates the model's reasoning contribution in terms of correctness, informativeness, and relevance to the task state and interaction history. The interaction score $s_t^{\mathrm{act}}$ evaluates whether its decision to continue, hand off the task state, or terminate appropriately advances the collaborative process. Separating these dimensions avoids conflating reasoning quality with interaction quality. We average the two scores to obtain a step-level score:

$$s_t^{\mathrm{del}} = \frac{1}{2}\left(s_t^{\mathrm{rea}} + s_t^{\mathrm{act}}\right) \tag{11}$$

*Reward integration.* Step-level scores provide fine-grained supervision for intermediate reasoning and interaction decisions, but local decision quality does not necessarily translate into successful task completion. To provide decision-specific feedback while also accounting for the overall task objective, we integrate each step-level score with the final task reward. Let $m(q, \hat{y}) \in [0, 1]$ denote the task-specific evaluation metric for the final prediction $\hat{y}$. We map it to the same $[-1, 1]$ range as the step-level score and define

$$r_{\mathrm{task}} = 2m(q, \hat{y}) - 1, \qquad r_t = \alpha r_{\mathrm{task}} + (1 - \alpha) s_t^{\mathrm{del}} \tag{12}$$

where $\alpha \in [0, 1]$ controls the trade-off between the task-level reward and the step-level score. The resulting reward distinguishes the contributions of intermediate steps across the complete collaborative trajectory while aligning their optimization with the final task objective.

*Collaborative Policy Refinement.* Using the combined rewards, we extract the steps at which each model was active from the collaborative trajectories and construct model-specific training data. For model $M_i$, the corresponding training examples are collected as

$$\mathcal{D}_i = \left\{(x_t^{(m)}, o_t^{(m)}, r_t^{(m)}) \mid i_t^{(m)} = i\right\} \tag{13}$$

where $o_t^{(m)}$ denotes the complete response at step $t$ of trajectory $m$, serialized under the shared interaction protocol and including its reasoning, answer, and interaction fields. Following the score-function policy-gradient form, we optimize each model with the reward-weighted objective

$$\mathcal{L}_i^{\mathrm{col}}(\theta_i) = -\mathbb{E}_{(x,o,r)\sim\mathcal{D}_i}\left[r \log p_{\theta_i}(o \mid x)\right] \tag{14}$$

This objective increases the likelihood of responses assigned positive rewards and decreases the likelihood of those assigned negative rewards. Because each reward combines step-level evaluation conditioned on the complete trajectory with the final task outcome, each model can adapt its policy according to how its own decisions contribute to the collaborative result, allowing differentiated collaboration patterns to progressively emerge through training (Cui et al., 2026).

# 3 Experiments

## 3.1 Experimental Setup

**Datasets.** We use five benchmarks across four task categories: multi-hop question answering (MuSiQue (Trivedi et al., 2022)), mathematical reasoning (GSM-Hard (Gao et al., 2023) and MATH (Hendrycks et al., 2021)), code generation (an eight-language MultiPL-E subset (Cassano et al., 2023))[4], and open-ended constrained generation (Conifer (Sun et al., 2024)).

[4]Following the language selection in the Qwen3 technical report (Yang et al., 2025a), we restrict code-generation evaluation to Python, C++, Java, PHP, TypeScript, C#, Bash, and JavaScript.

**Baselines.** (1) Single-agent baselines: Qwen3-14B (Yang et al., 2025a) and its self-evaluated RL variant (SE-RL)[5]. (2) A homogeneous MAS of three Qwen3-8B models (24B total) without collaborative training, serving as a higher-capacity multi-model baseline. (3) A homogeneous collaborative-learning system (Homo. `CL`) of three Qwen3-4B models (12B total), trained with the same procedure as our heterogeneous system to assess the effect of model heterogeneity in collaborative reasoning. (4) Established multi-agent frameworks: MAD (Du et al., 2024), AgentVerse (Chen et al., 2024c), AFlow (Zhang et al., 2025), and GPTSwarm (Zhuge et al., 2024). (5) MARL methods: MAGRPO (Liu et al., 2026b), MAPoRL (Park et al., 2025), and AT-GRPO (Zhao et al., 2026b).

**Metrics.** We report accuracy and F1 on MuSiQue (Trivedi et al., 2022), GSM-Hard (Gao et al., 2023), and MATH (Hendrycks et al., 2021). We use weighted pass@1 and macro-language pass@1 for MultiPL-E (Cassano et al., 2023), and Coverage and Explicit for Conifer (Sun et al., 2024)[6]. Beyond task performance, we analyze handoffs, anomaly rates, reasoning redundancy, and model-wise behavioral distributions to characterize training-induced changes in collaboration patterns.

**Implementation Details.** Model slicing replaces Qwen3-14B ($M_L$) (Yang et al., 2025a) with Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (13.7B total). The three collaborating models share the same prompt and action space. Three Qwen3-8B instances generate protocol distillation demonstrations, while Qwen3-14B evaluates reasoning and interaction decisions within complete trajectories for step-level supervision. Experiments run on 8× NVIDIA A800-SXM4-80GB GPUs. Each model is trained with a rank-64 LoRA adapter (Hu et al., 2022) while its base parameters remain frozen[7].

### 3.2 Does Collaborative Learning Lead to Improved Performance?

Table 1: **Performance comparison with baseline methods.** Scores are percentages (%); higher is better. Acc. denotes accuracy; Wtd. and Mac. denote weighted and macro-language pass@1, respectively; Cov. and Expl. denote Coverage and Explicit, respectively. `CL.` denotes collaborative learning. Best and second-best results are shown in **bold** and underline, respectively. ↑ and ↓ indicate scores above and below the corresponding baseline mean for each metric in the table. † indicates that the corresponding baseline differs significantly from `CL.` ($p \leq 0.05$).

| Method | MuSiQue | | GSM-Hard | | MATH | | MultiPL-E | | Conifer | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | F1 | Acc. | F1 | Acc. | F1 | Wtd. | Mac. | Cov. | Expl. |
| Qwen3-14B | 39.55† ↑0.87 | 49.70† ↑0.47 | 68.94 ↑1.65 | 80.69† ↑3.19 | 74.60† ↑4.40 | 82.86† ↑3.25 | 55.18† ↑1.08 | 57.34† ↑2.59 | 88.00† ↑0.17 | 95.72 ↑0.62 |
| SE-RL | 41.75 ↑3.07 | 52.71† ↑3.48 | 68.18 ↑0.89 | 79.67† ↑2.17 | 76.00 ↑5.80 | 80.70 ↑1.09 | **60.43**† ↑6.33 | **62.55**† ↑7.80 | 88.00 ↑0.17 | 95.81 ↑0.71 |
| Homo. MAS | 26.40† ↓12.28 | 32.82 ↓16.41 | 71.21 ↑3.92 | 79.93† ↑2.43 | 68.20† ↓2.00 | 76.41† ↓3.20 | 56.36† ↑2.26 | 57.20 ↑2.45 | 88.02† ↑0.19 | 94.27† ↓0.83 |
| Homo. `CL.` | 36.00† ↓2.68 | 47.19† ↓2.04 | 60.61† ↓6.68 | 76.51 ↓0.99 | 68.60† ↓1.60 | 77.95 ↓1.66 | 53.33† ↓0.77 | 52.65† ↓2.10 | 88.15 ↑0.32 | 95.22† ↑0.12 |
| MAD | 39.51† ↑0.83 | 50.59† ↑1.36 | 66.67† ↓0.62 | 74.18 ↓3.32 | 72.40† ↑2.20 | 81.32 ↑1.71 | 55.03† ↑0.93 | 55.35† ↑0.60 | 88.55 ↑0.72 | 95.84 ↑0.74 |
| AgentVerse | 41.99 ↑3.31 | 52.85† ↑3.62 | **71.97** ↑4.68 | 78.44† ↑0.94 | 68.20† ↓2.00 | 79.42 ↓0.19 | 59.25 ↑5.15 | 59.83† ↑5.08 | 87.74† ↓0.09 | 95.86† ↑0.76 |
| AFlow | 41.83† ↑3.15 | 50.36† ↑1.13 | 68.18 ↑0.89 | 76.04 ↓1.46 | 65.00† ↓5.20 | 76.53† ↓3.08 | 42.83 ↓11.27 | 42.46† ↓12.29 | 87.92† ↑0.09 | **96.51**† ↑1.41 |
| GPTSwarm | 36.20† ↓2.48 | 47.37† ↓1.86 | 70.45 ↑3.16 | 78.39† ↑0.89 | 69.20† ↓1.00 | 79.90 ↑0.29 | 51.63† ↓2.47 | 52.43 ↓2.32 | 88.17† ↑0.34 | 95.96† ↑0.86 |
| MAGRPO | 40.13† ↑1.45 | 51.74 ↑2.51 | 65.91 ↓1.38 | 76.44 ↓1.06 | 65.00† ↓5.20 | 76.01† ↓3.60 | 49.93 ↓4.17 | 50.48† ↓4.27 | 88.58† ↑0.75 | 95.82† ↑0.72 |
| MAPoRL | 40.59† ↑1.91 | 52.74† ↑3.51 | 63.64† ↓3.65 | 75.34† ↓2.16 | 70.80† ↑0.60 | 80.87† ↑1.26 | 53.25† ↓0.85 | 53.27† ↓1.48 | 86.37† ↓1.46 | 89.19† ↓5.91 |
| AT-GRPO | 41.54 ↑2.86 | **53.51**† ↑4.28 | 64.39† ↓2.90 | 76.86 ↓0.64 | 74.20† ↑4.00 | 83.70† ↑4.09 | 57.91† ↑3.81 | 58.74† ↑3.99 | 86.60† ↓1.23 | 95.85† ↑0.75 |
| `CL.` | **42.37** ↑3.69 | 53.43 ↑4.20 | **71.97** ↑4.68 | **81.15** ↑3.65 | **78.40** ↑8.20 | **85.25** ↑5.64 | 59.39 ↑5.29 | 59.63 ↑4.88 | **88.90** ↑1.07 | 95.90 ↑0.80 |

As shown in Table 1, **collaborative learning achieves the highest average performance across five benchmarks, with best or joint-best results on most metrics**. With a total parameter budget of 13.7B, it outperforms the original Qwen3-14B on all ten metrics, with an average absolute gain of 2.38%, and surpasses SE-RL on eight. It also outperforms the 24B homogeneous MAS without collaborative training on all ten metrics, despite using fewer parameters. Compared with Homo. CL, which trains homogeneous models using the same procedure, the heterogeneous collaborative system achieves higher scores across all ten metrics, with an average absolute gain of 6.02%. Its average absolute gains over established MAS and MARL baselines range from 2.08% to 6.87% across the ten metrics, demonstrating competitive performance against both classes.

[5] SE-RL uses Qwen3-14B self-feedback for comparison with cross-model feedback; see Appendix E.

[6] Definitions and computation details for the Conifer metrics are provided in Appendix E.11.

[7] See Appendix D for data, training, and inference details, and Appendix E for baseline implementations.

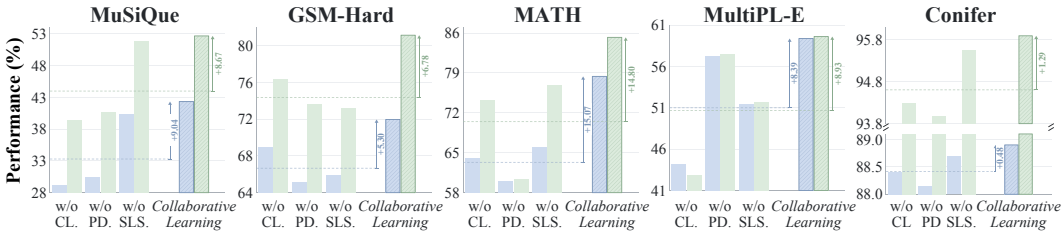


Figure 3: **Ablation study of the mechanism across five benchmarks.** We compare a zero-shot variant that receives no training (w/o CL.), a variant without Protocol Distillation (w/o PD.), and a variant without the step-level score (w/o SLS.). Arrows indicate the gain of *collaborative learning* over the arithmetic mean of the available variants. For each benchmark, denotes Acc., Wtd., and Cov., while denotes the corresponding second metric (F1, Mac., or Expl.).

Trajectory analysis reveals task-specific characteristics of collaboration and how intermediate behaviors translate into final outcomes. On MuSiQue, subsequent interactions correct initial errors, while on MATH, the trained heterogeneous models better preserve correct intermediate answers than MA-PoRL. On MuSiQue and GSM-Hard, differences in partial-match scores on questions missed by both the collaborative system and the baseline help explain why accuracy and F1 rankings do not fully align. Code-generation performance varies across task subsets: the collaborative system solves more MBPP tasks but fewer HumanEval tasks than SE-RL. On most tasks solved only by SE-RL, it produces no passing intermediate program, suggesting that this gap primarily reflects failure to generate correct solutions rather than the loss of correct candidates during subsequent interactions. On Conifer, the collaborative system achieves the highest Coverage and almost always maintains compliance in the final output when an intermediate answer already satisfies all constraints.

### 3.3 How Much Does Each Mechanism Contribute to Performance?

As shown in Figure 3, we compare three ablation variants: without collaborative training (w/o CL.), without protocol distillation (w/o PD.), and without the step-level score (w/o SLS.). Averaged over the ten reported metrics, these variants incur absolute score drops of 9.37, 8.85, and 5.41 points, respectively, and each underperforms the full method on every metric. These results indicate that **heterogeneity alone is insufficient to achieve the full performance gains, and that both protocol distillation and step-level scoring contribute meaningfully to learned collaboration**.

Beyond aggregate performance, trajectory-level behavior provides a more direct view of how the two training components shape collaboration throughout collaborative reasoning. For example, without protocol distillation, first-answer EM on MuSiQue increases from 21.02% to 25.24%, yet the number of initially incorrect answers corrected by the final response falls from 594 to 171. Meanwhile, the proportion of Conifer trajectories that initially satisfy all constraints but fail at termination increases, and protocol-following failures on MATH also become more frequent. These observations suggest that protocol distillation primarily supports effective downstream interaction rather than simply improving the quality of the initial response. Similarly, removing the step-level score increases the average MuSiQue trajectory length from 3.4 to 3.8 steps while reducing the number of corrected answers by 18.5%, indicating that step-level feedback helps models use interaction steps more productively and achieve more corrections with fewer interactions on this task.

### 3.4 How Efficient and Effective Is Collaborative Reasoning?

Alongside improvements in task performance, collaborative learning leads to more efficient inference. As shown in Figure 4, the collaborative system requires fewer model invocations than MAD, AgentVerse, AFlow, and GPTSwarm across all five benchmarks, with reductions ranging from 39.1% to 84.8%. Specifically, it requires an average of 4.0, 4.1, 1.8, 3.0, and 3.3 invocations per question on MuSiQue, GSM-Hard, MultiPL-E, MATH, and Conifer, respectively. This efficiency advantage also extends to the evaluated MARL baselines, with invocation reductions ranging from 46.7% to 86.0%. In terms of parameter-weighted output tokens[8], **the collaborative system achieves the lowest count among the compared methods on four of five benchmarks**. Compared with the MARL baselines, it further reduces parameter-weighted output tokens by 8.2%–89.4%.

[8]The parameter-weighted output-token metric is detailed in Appendix F.1.

We further compare reasoning-overlap rates before and after collaborative learning to examine redundancy across successive reasoning steps. On MuSiQue, the overlap rates of the three collaborating models decrease from 48.3%, 24.1%, and 55.2% to 0.2%, 0.3%, and 0.5%, respectively. On MultiPL-E, they decrease from 34.8%, 2.3%, and 17.6% to 7.4%, 1.8%, and 10.7%. These results suggest that **collaborative learning substantially reduces repetitive reasoning across model interactions, yielding less redundant subsequent contributions throughout the process**.

To further assess collaborative effectiveness, we measure correct-answer retention, defined as the proportion of trajectories whose final answer remains correct among those containing a correct intermediate candidate. On MuSiQue, GSM-Hard, MultiPL-E, and MATH, the trained heterogeneous models achieve 80.42%, 97.94%, 98.26%, and 98.00% retention, respectively, ranking first on the latter two tasks, second on GSM-Hard, and third on MuSiQue among comparison methods with recoverable intermediate candidates[9]. These results show that **the trained models generally preserve available correct intermediate answers throughout subsequent collaborative interactions**.

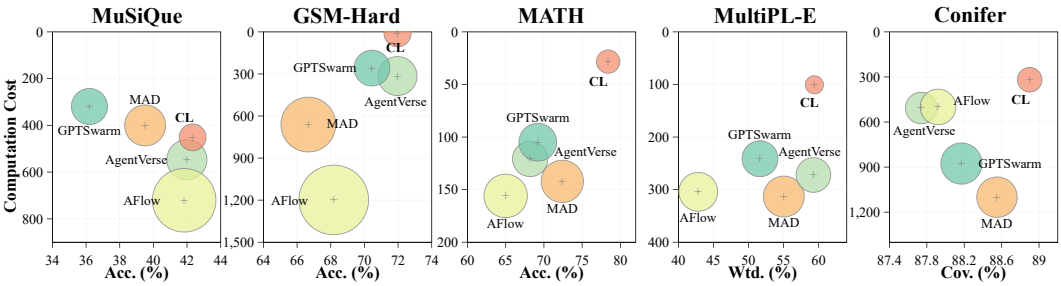


Figure 4: **Computation cost versus task performance.** The center of each circle indicates task performance and parameter-weighted output tokens for the collaborative system and four multi-agent baselines across five benchmarks; its area is proportional to the number of model invocations.

### 3.5 What Collaboration Patterns Emerge from Collaborative Learning?

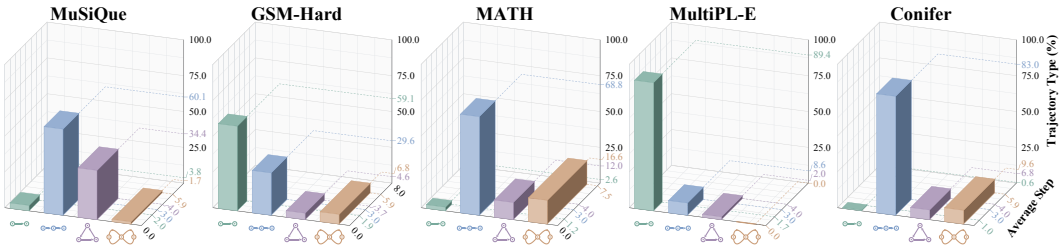


Figure 5: **Post-training interaction topologies across tasks.** , , , and denote short linear, long linear, short cyclic, and long cyclic trajectories, respectively. Bar height indicates the proportion of each trajectory type, while depth indicates the average number of interaction steps.

To examine how collaborative learning reorganizes model interactions across tasks, we compare trajectory topologies and interaction lengths before and after training. We categorize trajectories as short linear, long linear, short cyclic, or long cyclic. Linear trajectories involve each model at most once, with at most two interaction steps considered short; cyclic trajectories revisit a previously participating model, with at most four steps considered short. On MuSiQue, the proportion of long linear trajectories increases from 9.18% to 60.12%, showing a shift toward longer linear interactions. In contrast, this proportion decreases from 10.95% to 8.58% on MultiPL-E. Short cyclic trajectories evolve differently across tasks: their average interaction length decreases from 4.00 to 3.67 steps on GSM-Hard, but increases from 3.33 to 4.00 on MATH. These results show that collaborative learning does not uniformly increase interaction complexity or length, but instead induces task-dependent collaboration topologies rather than converging to a single universal interaction pattern.

**Task-dependent functional specialization also emerges across models during collaborative learning.** On MuSiQue, Qwen3-4B's answer-revision rate increases from 13% to 69%, its hand-off rate from 48% to 96%, and its termination rate decreases from 50% to 4%, indicating a shift

[9]Conifer is excluded because its open-ended outputs lack an intermediate-candidate correctness criterion.

toward intermediate revision and handoff. Meanwhile, Qwen3-8B increasingly provides the final answer, with its share rising from 13.0% to 58.4%. This division of labor changes across tasks: on GSM-Hard, Qwen3-4B instead becomes the more frequent final-answer provider, with its share increasing from 34.09% to 64.39%. Thus, the same model can assume different functions across tasks, revealing task-dependent divisions of labor rather than a fixed collaboration structure[10].

## 4 Related Work

**Reasoning in LLMs.** Existing work improves LLM reasoning by eliciting existing capabilities or strengthening them through training (Zhao et al., 2026a; Li et al., 2026c; Liu et al., 2026a). Inference-time methods use chain-of-thought prompting (Wei et al., 2022), systematic exploration (Hariri et al., 2026; Li et al., 2026a), evaluation (Wu & Raghunathan, 2026), and revision (Yang et al., 2026a); training-based methods learn from bootstrapped demonstrations (Zelikman et al., 2022; Dong et al., 2026), distill teacher reasoning (Yang et al., 2025b; Liu et al., 2026c; Fu et al., 2026), or use outcome- and process-level supervision. Yet these methods remain bounded by a single model's knowledge and reasoning biases, leaving blind spots unaddressed and allowing intermediate errors to persist through subsequent reasoning (Xie et al., 2026).

**From Single-Model Reasoning to Multi-Model Composition.** These limitations motivate moving beyond single-model reasoning toward multi-model composition, which exploits complementary knowledge and task-specific strengths across models (Yang et al., 2026b; Zhang et al., 2026c; Huot et al., 2026). Existing approaches select models suited to an input (Ong et al., 2025; Cao et al., 2026; Feng et al., 2026b; Fisch et al., 2026), distribute tasks or subtasks (Agashe et al., 2025; Chen et al., 2026), organize sequential model invocations (Dekoninck et al., 2025; Li et al., 2026b), determine when additional models are needed (Rajib et al., 2026; Qin et al., 2026), and aggregate candidate outputs (Zhang et al., 2026b). However, system-level composition does not necessarily constitute collaborative reasoning in which models jointly develop, evaluate, and revise a shared solution.

**Multi-Model Collaborative Reasoning.** Collaborative reasoning directly addresses how models jointly develop solutions through interaction. Existing approaches range from hand-designed roles and protocols to optimized workflows (Hu et al., 2025; Zhou et al., 2026; Lee et al., 2026; Koh et al., 2026) and learned collaborative policies (Xue et al., 2026; Zhang et al., 2026a). Representative systems, however, often retain an explicitly prescribed division of labor (Zhao et al., 2026b; Feng et al., 2026a; Ding et al., 2026). In contrast, we train heterogeneous small models to collaborate without predefined roles under a total parameter budget comparable to that of a single large model.

## 5 Conclusion

We introduced a collaborative learning framework for reasoning across heterogeneous models. Model slicing replaces a large model with smaller heterogeneous models under a comparable parameter budget, enabling reasoning-state isolation through task-state handoffs. Pyramid supervision enables cross-model collaboration by combining protocol distillation for interaction initialization with step-level supervision over model-generated trajectories. Across four task categories, the trained heterogeneous models match or outperform the large model and MAS and MARL baselines, while requiring fewer model invocations than multi-agent frameworks. Analysis reveals reduced reasoning redundancy, stronger correct-answer retention, and task-dependent interaction topologies and functional specialization. These results suggest that collaborative learning can transform model heterogeneity into complementary reasoning capabilities across distinct model parameterizations.

## 6 Discussion

Our study covers three model sizes and five benchmarks, leaving generalization to larger model pools, longer interaction horizons, and unseen task distributions unexplored. Pyramid supervision depends on inspector and evaluator quality, whose biases may affect step-level feedback. Comparable parameter counts do not imply equal computational cost, as collaborative inference adds latency and memory overhead beyond our parameter-weighted output-token metric. Future work will study generalization and robust supervision through multiple inspectors or uncertainty-aware aggregation.

[10]Cases in Appendix G illustrate how models collaborate through different interaction patterns across tasks.

## AI USE STATEMENT

We used AI tools to assist with drafting and polishing the manuscript, writing portions of the experimental code, preparing figures, and searching for literature. The authors reviewed and revised the AI-assisted text and figures, and modified, verified, and tested the AI-assisted code. For the literature, AI tools served only as an aid to searching; the authors independently selected, reviewed, organized, and synthesized the relevant publications. We take full responsibility for the content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

## ETHICS STATEMENT

Our experiments use existing research benchmarks and pretrained language models and do not involve the collection of new human-subject data or sensitive personal information. As the models and the model-based inspector may inherit biases or errors from their training data, our results should not be interpreted as evidence of safety or fairness in deployment. Applications in high-stakes settings would require additional domain-specific evaluation and appropriate human oversight.

## REPRODUCIBILITY STATEMENT

We provide the full training code, benchmark scorers, result records, configurations, and analysis artifacts in the anonymous repository at `https://anonymous.4open.science/r/collaborativelearning`. The main text specifies models, datasets, baselines, metrics, and hardware. Appendix B presents the training procedure; Appendix D documents rewards, optimization, and LoRA settings; and Appendix E details baselines and the comparison protocol. The appendix also describes interaction schema, inspector rubrics, task evaluators, efficiency metrics, and behavioral analyses. These materials enable reproduction. Since generation is stochastic, outputs may vary across runs; we recommend comparing aggregate metrics and qualitative trends.

## REFERENCES

Ahmed Abdulaal, Chen Jin, Nina Montaña Brown, Aryo Pradipta Gema, Daniel Castro, Daniel Alexander, Philip Teare, Tom Diethe, Dino Oglic, and Amrutha Saseendran. Balancing act: Diversity and consistency in large language model ensembles. In *International Conference on Learning Representations (ICLR)*, 2025. `https://proceedings.iclr.cc/paper_files/paper/2025/hash/a5554e55d7a21e62d0d7a028ec0ea1c7-Abstract-Conference.html`.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. In *Conference on Language Modeling (COLM)*, 2025. `https://openreview.net/forum?id=zg5is4GJ3R`.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024. `https://ojs.aaai.org/index.php/AAAI/article/view/29720`.

Qi Cao, Shuhao Zhang, Ruizhe Zhou, Ruiyi Zhang, Peijia Qin, and Pengtao Xie. Models under SCOPE: Scalable and controllable routing via pre-hoc reasoning. In *International Conference on Machine Learning (ICML)*, 2026. `https://arxiv.org/abs/2601.22323`.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. MultiPL-E: A scalable and polyglot approach to benchmarking neural code generation. In *IEEE Transactions on Software Engineering (TSE)*, 2023. `https://doi.org/10.1109/TSE.2023.3267446`.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. In *arXiv preprint arXiv:2310.05915*, 2023. https://arxiv.org/abs/2310.05915.

Guanzhong Chen, Shaoxiong Yang, Chao Li, Wei Liu, Jian Luan, and Zenglin Xu. End-to-end optimization of LLM-driven multi-agent search systems via heterogeneous-group-based reinforcement learning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. https://aclanthology.org/2026.acl-long.1399/.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a. https://aclanthology.org/2024.acl-long.381/.

Justin Chen, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. MAGDi: Structured distillation of multi-agent interaction graphs improves reasoning in smaller language models. In *International Conference on Machine Learning (ICML)*, 2024b. https://proceedings.mlr.press/v235/chen24ah.html.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations (ICLR)*, 2024c. https://proceedings.iclr.cc/paper_files/paper/2024/hash/578e65cdee35d00c708d4c64bce32971-Abstract-Conference.html.

Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *International Conference on Learning Representations (ICLR)*, 2025. https://openreview.net/forum?id=o1Et3MogPw.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. In *Transactions on Machine Learning Research (TMLR)*, 2026. https://openreview.net/forum?id=9SkkifLopZ.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *International Conference on Machine Learning (ICML)*, 2025. https://proceedings.mlr.press/v267/dekoninck25a.html.

Jie Ding, Rui Sun, Xinyuan Zhang, Zeyu Zhang, and Xin Liu. APEx: Distillation of agent procedural experience for adaptive deep research question answering. In *arXiv preprint arXiv:2609.02253*, 2026. https://arxiv.org/abs/2609.02253.

Leichao Dong, Dongxu Zhang, Yiding Sun, Qirui Wang, Yuhan Wang, Lin Chen, and Jihua Zhu. Better starts, better ends: Bootstrapped iterative self-reasoning distillation for compressed reasoning. In *arXiv preprint arXiv:2607.15736*, 2026. https://arxiv.org/abs/2607.15736.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, 2024. https://proceedings.mlr.press/v235/du24e.html.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: Stable reinforcement learning for multi-agent LLM systems. In *arXiv preprint arXiv:2602.08847*, 2026a. https://arxiv.org/abs/2602.08847.

Tao Feng, Fangxu Yu, Haozhen Zhang, Zhongjie Dai, Liangqi Yuan, Zijie Lei, Weizhi Zhang, Kunlun Zhu, Haodong Yue, Keyang Xuan, Ge Liu, and Jiaxuan You. LLMRouter: Unified infrastructure for developing, evaluating, and deploying LLM routers. In *arXiv preprint arXiv:2608.06867*, 2026b. https://arxiv.org/abs/2608.06867.

Adam Fisch, Shubhendu Trivedi, Fantine Huot, William W. Cohen, Michael Kaisers, Mirella Lapata, Kate Larson, and Jacob Eisenstein. Pandora's AI model routing box: Efficient allocation with costly value estimation. In *arXiv preprint arXiv:2608.20316*, 2026. https://arxiv.org/abs/2608.20316.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. https://ojs.aaai.org/index.php/AAAI/article/view/11794.

Zixuan Fu, Bingxiang He, Yuxin Zuo, Haohuan Huang, Jinqian Zhang, Ruhang Xiao, Cheng Qian, Qinyu Luo, Huan-ang Gao, Yudong Wang, Zhiyuan Liu, Ning Ding, and Chaojun Xiao. Rethinking on-policy distillation of large language models II: One training example. In *arXiv preprint arXiv:2609.04172*, 2026. https://arxiv.org/abs/2609.04172.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023. https://proceedings.mlr.press/v202/gao23f.html.

Mohsen Hariri, Weicong Chen, Nahal Shahini, Vikash Singh, Kai Ye, Amirhossein Samandar, Debargha Ganguly, Sreehari Sankar, Yanyan Zhang, Shouren Wang, Jerry Peng, Biyao Zhang, Michael Hinczewski, and Vipin Chaudhary. Test-time scaling in reasoning LLMs: Inference regimes, evaluation, and reproducibility. In *arXiv preprint arXiv:2608.04001*, 2026. https://arxiv.org/abs/2608.04001.

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. In *Conference on Neural Information Processing Systems, Datasets and Benchmarks Track (NeurIPS)*, 2021. https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. https://openreview.net/forum?id=nZeVKeeFYf9.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024. https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Fantine Huot, Michael Kaisers, and Mirella Lapata. When is routing meaningful? diversity and robustness in language model societies. In *arXiv preprint arXiv:2607.09197*, 2026. https://arxiv.org/abs/2607.09197.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. https://aclanthology.org/D16-1139/.

Donghyeok Koh, Gyuwan Kim, Jinyeong Bak, Seung-Hoon Na, Tao Yang, Haneol Jang, and Cheoneum Park. Global optimization and inference-time region grafting for agentic workflows. In *arXiv preprint arXiv:2608.02353*, 2026. https://arxiv.org/abs/2608.02353.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha

Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Jung Hwan Lee, Kyu Ho Lee, and Gwang Hoon Yoo. MEGA: Self-evolving agent optimization infrastructure via wisdom graph. In *arXiv preprint arXiv:2608.10504*, 2026. https://arxiv.org/abs/2608.10504.

Muheng Li, Jian Qian, and Wenlong Mou. Predicting and improving test-time scaling laws via reward tail-guided search. In *arXiv preprint arXiv:2602.01485*, 2026a. https://arxiv.org/abs/2602.01485.

Songyuan Li, Ahmed M. Abdelmoniem, and Shiqiang Wang. ProgRouter: Online progress-guided orchestration for multi-agent LLM workflows under quality-cost tradeoffs. In *Findings of the Association for Computational Linguistics: Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026b. https://arxiv.org/abs/2608.25992.

Yang Li, Semih Yavuz, and Shafiq Joty. RISE: Recursive improvement via self-extrapolating policy distillation. In *arXiv preprint arXiv:2609.05295*, 2026c. https://arxiv.org/abs/2609.05295.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. https://aclanthology.org/2024.emnlp-main.992/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, Mickel Liu, Hanfei Yu, Zhaorun Chen, Weiyan Shi, Paul Pu Liang, Luke Zettlemoyer, Yejin Choi, and Natasha Jaques. SPADE: Self-play in adaptive synthetic executable environments. In *arXiv preprint arXiv:2608.19197*, 2026a. https://arxiv.org/abs/2608.19197.

Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026b. https://ojs.aaai.org/index.php/AAAI/article/view/40487.

Zhishuai Liu, Xingzi Xu, Mehmet Saygin Seyfioglu, Pan Xu, and Karim Bouyarmane. Extremely sparse supervision incentivizes reasoning ability. In *arXiv preprint arXiv:2609.04565*, 2026c. https://arxiv.org/abs/2609.04565.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025. https://aclanthology.org/2025.emnlp-main.1025/.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. https://aclanthology.org/2025.acl-long.1459/.

Peijia Qin, Qi Cao, and Pengtao Xie. ATLAS: Agentic test-time learning-to-allocate scaling. In *arXiv preprint arXiv:2606.01667*, 2026. https://arxiv.org/abs/2606.01667.

Rakibul Hasan Rajib, Mengxin Zheng, and Qian Lou. Learning what to retain: Gated-memory routing for efficient collaboration in multi-agent LLM systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. https://arxiv.org/abs/2609.00237.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. Conifer: Improving complex constrained instruction-following ability of large language models. In *arXiv preprint arXiv:2404.02823*, 2024. https://arxiv.org/abs/2404.02823.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. In *Transactions of the Association for Computational Linguistics (TACL)*, 2022. https://aclanthology.org/2022.tacl-1.31/.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y. Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/5434be94e82c54327bb9dcaf7fca52b6-Abstract-Conference.html.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a. https://aclanthology.org/2024.acl-long.510/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Conference on Neural Information Processing Systems, Datasets and Benchmarks Track (NeurIPS)*, 2024b. https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, 1992. https://doi.org/10.1007/BF00992696.

Chen Henry Wu and Aditi Raghunathan. Self-trained verification for training- and test-time self-improvement. In *arXiv preprint arXiv:2605.30290*, 2026. https://arxiv.org/abs/2605.30290.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/8c3caae2f725c8e2a55ecd600563d172-Abstract-Conference.html.

Si'an Xie, Jiaxun Liu, Biao Yang, Wei Yuan, Fan Yang, Tingting Gao, and Ming Wu. From reasoning depth to reasoning breadth: Evaluating multi-point associative reasoning in large language models. In *arXiv preprint arXiv:2608.10444*, 2026. https://arxiv.org/abs/2608.10444.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. Pride and prejudice: LLM amplifies self-bias in self-refinement. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. https://aclanthology.org/2024.acl-long.826/.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. CoMAS: Co-evolving multi-agent systems via interaction rewards. In *International Conference on Learning Representations (ICLR)*, 2026. https://openreview.net/forum?id=ihwAzktmWc.

Aijun Yang, Qianxue Guo, Ziyi Huang, Yuxuan Chen, Shiyou Qian, and Jian Cao. Self-correcting long-horizon search agents via tree-structured memory. In *arXiv preprint arXiv:2608.10676*, 2026a. https://arxiv.org/abs/2608.10676.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. In *arXiv preprint arXiv:2505.09388*, 2025a. https://arxiv.org/abs/2505.09388.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin Cui, and Shuicheng Yan. SuperCorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations (ICLR)*, 2025b. https://proceedings.iclr.cc/paper_files/paper/2025/hash/0967d7c8b171dd81b77c43067c02bebf-Abstract-Conference.html.

Yunhao Yang, Yuexin Bian, Yunjie Tian, Di Fu, Tianjin Huang, Yuanyuan Shi, Ziang Xiao, Nuno Vasconcelos, and Yijiang Li. Co-RL: Unsupervised reasoning emerges from diverse cohort in multi-agent RL. In *arXiv preprint arXiv:2608.17253*, 2026b. https://arxiv.org/abs/2608.17253.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. https://aclanthology.org/2023.emnlp-main.936/.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. https://aclanthology.org/2024.findings-acl.181/.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/hash/76ec4dc30e9faaf0e4b6093eaa377218-Abstract-Conference.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Yaolun Zhang, Yujie Zhao, Nan Wang, Yiran Wu, Jiayu Chang, Yizhao Chen, Qingyun Wu, Jishen Zhao, and Huazheng Wang. MetaAgent-X: Breaking the ceiling of automatic multi-agent systems via end-to-end reinforcement learning. In *arXiv preprint arXiv:2605.14212*, 2026a. https://arxiv.org/abs/2605.14212.

Zheyuan Zhang, Kaiwen Shi, Zhengqing Yuan, Zehong Wang, Tianyi Ma, Keerthiram Murugesan, Vincent Galassi, Chuxu Zhang, and Yanfang Ye. AgentRouter: A knowledge-graph-guided LLM router for collaborative multi-agent question answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026b. https://aclanthology.org/2026.acl-long.33/.

Zhixia Zhang, Zixuan Huang, Gongxun Li, Huaiyang Wang, Chengyi Yuan, Xin Xia, Deqing Wang, Fuzhen Zhuang, Shuai Ma, Ning Ding, Yaodong Yang, Jianxin Li, and Yikun Ban. Heterogeneous agent collaborative reinforcement learning. In *arXiv preprint arXiv:2603.02604*, 2026c. https://arxiv.org/abs/2603.02604.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning (ICML)*, 2026a. https://openreview.net/forum?id=Jpxfof0EaS.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-MAS: Multi-agent reinforcement learning for collaborative LLMs. In *International Conference on Learning Representations (ICLR)*, 2026b. https://openreview.net/forum?id=IdF6JqXWzx.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. https://aclanthology.org/2025.acl-long.50/.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *International Conference on Learning Representations (ICLR)*, 2026. https://iclr.cc/virtual/2026/poster/10010348.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: Large language models self-compose reasoning structures. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/hash/e41efb03e20ca3c231940a3c6917ef6f-Abstract-Conference.html.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *International Conference on Machine Learning (ICML)*, 2024. https://proceedings.mlr.press/v235/zhuge24a.html.

# APPENDIX

## A NOTATION

Table 2 summarizes the notation used in the collaborative reasoning and learning framework.

Table 2: Summary of the Main Notation.

| Notation | Description |
|---|---|
| $q$ | Task input. |
| $N$ | Number of models in the collaborative system. |
| $M_L, \theta_L$ | Original large model and its parameters. |
| $\mathcal{M}$ | Heterogeneous model set comprising $M_1, \ldots, M_N$. |
| $M_i, \theta_i, \lvert\theta_i\rvert$ | Model $M_i$, its parameters, and its parameter count. |
| $M^\star$ | Largest model in $\mathcal{M}$, used as the protocol-distillation teacher. |
| $\mathcal{M}_{\text{teacher}}$ | Homogeneous teacher system comprising $N$ copies of $M^\star$. |
| $\mathcal{J}$ | Evaluator model that assesses reasoning and interaction decisions. |
| $z_t = (i_t, x_t, y_t, a_t)$ | Recorded interaction step at index $t$. |
| $i_t$ | Index of the active model $M_{i_t}$ at step $t$. |
| $x_t$ | Step context containing the task input and preceding interaction history. |
| $y_t$ | Reasoning generated at step $t$. |
| $a_t$ | Interaction action: `continue`, `handoff`$(j)$, or `terminate`. |
| $o_t$ | Complete serialized response at step $t$, including reasoning, answer, and interaction fields. |
| $\tau = (z_1, \ldots, z_T)$ | Complete collaborative trajectory of $T$ interaction steps. |
| $P_{\mathcal{M}}(\tau \mid q)$ | Trajectory distribution induced by model pool $\mathcal{M}$ and the shared interaction protocol. |
| $\mathcal{Q}_{\text{distill}} = \{q^{(m)}\}_{m=1}^{K}$ | Collection of $K$ task inputs used for protocol distillation. |
| $\mathcal{Q}_{\text{step}}$ | Collection of task inputs used for step-level supervision. |
| $S_N, \pi^{(m)}$ | Set of permutations of $N$ model identities and the permutation sampled for trajectory $m$. |
| $\mathcal{D}_{\text{distill}}$ | Retained and identity-remapped steps used for protocol distillation. |
| $\mathcal{D}_i^{\text{pro}}$ | Protocol-distillation examples assigned to model $M_i$. |
| $s_t^{\text{rea}}, s_t^{\text{act}}$ | Evaluator scores for the reasoning and interaction action at step $t$. |
| $s_t^{\text{del}}$ | Step-level score obtained by combining the reasoning and action scores. |
| $m(q, \hat{y})$ | Task-specific evaluation metric for final prediction $\hat{y}$. |
| $r_{\text{task}}, r_t$ | Task-level reward and integrated reward for step $t$. |
| $\alpha$ | Coefficient balancing task-level and step-level feedback. |
| $\mathcal{D}_i$ | Collaborative-refinement training examples collected for model $M_i$. |
| $\mathcal{L}_i^{\text{pro}}, \mathcal{L}_i^{\text{col}}(\theta_i)$ | Protocol-distillation and collaborative-refinement objectives for model $M_i$. |

## B OVERVIEW OF THE TRAINING ALGORITHM

Algorithm 1 summarizes pyramid supervision, which consists of two stages: protocol distillation and step-level collaborative refinement. Let $\mathcal{M} = \{M_1, \ldots, M_N\}$ denote the heterogeneous model pool. The teacher model $M^\star$, selected as the largest model in this pool by parameter count, generates trajectories for protocol distillation; we use Qwen3-8B in our experiments. The original large model $M_L$ (Qwen3-14B) serves as the evaluator $\mathcal{J}$ during step-level supervision on all five datasets but is not deployed during collaborative inference.

Let $\mathcal{Q}_{\text{distill}}$ and $\mathcal{Q}_{\text{step}}$ denote the task inputs used for protocol distillation and step-level supervision, respectively. For each task $q$, we use the trajectory notation $\tau = (z_1, \ldots, z_T)$ with $z_t = (i_t, x_t, y_t, a_t)$ defined in Section 2.1, where $x_t$ contains $q$ and the preceding interaction history. The training target $o_t$ is the full serialized response at step $t$, including its reasoning, answer, and interaction fields. The distribution $P_{\mathcal{M}}(\tau \mid q)$ is induced by the model pool and its interaction protocol. During protocol distillation, a permutation $\pi$ is sampled uniformly from $S_N$, the set of permutations of the $N$ model identities, and used to remap identities consistently throughout

each teacher trajectory. The retained, remapped steps form the supervised dataset $\mathcal{D}_{\text{distill}}$. During step-level supervision, the evaluator produces reasoning and action scores $(s_t^{\text{rea}}, s_t^{\text{act}})$ for each step, which are averaged to obtain $s_t^{\text{del}}$. Task feedback and step-level scores determine the training weight $r_t$ according to Appendix D. For each model $M_i$, the dataset $\mathcal{D}_i$ contains $(x_t, o_t, r_t)$ from steps with $i_t = i$, so each model is optimized using only its own actions during collaborative refinement.

In our implementation, step-level collaborative refinement is performed offline. The heterogeneous models initialized by protocol distillation are held fixed while all collaboration trajectories are collected. Thus, $P_{\mathcal{M}}(\tau \mid q)$ in this stage denotes the trajectory distribution of the PD-initialized model pool at collection time. The evaluator scores each retained trajectory once, after which the model-specific datasets $\mathcal{D}_1, \ldots, \mathcal{D}_N$ and their rewards remain fixed. We then optimize each model for the task-specific number of epochs reported in Appendix J, without regenerating trajectories or recomputing evaluator scores.

**Algorithm 1** Pyramid Supervision for Heterogeneous Collaboration

**Require:** Model pool $\mathcal{M}$; largest model in the pool $M^\star$; task collections $\mathcal{Q}_{\text{distill}}, \mathcal{Q}_{\text{step}}$; evaluator $\mathcal{J}$; task-specific reward configuration; epoch counts $E_1, \ldots, E_N$

Construct homogeneous teacher system $\mathcal{M}_{\text{teacher}} = \{M_1^\star, \ldots, M_N^\star\}$
**Stage I: Protocol Distillation**
$\mathcal{D}_{\text{distill}} \leftarrow \emptyset$
**for** $q \in \mathcal{Q}_{\text{distill}}$ **do**
    Roll out teacher trajectory $\tau \sim P_{\mathcal{M}_{\text{teacher}}}(\tau \mid q)$
    Retain examples satisfying the task-specific data filters
    Sample identity permutation $\pi \sim \text{Uniform}(S_N)$
    Apply $\pi$ consistently to all model identities in $\tau$
    Add remapped steps to $\mathcal{D}_{\text{distill}}$
**end for**
**for** $i = 1, \ldots, N$ **do**
    Distill protocol behavior into $M_i$ using its corresponding steps in $\mathcal{D}_{\text{distill}}$
**end for**
**Stage II: Step-Level Collaborative Refinement**
Hold the PD-initialized models fixed during trajectory collection
$\mathcal{D}_1, \ldots, \mathcal{D}_N \leftarrow \emptyset$
**for** $q \in \mathcal{Q}_{\text{step}}$ **do**
    Roll out the task-specific number of trajectories for $q$ using the PD-initialized models
    **for** each collected trajectory $\tau$ **do**
        Compute task reward $r_{\text{task}}$
        Obtain $\{(s_t^{\text{rea}}, s_t^{\text{act}})\}_{t=1}^{T}$ once from $\mathcal{J}(q, \tau)$
        **for** $t = 1, \ldots, T$ **do**
            Compute $s_t^{\text{del}} = \frac{1}{2}(s_t^{\text{rea}} + s_t^{\text{act}})$
            Compute $r_t$ using task feedback and $s_t^{\text{del}}$ (Appendix D)
            Add $(x_t, o_t, r_t)$ to $\mathcal{D}_{i_t}$
        **end for**
    **end for**
**end for**
Fix $\mathcal{D}_1, \ldots, \mathcal{D}_N$ and their rewards
**for** $i = 1, \ldots, N$ **do**
    Fine-tune $M_i$ on $\mathcal{D}_i$ for $E_i$ epochs using $\mathcal{L}_i^{\text{col}}(\theta_i)$
**end for**
**return** Trained heterogeneous models $\mathcal{M} = \{M_1, \ldots, M_N\}$

# C Interaction and Inspection Protocol

This section details response requirements, inspector inputs, and task-specific scoring criteria.

## C.1 Response Schema and Field Requirements

Following the interaction protocol defined in Section 2.1, each model produces a serialized response with the following fields and action-specific requirements at every collaboration step.

**Field requirements.**

- `reasoning`: the current step's explanation or verification.
- `tentative_answer`: the current candidate—a MuSiQue answer, a GSM-Hard number, a MATH expression, a Conifer response, or code in its designated field.
- `action`: the serialized control decision corresponding to $a_t$.
- `handoff_target`: a permitted recipient ID within the model pool for the next handoff.
- `handoff_note`: the specific issue the recipient should check.
- `confirmed_answer`: the final answer for `terminate`, and null otherwise. For termination, the handoff fields remain null because no recipient is selected.

### C.2 Inspector Inputs

**Inspector model.** Qwen3-14B serves as the shared inspector for all five datasets.

**Input visibility.** Table 3 summarizes inspector inputs across tasks.

Table 3: Information Available to the Inspector.

| Information | MuSiQue | GSM-Hard | MATH | MultiPL-E | Conifer |
|---|---|---|---|---|---|
| Task context | Question and reference paragraphs | Problem statement | Problem statement | Source prefix and language | Instruction and parsed constraints |
| Reference information | Gold answer | Gold numeric answer | Gold mathematical answer | No reference code supplied | Reference response for audit |
| Candidate and final outputs | Tentative and final answers | Tentative and confirmed answers | Tentative and confirmed answers | Tentative and confirmed code completions | Response drafts and final response |
| Model identity | Model ID per step | Model ID per step | Model ID per step | Model ID per step | Model ID per step |
| Step sequence | All recorded steps and actions | All recorded steps and actions | All recorded steps and actions | All recorded steps and actions | All recorded steps and actions |
| Reasoning text | Up to 400 characters per step | Untruncated reasoning | Untruncated reasoning | Up to 1,200 characters per step | Up to 1,200 characters per step |

### C.3 Task-Specific Inspection

The inspector evaluates reasoning quality and the usefulness of each step's interaction decision. Table 4 summarizes the scoring criteria for all five datasets used in our experiments.

Table 4: Scoring Dimensions and Criteria by Dataset.

| Dataset | Dimensions | Score range | Criteria |
|---|---|---|---|
| MuSiQue | Reasoning; action | $[-1, 1]$, continuous | Evidence citations and multi-hop reasoning; supported correction and verification; penalties for regression and redundant interaction. |
| GSM-Hard | Process | $[-1, 1]$, step size $0.5$ | Correct, independently checkable arithmetic and appropriate actions; penalties for weak verification, misleading computation, and unsafe stopping. Incorrect candidates cannot receive positive scores. |

| Dataset | Dimensions | Score range | Criteria |
|---|---|---|---|
| MATH | Process | $[-1, 1]$, step size 0.5 | Correct mathematical reasoning and useful verification or correction; penalties for misleading work, regression, and unsafe stopping. Incorrect candidates cannot receive positive scores. |
| MultiPL-E | Reasoning; action | $[-1, 1]$, continuous | Implementation logic, edge cases, syntax, and completion boundaries; useful bug fixes and code audits; penalties for invalid completions and unsupported confirmation. |
| Conifer | Reasoning; action; content; final quality | Step: $[-1, 1]$; final: $[0, 1]$ | Constraint analysis, draft quality, and useful handoff or stopping decisions; penalties for repetition, premature stopping, and regression. Final quality evaluates the submitted response. |

# D Training and Inference Details

## D.1 Task Rewards and Training Weights

Let $m(q, \hat{y}) \in [0, 1]$ denote the task-specific evaluation metric for the final prediction $\hat{y}$, and let $s_t^{\text{del}} \in [-1, 1]$ denote the step-level score obtained by averaging the reasoning and action scores. The integrated training reward is

$$r_{\text{task}} = 2m(q, \hat{y}) - 1, \qquad r_t = \alpha r_{\text{task}} + (1 - \alpha) s_t^{\text{del}} \tag{15}$$

with $\alpha = 0.6$ for MuSiQue and $\alpha = 0.5$ for Conifer. The corresponding task-feedback coefficients in the base reward are $\alpha = 0.35$ for GSM-Hard, $\alpha = 0.35$ for MATH, and $\alpha = 0.3$ for MultiPL-E.

## D.2 Model-Wise Optimization

We optimize each model $M_i$ on its model-specific dataset $\mathcal{D}_i$ using the following reward-weighted collaborative-refinement objective, which has the score-function policy-gradient form (Williams, 1992):

$$\mathcal{L}_i^{\text{col}}(\theta_i) = -\mathbb{E}_{(x,o,r)\sim\mathcal{D}_i} \left[ r \log p_{\theta_i}(o \mid x) \right] \tag{16}$$

This objective increases the likelihood of responses assigned positive rewards and decreases the likelihood of those assigned negative rewards. In our implementation, it is optimized as offline reward-weighted fine-tuning: all trajectories are collected from the PD-initialized models and scored before parameter updates, and the datasets and rewards remain fixed across training epochs.

## D.3 Hardware and LoRA Configuration

Training uses $8\times$ NVIDIA A800-SXM4-80GB GPUs with a separate LoRA adapter (Hu et al., 2022) per model and frozen base parameters. The adapters use rank 64, scaling alpha 64, and dropout 0.05. LoRA targets the attention projections `q_proj`, `k_proj`, `v_proj`, and `o_proj`, and the MLP projections `gate_proj`, `up_proj`, and `down_proj` in every model-specific adapter.

# E Baseline Implementations and Comparison Protocol

This section details the implementation and evaluation settings of each baseline. Single-agent controls are Qwen3-14B (Yang et al., 2025a) and Qwen3-14B SE-RL. Homogeneous controls are an untrained MAS of three Qwen3-8B agents (Homo. MAS) and a collaboratively trained system of three Qwen3-4B agents (Homo. CL). Established multi-agent comparisons are MAD, AgentVerse, AFlow, and GPTSwarm (Du et al., 2024; Chen et al., 2024c; Zhang et al., 2025; Zhuge et al., 2024); multi-agent reinforcement-learning comparisons are MAGRPO, MAPoRL, and AT-GRPO (Liu et al., 2026b; Park et al., 2025; Zhao et al., 2026b). All use Qwen3 models with the assignments and task-specific configurations detailed below.

### E.1 Single-Agent and Homogeneous Controls

**Qwen3-14B.** We use the original Qwen3-14B checkpoint as a single-agent baseline for every task.

**Qwen3-14B SE-RL.** We initialize the policy and evaluator from Qwen3-14B, freeze the evaluator during training, and use it to score the policy's generated trajectories.

**Homogeneous MAS.** We use three original Qwen3-8B checkpoints as agents for multi-step collaborative reasoning under the shared interaction protocol used by our system.

**Homogeneous collaborative learning (Homo. CL).** We use the same training and evaluation configuration as CL, replacing its heterogeneous Qwen3-1.7B, Qwen3-4B, and Qwen3-8B model pool with three independently adapted Qwen3-4B agents (12B total).

### E.2 MAD

MAD organizes reasoning as a multi-round debate in which multiple agents first solve a task independently and then revise their answers after observing the other agents' responses from the preceding round (Du et al., 2024). After the final round, the agents' answers are aggregated by majority vote. Our implementation follows MAD's core workflow of independent initial reasoning, iterative peer-conditioned revision, and majority-vote aggregation at test time.

We deploy three debaters using Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, matching our heterogeneous model pool. With three total rounds, the nominal cost is nine agent calls per problem, excluding retries. This preserves MAD's three-agent debate and majority-vote structure while making minor adaptations for our Qwen3-based comparison setting.

### E.3 AgentVerse

AgentVerse recruits three task-specific specialists to generate responses and uses an evaluator to determine whether another iteration is needed (Chen et al., 2024c). The default maximum is three iterations. The evaluator scores the team's responses on a 0–10 scale, with early stopping at a score of at least 8. It receives the task input and agent responses, not a gold answer. The collective answer uses normalized voting with tie handling. This implementation follows AgentVerse's core workflow of expert recruitment, collaborative decision-making, and evaluation-guided iteration.

The three specialist agents use Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, while both the recruiter and evaluator use Qwen3-8B by default. With one recruitment call and four calls per iteration, the nominal range is 5–13 calls, excluding retries. This preserves AgentVerse's separation of specialist, recruitment, and evaluation calls while adapting the backbones and team size to the shared setting.

### E.4 AFlow

AFlow automates agentic workflow generation by using an optimizer model to propose executable workflows composed of reusable operators, evaluating candidate workflows on task examples, and iteratively refining them through search (Zhang et al., 2025). The selected workflow is then frozen and executed independently at evaluation time. Our implementation follows AFlow's core paradigm of workflow search, execution-based evaluation, and deployment of the selected workflow.

We use Qwen3-14B as the workflow optimizer. During workflow execution, solving calls rotate across Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, while candidate selection and answer extraction use Qwen3-8B by default. Workflow search uses 20 examples and 20 iterations. The optimizer is not invoked during evaluation, preserving AFlow's search-then-execute design.

### E.5 GPTSwarm

GPTSwarm represents a multi-agent system as an optimizable computational graph, where typed language-model nodes perform specialized operations and directed edges determine how intermediate outputs are exchanged (Zhuge et al., 2024). The original method optimizes graph connectivity,

while graph execution propagates information toward a designated output node. Our implementation retains GPTSwarm's core graph-structured execution and typed-node composition.

For a training-free comparison, we use GPTSwarm's graph-execution framework without graph optimization, executing a seven-node directed acyclic graph comprising two input/output nodes, three chain-of-thought nodes, one debate node, and one final aggregator. The default routing maps input/output nodes to the 1.7B model, chain-of-thought nodes to the 4B model, and debate/aggregation nodes to the 8B model. Using the original task input, the aggregator synthesizes one answer from its predecessors rather than taking a majority vote.

### E.6 MAGRPO

We adapt MAGRPO (Liu et al., 2026b) to three agents backed by Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, each with an independent LoRA adapter. Training samples groups of eight collaborative episodes and computes episode-relative advantages, with dynamic sampling used to maintain usable groups. The clipped objective uses clip ratio 0.2, advantage clipping at 4, and sequence-mean reduction. The KL coefficient is 0.02 except on MultiPL-E, where it is zero. All three policies use learning rate $5\times10^{-6}$. Evaluation runs two synchronous rounds.

### E.7 MAPoRL

We implement MAPoRL (Park et al., 2025) with Qwen3-1.7B, Qwen3-4B, and Qwen3-8B agents and independent LoRA adapters. Each collaborative episode contains three rounds with one proposal from each agent per round. Training uses reward discount 0.3, PPO clip ratio 0.2, KL coefficient 0.002, one PPO epoch, and eight minibatches, with learning rate $10^{-5}$.

### E.8 AT-GRPO

We use the agent-and-turn grouped optimization procedure of AT-GRPO (Zhao et al., 2026b). The three Qwen3-1.7B, Qwen3-4B, and Qwen3-8B policies have independent LoRA adapters and use spine-tree sampling with three training rounds and group size eight. The clipped objective uses ratio 0.2, KL coefficient 0.02, advantage clipping at 4, and sequence-mean reduction. The A1/A2/A3 learning rates are $2\times10^{-5}$, $1.5\times10^{-5}$, and $10^{-5}$, respectively. Evaluation uses four synchronous rounds on MuSiQue, GSM-Hard, MATH, and Conifer, and three rounds on MultiPL-E.

For each dataset, our method and all baselines are evaluated using the same task-specific evaluator.

### E.9 Soft F1 Metrics for GSM-Hard and MATH

The F1 columns reported for GSM-Hard and MATH measure soft numeric similarity.

For GSM-Hard, a prediction is considered exact when its absolute error is at most $10^{-6}$ or its relative error is below $10^{-6}$. Exact predictions receive a score of one. Otherwise, for prediction $\hat{y}$ and reference $y$, the per-example soft score is

$$s_{\mathrm{GSM}}(\hat{y}, y) = \begin{cases} \dfrac{\min(|\hat{y}|, |y|)}{\max(|\hat{y}|, |y|)}, & \hat{y}y > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Thus, zero-valued mismatches and predictions with the wrong sign receive zero. Non-exact per-example scores are rounded to four decimal places.

For MATH, let $\hat{y} \equiv y$ denote symbolic equivalence. The per-example soft score is

$$s_{\mathrm{MATH}}(\hat{y}, y) = \begin{cases} 1, & \hat{y} \equiv y, \\ \dfrac{\min(|\hat{y}|, |y|)}{\max(|\hat{y}|, |y|)}, & \hat{y}, y \in \mathbb{R} \text{ are finite and } \hat{y}y > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

For both benchmarks, the reported score is the arithmetic mean of the per-example scores over the complete evaluation set, presented as a percentage.

### E.10 MULTIPL-E EVALUATION METRICS

For MultiPL-E, we report two aggregate pass@1 metrics over the 16 dataset–language cells formed by the two component datasets, HumanEval and MBPP, and the eight programming languages. Let $N_{d,\ell}$ denote the number of evaluation problems in cell $(d, \ell)$, and let $u_{d,\ell,i} \in \{0, 1\}$ indicate whether the final program for problem $i$ passes its test suite. The pass@1 score within a cell is

$$s_{d,\ell} = \frac{1}{N_{d,\ell}} \sum_{i=1}^{N_{d,\ell}} u_{d,\ell,i}, \tag{19}$$

where $N_{d,\ell} = 49$ for each HumanEval cell and $N_{d,\ell} = 120$ for each MBPP cell. The macro-language pass@1 reported in the main table is the unweighted mean over the 16 dataset–language cells:

$$\text{Macro} = \frac{1}{16} \sum_{d} \sum_{\ell} s_{d,\ell}. \tag{20}$$

The weighted pass@1 weights each cell by its number of evaluation problems:

$$\text{Weighted} = \frac{\sum_d \sum_\ell N_{d,\ell} s_{d,\ell}}{\sum_d \sum_\ell N_{d,\ell}}. \tag{21}$$

Thus, the macro metric gives equal weight to every HumanEval–language and MBPP–language cell, whereas the weighted metric gives greater weight to the larger MBPP cells. The macro metric is computed over dataset–language cells, not over the eight languages after merging HumanEval and MBPP.

### E.11 CONIFER EVALUATION METRICS

For Conifer, we report two deterministic constraint-compliance metrics computed from each final response. The evaluator first applies a conservative parser to the instruction. For Coverage, it retains numbered requirements containing 2–12 words and at most 120 characters. For Explicit, it extracts mechanically verifiable constraints comprising response presence; requested output formats (bulleted, ordered, or general lists, tables, code, and paragraphs); word, sentence, and list-item limits; and quoted terms that must appear in the response. Neither uses inspectors or references.

Let $a_q$ be the final response for example $q$, $\mathcal{R}_q$ the set of parsed numbered requirements, and $\mathcal{T}(s)$ the set of unique lower-cased, non-stopword alphanumeric tokens in text $s$, allowing apostrophes, underscores, and hyphens within tokens. For an individual requirement $r$, its lexical coverage is

$$c(a_q, r) = \begin{cases} 1, & |\mathcal{T}(r)| = 0, \\ \dfrac{|\mathcal{T}(r) \cap \mathcal{T}(a_q)|}{|\mathcal{T}(r)|}, & \text{otherwise.} \end{cases} \tag{22}$$

The per-example Coverage score is

$$\text{Cov}_q = \frac{1}{|\mathcal{R}_q|} \sum_{r \in \mathcal{R}_q} c(a_q, r). \tag{23}$$

Coverage measures the average fraction of each parsed requirement's content tokens that appears in the final response; it is a lexical recall measure rather than a semantic correctness judgment.

For Explicit, let $\mathcal{B}_q$ be the set containing the response-presence check and every mechanically testable constraint extracted for example $q$, and let $b(a_q) \in \{0, 1\}$ indicate whether check $b$ passes. The per-example score is

$$\text{Expl}_q = \frac{1}{|\mathcal{B}_q|} \sum_{b \in \mathcal{B}_q} b(a_q). \tag{24}$$

All checks receive equal weight. Format and required-term checks use pattern or case-insensitive substring matching, while count constraints compare the measured numbers of words, sentences, or list items with the requested limits. Finally, for $N_{\text{eval}}$ evaluated responses, the reported table entries are $100 N_{\text{eval}}^{-1} \sum_q \text{Cov}_q$ and $100 N_{\text{eval}}^{-1} \sum_q \text{Expl}_q$. All evaluation items, including unsuccessful predictions, remain in the denominator for every reported aggregate.

# F Additional Analyses

## F.1 Efficiency and Behavioral Metrics

**Parameter-weighted output-token cost.** The computation cost of generating a given number of output tokens varies with model scale. To account for this variation across methods that use models of different sizes, we quantify computation cost using parameter-weighted output tokens. The shared normalization constant is $1.7 + 4 + 8 + 14 = 27.7$ (in billions), corresponding to the four nominal model sizes used across methods. For a given method and benchmark, the per-problem cost is

$$C_{\text{out}}(q) = \sum_{c \in \mathcal{C}_{\text{eval}}(q)} \frac{B_{m(c)}}{27.7} L_c^{\text{out}} \tag{25}$$

where $B_{m(c)}$ is the nominal parameter count, in billions, of the model producing generation $c$, and $L_c^{\text{out}}$ is the corresponding recorded output-token count. The set $\mathcal{C}_{\text{eval}}(q)$ contains all model generations included during evaluation for problem $q$, including those arising from retries, meta-model evaluations, aggregation, and separate bootstrap solvers. Reported means combine evaluation token totals, apply the model weights, and divide by $N_{\text{eval}}$ within each benchmark.

**Reasoning overlap.** The coordination analysis measures lexical overlap on adjacent steps in which the source model hands off to the model that acts next. For receiving model $i$, the overlap rate is

$$O_i(\delta) = \frac{\sum_{(u,v)\in\mathcal{H}_i} \mathbb{I}[\text{ROUGEL}_{F_1}(y_u, y_v) \geq \delta]}{|\mathcal{H}_i|} \tag{26}$$

where $\mathcal{H}_i$ contains eligible adjacent handoff pairs $(u, v)$, with source step $u$ handing off to $i$ and the immediately following step $v$ executed by $i$; pairs with missing reasoning or a mismatched or invalid handoff are excluded. We use $\delta = 0.8$. This is a threshold-exceedance rate.

**Correct-answer discovery and retention.** Let $\mathcal{Q}_{\text{eval}}$ be the evaluation set, $\mathcal{V}_q$ the method-specific set of extracted intermediate candidate answers for task $q$, and $\widehat{a}_q$ the final answer submitted for $q$. Let $g_q(v)$ indicate correctness under the benchmark's grader. Define

$$D_q = \mathbb{I}[\exists v \in \mathcal{V}_q : g_q(v) = 1], \qquad \text{Discovery} = \frac{\sum_q D_q}{|\mathcal{Q}_{\text{eval}}|}, \qquad \text{Retention} = \frac{\sum_q D_q g_q(\widehat{a}_q)}{\sum_q D_q} \tag{27}$$

Retention is undefined if no correct candidate is discovered. The candidate set $\mathcal{V}_q$ contains intermediate proposals, which are distinct from final aggregation outputs. A final answer can sometimes be correct even when none of the intermediate candidates is correct, so discovery multiplied by retention need not equal final accuracy. Retention measures preservation of available correct answers.

## F.2 Self-Evaluation Bias of the 14B Inspector

We compare Qwen3-14B inspection scores for heterogeneous CL and SE-RL trajectories. Table 5 reports mean scores and the share of paired questions favoring the self-generated trajectory. Paired comparisons use questions in the training scoring pool with valid scores for both trajectory sources, with ties retained in the denominator.

Across all five datasets, the inspector assigns higher mean scores to self-generated trajectories. The paired comparison shows the same asymmetry: self-generated trajectories score higher on 80.37% to 97.25% of questions, depending on the dataset in the paired evaluation.

Table 5: Qwen3-14B inspection scores for heterogeneous CL and self-generated SE-RL trajectories. The final column reports the proportion of paired questions on which the self-generated trajectory scores higher than its paired CL trajectory.

| Dataset | CL score | Self-generated score | Questions with higher self score |
|---|---|---|---|
| MuSiQue | 0.0194 | 0.8692 | 97.25% |
| GSM-Hard | 0.0921 | 0.7991 | 93.26% |
| MultiPL-E-8Lang | 0.6381 | 0.8283 | 90.05% |
| MATH | 0.1054 | 0.4911 | 80.37% |
| Conifer | 0.7076 | 0.8715 | 83.13% |

# G Case Studies

The following cases illustrate distinct collaboration behaviors in recorded trajectories. Unless explicitly marked as excerpts or summaries, displayed reasoning fields are reproduced in full, including errors; only typography and emphasis are added. Any omitted steps are identified explicitly. Agent identities and zero-based step indices follow the logs. Arrows denote handoffs, and `confirm_stop` denotes termination. Muted brown bold text highlights repeated or incorrect reasoning; black bold text highlights the checks discussed in the analysis.

## G.1 MuSiQue: Revisiting Evidence

We compare five zero-shot steps with three trained steps on the same postal-service question.

> **Question.** When did the person who first brought a postal service into Umayyad lands become caliph?
> **Reference:** 661. **ID:** `2hop__16581_16525`.

**Zero-shot: repeated reasoning across handoffs.** **Muted brown bold text** marks repeated spans. The expansion claim recurs in all five steps; steps 1–4 share the same final sentence, and steps 2 and 4 have identical reasoning. These highlights show textual repetition rather than factual correctness.

Table 6: MuSiQue, before collaborative learning: all five steps repeat the same incorrect candidate, 711.

| Step / action | Original reasoning (emphasis added) |
|---|---|
| 0 / $A_1$ $\rightarrow A_2$ | "The question asks when the person who first brought postal service into Umayyad lands became caliph. From paragraph [12], it states that the **Umayyad Caliphate started expanding rapidly in the peninsula beginning in 711**, and the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. The Emirate of Córdoba was established in 711, and Abd-ar-Rahman I was the first to bring postal service into Umayyad lands. Therefore, the person who first brought postal service into Umayyad lands became caliph in 711." |
| 1 / $A_2$ $\rightarrow A_3$ | "The question asks when the person who first brought postal service into Umayyad lands became caliph. Paragraph [12] states that the **Umayyad Caliphate started expanding rapidly in the peninsula beginning in 711**, and the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. **Abd-ar-Rahman I is noted as the first to bring postal service into Umayyad lands, and the Emirate of Córdoba was established in 711.**" |
| 2 / $A_3$ $\rightarrow A_1$ | "Paragraph [12] states that the **Umayyad Caliphate started expanding rapidly in the peninsula beginning in 711**, and the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. **Abd-ar-Rahman I is noted as the first to bring postal service into Umayyad lands, and the Emirate of Córdoba was established in 711.**" |

| Step / action | Original reasoning (emphasis added) |
|---|---|
| 3 / $A_1$ $\rightarrow A_2$ | "The question asks when the person who first brought postal service into Umayyad lands became caliph. From paragraph [12], it states that the **Umayyad Caliphate started expanding rapidly in the peninsula beginning in 711**, and the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. **Abd-ar-Rahman I is noted as the first to bring postal service into Umayyad lands, and the Emirate of Córdoba was established in 711.**" |
| 4 / $A_2$ `confirm_stop` | "Paragraph [12] states that the **Umayyad Caliphate started expanding rapidly in the peninsula beginning in 711**, and the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. **Abd-ar-Rahman I is noted as the first to bring postal service into Umayyad lands, and the Emirate of Córdoba was established in 711.**" |

**Outcome:** 5 steps; 4 handoffs; final answer 711; EM = 0, F1 = 0.

**Identical handoff note at steps 0–3:** "Please verify that the person who first brought postal service into Umayyad lands became caliph in 711 using paragraph 12."

**After training: revisiting the requested relation.** The trained trajectory initially proposes two incorrect candidates. At the final step, $A_3$ distinguishes *introducing* postal service from *extending* it. Bold highlights mark the question-specific check and conclusion while retaining surrounding errors.

Table 7: MuSiQue, after collaborative learning: the final model distinguishes the introduction of postal service from its later expansion, successfully correcting the earlier answer and arriving at the correct date.

| Step / action | Original reasoning and candidate (emphasis added) |
|---|---|
| 0 / $A_1$ $\rightarrow A_2$ | "Muhammad bin Qasim (paragraph 2) was the first to bring Islam to Punjab, and he was the Umayyad Caliphate's representative in the region. His appointment as a governor and his role in the Umayyad Caliphate's expansion (paragraph 2) implies he was a key figure in the Umayyad administration in the region. Thus, his role in the Umayyad Caliphate's establishment in the region (and not as a caliph in the broader sense) is tied to the Umayyad Caliphate's expansion."<br>*Candidate: Muhammad bin Qasim.* |
| 1 / $A_2$ $\rightarrow A_3$ | "Paragraph 2 states Muhammad bin Qasim introduced Islam to Punjab under Umayyad rule. Paragraph 8 confirms Umayyad Caliph Abd al-Malik extended postal service. Since Abd al-Malik's reign (723–743) is explicitly mentioned, his postal reforms likely occurred during his caliphate."<br>*Candidate: 723–743.* |
| 2 / $A_3$ `confirm_stop` | "Paragraph 8 states the Umayyad Caliph Abd al-Malik extended the postal service. Paragraph 20 mentions Muawiyah I's reign (661–80) and his role in establishing the Umayyad dynasty. However, paragraph 8 specifies Abd al-Malik's caliphate (723–743) as the period when the postal service was expanded. **The question asks for the person who first introduced postal service into Umayyad lands.** Paragraph 8 attributes the introduction of postal service to Muawiyah (paragraph 20's 'Muawiyah I' is likely the same as paragraph 8's 'Muawiyah' who was the first Umayyad caliph, but paragraph 8 says Abd al-Malik 'extended' it. However, paragraph 8 also mentions 'Muawiyah introduced postal service' (paragraph 8: 'Mu'awiyah introduced postal service, Abd al-Malik extended it…'). **This indicates Muawiyah was the first to introduce it, and his caliphate began in 661 (paragraph 20).**"<br>*Candidate: 661.* |

**Outcome:** 3 steps; 2 handoffs; final answer 661; EM = 1, F1 = 1.

**Evidence check.** The reference chain links [8], "Mu'awiyah introduced postal service, Abd al-Malik extended it throughout his empire," to [1], "Caliph Muawiyah (661–80)". The zero-shot agents instead cite [12], whose supplied excerpt describes expansion into Portugal beginning in 711, not the introduction of postal service. Trained $A_3$ cites [20], which connects Muawiyah I and the beginning of the dynasty to events in 661, rather than the reference paragraph [1], while still supporting the correct accession date.

**Repetition versus substantive re-examination.** Although the trained trajectory begins with an incorrect candidate, subsequent agents do not simply reaffirm the initial hypothesis. Instead, the focus shifts from an irrelevant entity to postal-service evidence before the final agent distinguishes the service's introduction from its later expansion and identifies the correct person–date relation. This progression shows how cross-agent re-examination can recover from an incorrect initial direction, unlike the repetitive confirmation in the zero-shot trajectory.

**Successful correction through cross-perspective collaboration.** The final agent brings a complementary perspective to the shared reasoning by examining whether the cited evidence establishes the specific relation requested by the question. Distinguishing who first introduced postal service from who later extended it allows the agent to identify Muawiyah and return the correct accession date, 661. Here, cross-model collaboration enables a later model to revisit the interpretation of shared evidence rather than simply continue the preceding line of reasoning. The successful final correction illustrates how complementary reasoning perspectives can be translated into a more accurate answer through explicit cross-model interaction in the final response.

### G.2 MuSiQue: Questioning Prior Reasoning

This second MuSiQue example concerns the interpretation of the question rather than the distinction between two historical relations. We show the zero-shot trajectory's first and final steps and all three trained steps; the omitted zero-shot steps are described below.

> **Question.** When is Celebrity Big Brother coming to the network which, along with ABC and the network which hosts Westinghouse Playhouse, is the other major broadcaster based in NY?
>
> **Reference:** February 7, 2018.
> **ID:** `3hop1__140554_2053_52946`.

**Zero-shot: retaining an incomplete task interpretation.** Although the question asks *when* the program will air, the agents repeatedly treat it as asking for a network. They also select NBC rather than excluding it alongside ABC. All four candidates are NBC. The complete reasoning texts of the omitted steps 1 and 2 are identical word for word to that of the final step shown in Table 8, including punctuation and spacing. Both omitted steps hand off again without resolving the date.

Table 8: MuSiQue, before collaborative learning: the first and final steps answer which network rather than when the program will air.

| Step / action | Original reasoning and candidate (emphasis added) |
|---|---|
| 0 / $A_1$ $\rightarrow A_2$ | "**The question asks for the network** that is the other major broadcaster based in New York, along with ABC and the network that hosts Westinghouse Playhouse. From the paragraphs, we know that ABC is one of the major broadcasters in New York. The network that hosts Westinghouse Playhouse is NBC. Therefore, the other major broadcaster based in NY is NBC." *Candidate: NBC.* |
| 3 / $A_1$ `confirm_stop` | "**The question asks for the network** that is the other major broadcaster based in New York, along with ABC and the network that hosts Westinghouse Playhouse. From paragraph 1, ABC is a major broadcaster in New York. From paragraph 16, Westinghouse Playhouse aired on NBC. Therefore, the other major broadcaster based in NY is NBC." *Candidate: NBC.* |

**Outcome:** 4 steps; 3 handoffs; final answer NBC; EM = 0, F1 = 0.

**After training: returning to the original question.** The trained candidates progress from CBS to the 2017–2018 season and finally to February 7, 2018. The first two omit the requested premiere date; bold text in Table 9 marks the final agent's return to the temporal question.

Table 9: MuSiQue, after collaborative learning: three steps with candidates and handoff notes, ending in a revision of the task interpretation.

| Step / action | Original reasoning and candidate (emphasis added) |
|---|---|
| 0 / $A_1$ $\rightarrow A_2$ | "Celebrity Big Brother is associated with CBS (paragraph 19). Westinghouse Playhouse is on NBC (paragraph 16). The question asks for the network that is not ABC and is not the same as CBS, but is a major network in New York. The correct answer is CBS, which is not explicitly excluded from the 'major broadcasters in New York' list, but the question implies a network that is not ABC and not the same as NBC (since Westinghouse Playhouse is on NBC). Thus, the network in question is CBS." *Candidate: CBS.* **Handoff note:** "Verify that CBS is the correct network for the season of Celebrity Big Brother, and that it is not the same as ABC or NBC." |
| 1 / $A_2$ $\rightarrow A_3$ | "Paragraph 19 states Celebrity Big Brother aired on CBS during 2017-2018. Paragraph 16 confirms Westinghouse Playhouse (NBC) is a New York network. Paragraph 4 notes ABC and CBS are both NYC-based. Thus, CBS is the network alongside ABC and NBC." *Candidate: 2017-2018.* **Handoff note:** "Check if CBS's 2017-2018 schedule aligns with the NYC-based networks' overlap." |
| 2 / $A_3$ `confirm_stop` | "Paragraph 17 mentions Celebrity Big Brother (American TV series) will air on CBS during the 2017-18 network season. Paragraph 4 states ABC and CBS are both headquartered in NYC. Paragraph 16 notes Westinghouse Playhouse aired on NBC. The question asks for the network (other than ABC and NBC) based in NY. However, the phrasing implies the network that hosts Celebrity Big Brother (CBS) is being asked, but the user might have intended to ask about a different network. Wait, the question says 'along with ABC and the network which hosts Westinghouse Playhouse (NBC), is the other major broadcaster based in NY'. The three major NYC networks are ABC, CBS, and NBC (paragraph 4). The question seems to ask when Celebrity Big Brother (CBS) is coming to the network that is not ABC or NBC, but this is impossible. However, the correct interpretation is that the question is asking for the network (CBS) that, along with ABC and NBC (hosting Westinghouse Playhouse), is a major NYC-based network. **But the actual question is about when Celebrity Big Brother (CBS) is coming to the network (CBS) that is part of the NYC-based trio.** This seems redundant. Alternatively, the question might have a typo, but based on the given data, the answer refers to the CBS season premiere date. Paragraph 19 states the 2017-18 season premiered on February 7, 2018." *Candidate: February 7, 2018.* |

**Outcome:** 3 steps; 2 handoffs; final answer February 7, 2018; EM = 1, F1 = 1.

**Evidence check.** Paragraph [16] identifies NBC as the network on which *Westinghouse Playhouse* aired. Paragraph [4] lists ABC, CBS, and NBC as the three major networks headquartered in New York, identifying CBS as the remaining network. Paragraph [17] states that *Celebrity Big Brother* will air on CBS and that "CBS announced that the series is set to premiere on February 7, 2018 and conclude on February 25, 2018." The final model output attributes the date to [19]; this citation is retained as logged, while [17] supplies the date in the provided evidence excerpts used for comparison.

**Questioning and revising prior reasoning.** The distinction here is between identifying an intermediate entity and answering the original question. Whereas the zero-shot agents remain focused on network identification, the final trained agent explicitly recognizes that the question asks when the program will air. It retains CBS as the relevant network but replaces the preceding season-level candidate with the premiere date. This illustrates revision of the task interpretation, not merely substitution of one network name for another. In this case, a later model brings a different perspective to the shared reasoning, correcting the earlier focus on network identification rather than carrying that incomplete interpretation into the reported final answer in the evaluation results.

### G.3 GSM-HARD: VERIFYING A CORRECT ANSWER

This example shows correct-answer retention through an alternative arithmetic check. We show both recorded steps and the handoff note, with mathematical expressions typeset.

> **Question.** If there are 657881 eggs in a basket, and there are twice as many eggs in a second basket, how many eggs are in both baskets put together?
>
> **Reference:** 1973643. **ID:** `gsm_00016`.

Table 10: GSM-Hard: $A_2$ verifies $A_3$'s answer via multiplication and place-value decomposition.

| Step / action | Original reasoning and candidate (emphasis added) |
|---|---|
| 0 / $A_3$ $\rightarrow A_2$ | "First, calculate the eggs in the second basket: $657881 * 2 = 1315762$. Then add both baskets: $657881 + 1315762 = 1973643$." *Candidate: 1973643.* **Handoff note:** "Verify the addition step: $657881 + 1315762$" |
| 1 / $A_2$ `confirm_stop` | "**Total eggs = $3 \times 657881$.** Breakdown: $600000 \times 3 = 1800000$, $50000 \times 3 = 150000$, $7000 \times 3 = 21000$, $800 \times 3 = 2400$, $80 \times 3 = 240$, $1 \times 3 = 3$. Sum: $1800000 + 150000 = 1950000$; $+21000 = 1971000$; $+2400 = 1973400$; $+240 = 1973640$; $+3 = 1973643$." *Candidate: 1973643.* |

**Outcome:** 2 steps; 1 handoff; final answer 1973643; EM = 1.

**Preserving a correct answer through verification.** The receiving agent builds on the preceding reasoning by expressing the same total as three times the first basket's contents and verifying the multiplication through a place-value decomposition. This alternative calculation reaches the same answer while providing an additional computational basis for accepting it. The case illustrates that cross-model collaboration is valuable not only for revising answers, but also for substantiating correct ones: a later model can contribute a complementary reasoning perspective that strengthens the basis for retaining an existing solution. Diverse perspectives thus let agents cross-check the same problem through complementary calculations, improving final-answer reliability.

### G.4 GSM-HARD: RECOVERING FROM A REGRESSION

This second GSM-Hard example shows a correct answer becoming incorrect before being restored at termination. All three steps and the two handoff requests are shown. The highlighted spans contrast the intermediate quantity assumption with the final conversion from legs to pairs.

> **Question.** Mario needs to buy snowshoes for his 8374485 sled dogs. Assuming his dogs each has four legs and each pair of snowshoes costs $12.00, how much will it cost him to buy snowshoes for all of his dogs?
>
> **Reference:** 200987640. **ID:** `gsm_00076`.

Table 11: GSM-Hard: $A_1$ corrects $A_2$'s one-pair-per-dog regression, restoring the initial answer.

| Step / action | Original reasoning and answer (emphasis added) |
|---|---|
| 0 / $A_3$ (8B) $\rightarrow A_2$ | "First, calculate the total number of snowshoe pairs needed: 8,374,485 dogs * 2 pairs per dog = 16,748,970 pairs. Then multiply by the cost per pair: 16,748,970 * $12.00 = $200,987,640." *Candidate: 200987640.* **Handoff note:** "Verify the multiplication step: 16,748,970 * 12" |
| 1 / $A_2$ (4B) $\rightarrow A_1$ | "First, calculate the total number of snowshoe pairs needed: 8,374,485 dogs * **1 pair per dog** = 8,374,485 pairs. Then multiply by the cost per pair: 8,374,485 * $12.00 = $100,493,820." *Candidate: 100493820.* **Handoff note:** "Verify the multiplication step: 8,374,485 * 12" |

| Step / action | Original reasoning and answer (emphasis added) |
|---|---|
| 2 / $A_1$ (1.7B) `confirm_stop` | "First, determine the number of snowshoe pairs per dog: **4 legs / 2 legs per pair = 2 pairs.** Total pairs = 8,374,485 dogs * 2 pairs/dog = 16,748,970 pairs. Total cost = 16,748,970 pairs * \$12.00/pair = \$200,987,640." *Confirmed answer: 200987640.* |

**Outcome:** 3 steps; 2 handoffs; final answer 200987640; EM = 1.

**Note:** "Independently reverified the original problem's arithmetic and confirmed the final numeric answer."

**Recovering from an intermediate reasoning regression.** The initial agent correctly assigns two pairs of snowshoes to each dog. The intermediate agent changes this quantity to one pair, halving the total cost. Its multiplication is correct for that quantity; the error lies in the number of pairs per dog, not in the multiplication itself. The final agent returns to the problem's four-leg condition and explicitly derives two pairs per dog before recomputing the total. This check repairs the incorrect premise introduced in the preceding step and restores the original correct answer. Unlike the preceding GSM-Hard case, where correctness is retained throughout, this trajectory demonstrates recovery after an intervening regression. Here, the repair is made by the smallest agent, $A_1$ (1.7B). Although introducing another model's perspective may yield a reasoning step that does not fully account for the problem's conditions, a subsequent model can revisit that step from a different perspective and bring the reasoning back into alignment with those conditions. In this case, cross-model collaboration enables the pairs-per-dog assumption to be reassessed and corrected before the final answer, illustrating the value of successive cross-perspective checks.

### G.5 MultiPL-E: Completing a Missing Validation Check

This C++ example concerns a requirement acknowledged in the initial reasoning but omitted from the implementation. Here, $A_1$, $A_2$, and $A_3$ correspond to Qwen3-8B, Qwen3-4B, and Qwen3-1.7B, respectively. All three steps appear below: steps 0–1 use supplied reasoning excerpts, and the final step is summarized. Ellipses mark omitted text. Only the first handoff note is excerpted; the second note and final-step reasoning are not reproduced in the table.

> **Task.** Given the lengths of the three sides of a triangle, return its area rounded to two decimal places if the triangle is valid; otherwise return $-1$. A valid triangle requires the sum of any two sides to be greater than the third.
>
> **Signature:** `float triangle_area(long a, long b, long c)`
>
> **ID:** `humaneval/cpp/HumanEval_71_triangle_area`.
>
> **Evaluation:** Nine assertions; invalid triangles $(1, 2, 10)$, $(1, 2, 3)$, $(2, 6, 3)$, and $(2, 2, 10)$ must return $-1$.

Table 12: MultiPL-E: $A_2$ adds the validation check missing from $A_1$'s implementation, and $A_3$ retains the repair. Reasoning and handoff text is excerpted; the final step is summarized.

| Step / action | Reasoning excerpts and candidate code (emphasis added) |
|---|---|
| 0 / $A_1$ (8B) $\rightarrow A_2$ | "The function needs to check if the three sides form a valid triangle. If they do, it should calculate the area using Heron's formula… Otherwise, return -1." **Candidate body:** `float s = (a + b + c) / 2.0f;` `float area = sqrt(s * (s - a) * (s - b) * (s - c));` `return round(area * 100) / 100;` **Handoff excerpt:** "Verify the calculation… **Check if the validation of the triangle sides is properly handled before the calculation.**" |

| Step / action | Reasoning excerpts and candidate code (emphasis added) |
|---|---|
| 1 / $A_2$ (4B) $\rightarrow A_3$ | "The candidate code correctly implements Heron's formula... However, **the validation of the triangle sides is missing.** The function should first check if the sides form a valid triangle before proceeding." **Candidate body:** `if (a + b > c && a + c > b && b + c > a) { float s = (a + b + c) / 2.0f; float area = sqrt(s * (s - a) * (s - b) * (s - c)); return round(area * 100) / 100; } return -1L;` |
| 2 / $A_3$ (1.7B) `confirm_stop` | **Summary (not a quotation):** The final candidate retains the preceding implementation, compressed onto one line, including the triangle-inequality check and the fallback return value. The agent terminates the trajectory. |

**Outcome:** 3 steps; 2 handoffs; the final candidate passes all nine supplied assertions in post-hoc execution.

**Post-hoc execution check.** Each step's candidate was separately compiled and run against the supplied tests after the trajectory was collected. All three candidates compiled successfully, but only the latter two passed all nine assertions reported in Table 13.

Table 13: Post-hoc results for the three C++ candidates on the supplied nine-assertion test suite.

| Step | Agent | Compilation | Test result |
|---|---|---|---|
| 0 | $A_1$ (8B) | Success | Fails the $-1$ assertion for $(1, 2, 10)$ and aborts. |
| 1 | $A_2$ (4B) | Success | Passes all nine assertions. |
| 2 | $A_3$ (1.7B) | Success | Passes all nine assertions. |

**Repair mechanism.** For $(a, b, c) = (1, 2, 10)$, the initial implementation computes $s = 6.5$ and $s - c = -3.5$, making the argument to `sqrt` negative. The resulting NaN does not satisfy the required return value of $-1$. The added guard rejects this input before evaluating the area. It also rejects the degenerate triangle $(1, 2, 3)$, for which the initial implementation returns zero rather than $-1$.

**Completing an implementation rather than replacing a solution.** The initial agent notes but omits triangle validation. The recipient checks the candidate against the specification, identifies the missing condition, and adds a validation branch while preserving the area calculation. This complementary review fills a gap in the initial code, showing that cross-model collaboration can check and complete an existing solution. The final model retains the repair, and the failure-to-pass transition provides executable evidence that it fixes the defect.

**A targeted handoff and a corresponding repair.** The first handoff explicitly requests a check of triangle-side validation, and the next agent addresses precisely that gap. This alignment illustrates how a targeted request can accompany a substantive cross-model contribution. Together, the trajectory and execution results illustrate how complementary reasoning perspectives across models can be translated into a concrete improvement in the final implementation.

## H Dataset Splits and Evaluation Sets

This section documents the dataset partitions used for training, validation, and final evaluation. For each benchmark, we report the source split, the numbers of retained IDs, and the random seed or deterministic rule used to construct any new partition.

We use official benchmark partitions when available. MuSiQue provides official train, development, and test sets, while MATH provides official training and test sets. GSM-Hard is released as a single split, MultiPL-E is released as evaluation configurations without a unified training split, and Conifer is released as a single training split; we therefore construct task-specific partitions for these three

datasets. For MultiPL-E, the partition is defined at the underlying-problem level before expanding each problem into eight language-specific instances. Exact split files and IDs are included in the released experiment artifacts.

Table 14: Training and evaluation partitions used in the experiments.

| Dataset | Source split | Training examples | Evaluation examples |
|---|---|---|---|
| MuSiQue | Official train/dev | 19,938 | 2,417 |
| GSM-Hard | Official dataset divided into train/eval by us | 1,187 | 132 |
| MATH | Official train/test; evaluation uses test shard 04 | 5,411 | 500 |
| MultiPL-E | Official benchmark divided 70/30 by us | 3,128 | 1,352 |
| Conifer | Official training set divided into train/dev/test by us | 10,813 | 1,402 |

MuSiQue uses the official `problem_id`; its official training and development sets have zero ID overlap. Because GSM-Hard provides only one official split, we divide its 1,319 examples into 1,187 training and 132 evaluation examples. The stored examples have no native ID, so we identify them by a SHA-256 fingerprint of the input text; the two partitions have zero fingerprint overlap. MultiPL-E is partitioned by the underlying HumanEval or MBPP problem before language expansion using master seed 7658190907657085414, yielding zero underlying-problem overlap. Because Conifer provides only a training split, we construct training, validation, and evaluation sets using a deterministic SHA-256 hash of normalized prompt groups, with zero group overlap across the three sets.

The official MATH dataset contains 7,500 training problems and 5,000 test problems. After rollout filtering, holdout construction, and trajectory-class balancing, 5,411 training problem IDs are retained from the official training split. Our additional subdivisions preserve the official train–test boundary. To reduce evaluation cost, we stratify the official test problems by subject and use a seeded hash to assign the problems within each subject deterministically to ten 500-problem shards. We use seed 42 and shard 04 for all reported evaluations.

# I Training Data Statistics

This section reports the number of training examples used for each model slice during the two stages of pyramid supervision: protocol distillation and step-level collaborative refinement. We also summarize the trajectory generation and filtering statistics associated with these training examples. The tables report model-specific training records, whereas the accompanying trajectory counts refer to complete collaboration trajectories.

## I.1 Protocol-Distillation Data

During protocol distillation, the homogeneous teacher system generates collaboration trajectories from the distillation inputs. Table 15 reports the number of training examples loaded for each model slice. Additional trajectory statistics are given below.

Table 15: Training-data statistics for protocol distillation.

| Dataset | Qwen3-1.7B | Qwen3-4B | Qwen3-8B |
|---|---|---|---|
| MuSiQue | 3,813 | 3,813 | 3,813 |
| GSM-Hard | 2,200 | 2,200 | 2,200 |
| MATH | 2,200 | 2,200 | 2,200 |
| MultiPL-E | 3,953 | 3,857 | 627 |
| Conifer | 13,196 | 14,706 | 14,607 |

The table reports the records loaded by the corresponding SFT trainer. For MuSiQue, each trainer loaded 3,813 examples. GSM-Hard retains 3,166 of 9,758 trajectories, while MATH retains 3,137 trajectories. MultiPL-E uses 8,437 training examples. Conifer retains 42,509 response records from 14,162 trajectories.

### I.2 Step-Level Collaborative-Refinement Data

During step-level supervision, collaboration trajectories generated by the PD-initialized heterogeneous models are evaluated once and converted into fixed model-specific training datasets. Table 16 reports the number of offline reward-weighted training records used for each model slice. Additional rollout statistics are given below.

Table 16: Training-data statistics for step-level collaborative refinement.

| Dataset | Qwen3-1.7B | Qwen3-4B | Qwen3-8B |
|---|---|---|---|
| MuSiQue | 2,416 | 1,727 | 1,411 |
| GSM-Hard | 3,714 | 3,516 | 2,765 |
| MATH | 32,746 | 29,261 | 30,200 |
| MultiPL-E | PD only | 1,908 | 1,860 |
| Conifer | 46,746 | 47,063 | 44,913 |

The table reports the fixed offline reward-weighted training records constructed before optimization. Training epochs repeatedly traverse these records; no new trajectories are generated and no evaluator scores are recomputed between epochs. GSM-Hard uses eight trajectories per input and retains 7,344 trajectories after removing all-failed problem groups and applying trajectory-class balancing. For MATH, we sample eight trajectories for each of 7,500 source problems and retain 5,411 problem IDs after rollout filtering, holdout construction, and trajectory-class balancing. For MultiPL-E, the 1.7B slice uses its protocol-distilled adapter without an additional step-level collaborative-refinement update; the 4B and 8B slices use 1,908 and 1,860 paired training records, respectively. MuSiQue retains 1,748 of 1,996 generated trajectories. Conifer generates 43,252 trajectories, of which 43,221 contain at least one interaction step and contribute training records.

## J Training Hyperparameters and Optimization Details

Shared hardware and LoRA configurations are provided in Appendix D. This section reports the task- and stage-specific optimization hyperparameters used in the final runs.

In Table 17, PD and SLS denote protocol distillation and step-level supervision, respectively. The latter performs offline collaborative policy refinement using fixed integrated step rewards. The reported SLS epochs are repeated passes over the same fixed model-specific datasets, without trajectory regeneration between epochs. Device batch is the per-process batch size, while gradient accumulation specifies the number of device batches accumulated before each parameter update. Rows marked "All" use the same configuration for all three model sizes.

Table 17: Optimization hyperparameters for the two training stages. Shared configurations are marked as "All," while MultiPL-E uses model-specific rows because its settings differ across model sizes.

| Dataset | Stage | Model | Epochs | Learning rate | Device batch | Grad. accum. |
|---|---|---|---|---|---|---|
| MuSiQue | PD | All | 5 | $2\times10^{-5}$ | 4 | 4 |
| MuSiQue | SLS | All | 2 | $1\times10^{-5}$ | 1 | 2 |
| GSM-Hard | PD | All | 3 | $1\times10^{-5}$ | 1 | 4 |
| GSM-Hard | SLS | All | 1 | $3\times10^{-6}$ | 1 | 2 |
| MATH | PD | All | 3 | $1\times10^{-5}$ | 1 | 4 |
| MATH | SLS | All | 1 | $1\times10^{-6}$ | 1 | 2 |
| MultiPL-E | PD | 1.7B | 5 | $2\times10^{-5}$ | 2 | 4 |
| MultiPL-E | PD | 4B | 2 | $8\times10^{-6}$ | 2 | 4 |

| Dataset | Stage | Model | Epochs | Learning rate | Device batch | Grad. accum. |
|---|---|---|---|---|---|---|
| MultiPL-E | PD | 8B | 5 | $2\times10^{-5}$ | 1 | 8 |
| MultiPL-E | SLS | 4B | 1 | $5\times10^{-8}$ | 1 | 2 |
| MultiPL-E | SLS | 8B | 1 | $5\times10^{-8}$ | 1 | 2 |
| Conifer | PD | All | 3 | $2\times10^{-5}$ | 2 | 4 |
| Conifer | SLS | All | 3 | $5\times10^{-6}$ | 1 | 16 |