

How to Train a Leader: Hierarchical Reasoning in Multi-Agent LLMs

Andrew Estornell^{*,1}, Jean-Francois Ton^{*,1}, Muhammad Faaiz Taufiq^{*,1}, Hang Li¹

¹ByteDance Seed

*All authors contributed equally; ordering chosen at random.

Abstract

Large Language Models (LLMs) have achieved strong performance on a wide range of complex reasoning tasks, yet further gains are often possible by leveraging the complementary strengths of multiple models. While multi-agent frameworks can improve solution quality by leveraging multiple LLMs, existing methods are often computationally expensive, both at training and inference time. In this work, we introduce a hierarchical multi-agent framework that addresses these challenges by training only a single leader LLM to coordinate a team of untrained peer agents. To this end, we propose **M**ulti-agent guided **L**eader **P**olicy **O**ptimization (MLPO), a novel approach which trains the leader to evaluate and synthesize agent responses without auxiliary value networks or explicit agent feedback. Leaders trained with MLPO exhibit improved performance not only when interacting with the agent team at inference time, but also enjoy improved performance when deployed in single-agent settings without the team. Empirical results on Big-Bench Hard (BBH), MATH, and MMLU demonstrate that our framework achieves substantial performance improvements over both single-agent and multi-agent baselines. Our results highlight the effectiveness and efficiency of training a single, flexible leader for collaborative reasoning in multi-agent LLM systems.

Correspondence: {andrew.estornell, jeanfrancois, faaiz.taufiq, lihang.lh}@bytedance.com

1 Introduction

Large Language Models (LLMs) have rapidly become foundational in natural language processing, demonstrating remarkable versatility across tasks ranging from translation to complex question answering [2, 3]. While these models have shown impressive performance in a wide range of tasks, they still struggle with both factual accuracy and complex reasoning [24, 44].

To address some of these limitations, Multi-Agent approaches in particular have shown promise: by having two or more LLMs collaboratively solve a problem, the group of models may collectively produce better answers than a single model alone [4–7, 10–12, 14–18, 28, 29, 31, 35, 38, 45, 58, 61]. However, current multi-LLM frameworks mostly rely on off-the-shelf models that have not been explicitly trained to collaborate, treating effective collaboration as an emergent property of large models [6, 10, 29, 64]. These works assume that general-purpose LLMs are inherently capable of debating, verifying, or correcting each other, and focus on designing (or training) mechanisms to better elicit this behaviour from off-the-shelf models.

In contrast, some recent work has begun to explore training schemes that explicitly tune models for multi-agent collaboration [13, 32, 45, 46, 48, 57, 71]. For example, ACC-Collab [13] jointly trains a two-model team (one

actor and one critic) to solve tasks through iterative dialogue. Other works have extended this paradigm to teams of two or three similarly specialised models [32, 45, 57, 71], demonstrating that training specialised teams can yield performance improvements. Some approaches further scale to larger groups of trained agents [46, 48]. However, a key drawback of each of these methods is that they require jointly optimising multiple LLMs, an approach that is computationally expensive and difficult to scale. This raises a fundamental question:

Can we enhance a multi-agent team’s collaborative reasoning by training **only a single** model, rather than training multiple models?

In this paper, we answer this question by proposing a novel hierarchical multi-agent architecture for collaborative reasoning, **M**ulti-agent guided **L**eader **P**olicy **O**ptimization (MLPO). Our approach draws inspiration from hierarchical structures in multi-agent systems, where a designated leader coordinates a team of subordinate agents [23, 28, 34, 39, 52], and from recent advancements in reinforcement-learning-based reasoning techniques [72]. Specifically, we introduce a framework in which a single leader LLM is trained to solve tasks with the assistance of a team of untrained (off-the-shelf) LLM agents that provide candidate solutions. Only the leader is trained, while the supporting team members remain fixed. During inference, the leader queries the agent team, aggregates their outputs, and synthesises a final answer.

MLPO can be regarded as multi-agent guided training. During this process, the leader model both strengthens its collaboration with the agent team and broadens its exploration of the solution space by synthesising the diverse candidate answers supplied by the agent team into its own answer. In RL-based training, providing the leader with a richer pool of candidate solutions broadens its search space, enabling the discovery of more effective reasoning strategies and yielding higher-quality final outputs.

Since our approach involves training only a single model, our multi-agent guided training scheme offers clear advantages, including improved efficiency during training and greater flexibility at inference time. Unlike previous multi-agent methods, which require all agents to participate at test time, our trained leader can effectively function independently. We empirically observe that our training pipeline not only enhances the leader’s performance when interacting with the agent team at inference time, but also enhances its performance as an individual model operating without the team. In other words, our training approach enhances both collaborative and zero-shot performance. Consequently, when inference-time cost is a constraint, the leader alone can be deployed and still achieves competitive results, although full collaboration generally provides the best performance

We evaluate our hierarchical leader team architecture using 7–9B-parameter models on three challenging reasoning suites: BIG-BENCH HARD (BBH) [54], MATH [22], and MMLU [21]. Against two classes of baselines (i) strong prompt-only methods such as Chain-of-Thought and multi-agent debate with off-the-shelf LLMs [10, 60], and (ii) trained variants that employ the same agents but use different training techniques as well as components [13, 30, 43, 72].

The contributions of this paper are as follows:

- **Hierarchical framework:** We propose a hierarchical multi-agent architecture where a single leader LLM is explicitly trained to coordinate a team of untrained peer LLMs for collaborative reasoning.
- **Multi-agent guided GRPO objective:** We introduce a novel GRPO-based approach, which we coin MLPO , enabling the leader to implicitly evaluate and synthesize agent responses without the need for auxiliary value networks or explicit agent feedback signals.
- **Strong empirical results:** Extensive experiments demonstrate that our hierarchical multi-agent approach significantly outperforms existing baselines, including both trained and untrained single-agent and multi-agent methods.
- **Systematic ablation study:** We perform thorough ablations on team composition, aggregation methods, and sampling strategies, providing insights into their individual contributions and guiding further improvements to our framework.

2 Related Work

Our work is mostly closely related to the areas of multi-LLM collaboration and iterative improvement.

Multi-LLM Collaboration Similar our proposed method, several other works have investigated the use of leveraging multiple LLMs to improve answer quality. These works can be loosely grouped into two categories: collaborative methods, which engage multiple LLMs in collaboration, often in the form of iterative discussion [5, 10, 12, 14, 15, 17, 18, 29, 33, 35, 36, 38, 65, 68], and aggregation methods which attempt to bootstrap multiple LLMs (responses) together without direct collaboration [9, 25, 27, 64]. There are some methods that fall into both categories, such as mixture of agents (MoA) [7, 31, 58].

Several unique paradigms have been proposed within the space of collaborative multi-LLM approaches. Notably, [10] proposes a system of multi-agent debate in which a team of LLMs directly communicates with one another over rounds of discussion. Other works have extended this basic framework with a focus on designing mechanisms to enhance the collaborative ability of off-the-shelf-models; [29] introduces a judge/moderator LLM, [6, 70] propose alternating the order of model interaction, [33] allows LLMs to score each other’s responses in between interaction rounds, many other such mechanisms have been proposed such as [15, 17, 18, 33, 36, 39, 65, 68]. While effective in certain settings, prior approaches do not directly train models to collaborate. Instead, they aim to improve the collaboration of off-the-shelf models through either structured interaction schemes (e.g., two LLMs debating with a third acting as a judge) or learned selection mechanisms (e.g., a classifier that selects a single response from a pool of candidate outputs). Diverging from this work, our proposal involves training a reasoning-based leader LLM which critically analyzes the team responses, thereby explicitly guiding the team towards the right direction when needed.

More recently, several recent works have proposed training schemes to directly enhance the collaborative ability of LLMs. In particular, ACC-Collab [13] designs a scheme for training a two model team consisting of an actor and critic agent. Building upon ACC-Collab, other works have also designed two-agent [32, 41, 57, 71] (and three agent [45]) training paradigms. Extending this further, other works such as [37, 46, 48] train multiple models to collaborate to directly collaborate. However, these training schemes can be exceedingly expensive as they require training multiple models simultaneously or alternatively. Our framework aims to make multi-agent training more efficient by training only a *single leader agent* without the requirement of training the other models in the team.

Self-Improvement Our work is also related to self-improvement. Techniques in this area aim to improve LLM efficacy by iteratively refining given answers. This refinement can come in many forms, from prompting models to improve their previous answers [42], to iterative exploration or expansion of their reasoning [1, 60, 67], and even to directly training advanced single-shot reasoning capabilities into the model itself [51, 72]. Of particular relevance to our work is that of [30] which trains a model to directly update and refine its own responses. This can be thought of as an explicit form of the type of *backtracking* behaviour found in modern reasoning models [26, 47, 63, 72]. In our method, the leader agent learns to perform a type of macro-update to the agent team responses, while using those responses to synthesize its own refined answer and feedback.

3 Methodology

3.1 Hierarchical Multi-Agent Inference Pipeline

Multi-Agent Team Setup Our methodology utilizes a hierarchical multi-agent architecture comprising a single leader model L and a team of K off-the-shelf agent models, denoted as $\{a_1, a_2, \dots, a_K\}$. Although we set $K = 3$ in our experiments, this framework generalizes to any number of agents. Notably, we show that training only the leader model, without requiring that the agent team also be trained, is sufficient to achieve strong collaborative effectiveness and also makes our method considerably more scalable and practical than approaches requiring all agents to be trained.

Multi-Agent Team Inference The inference process unfolds iteratively over T rounds, as summarised in Figure 1. In the initial round (round 0), each agent independently generates a preliminary solution to a given

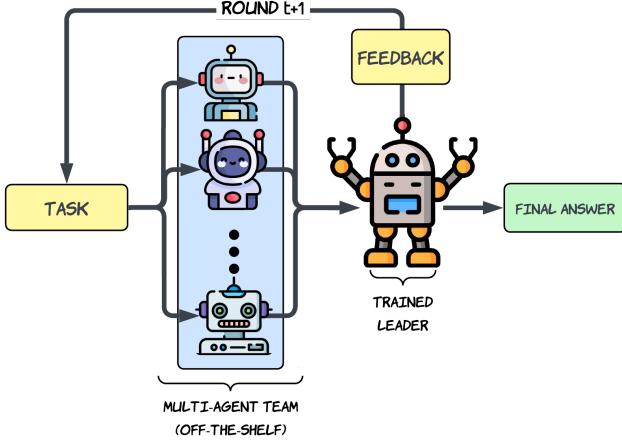


Figure 1 Overview of the proposed hierarchical multi-agent inference architecture. A user prompt is first processed by a team of K off-the-shelf agents whose intermediate generations are forwarded to a leader model trained using our MLPO pipeline. The leader’s output is then returned to the agent team together with the initial prompt, and this exchange is repeated for T iterations before the final answer is produced.

task input x . We denote Agent i ’s policy by π^{a_i} and its 0’th round preliminary solution by $s_i^{(0)} \sim \pi^{a_i}(x)$. The leader model, denoted by π_θ^L , then synthesizes these initial agent responses into its structured output, denoted by $z_L^{(0)}$, i.e.,

$$z_L^{(0)} \sim \pi_\theta^L \left(x, s_1^{(0)}, s_2^{(0)}, \dots, s_K^{(0)} \right).$$

This output encapsulates detailed reasoning within specific `<think>` tags and presents a consolidated solution within `<answer>` tags.

In subsequent rounds ($t = 1, 2, \dots, T - 1$), each agent revises its solution based on the leader’s previous output and its own earlier response:

$$s_i^{(t)} \sim \pi^{a_i} \left(x, s_i^{(t-1)}, z_L^{(t-1)} \right).$$

The leader then integrates these updated agent solutions to refine its reasoning and produce an updated consolidated output:

$$z_L^{(t)} \sim \pi_\theta^L \left(x, s_1^{(t)}, s_2^{(t)}, \dots, s_K^{(t)} \right).$$

This iterative cycle of feedback and refinement continues until the final round, after which the final solution is extracted from the leader’s output.

3.2 Multi-Agent Guided Leader Training

Our training approach comprises two phases similar to [20, 56]: Supervised Fine-Tuning (SFT), designed to develop the leader’s natural backtracking and self-correction abilities, followed by Group Relative Policy Optimization (GRPO) [72], aimed at enhancing multi-agent collaboration skills. Notably, our experimental results using Qwen2.5 7B Instruct as the leader indicate that the SFT step, while beneficial, is not strictly necessary before GRPO. Specifically, we observe that the largest accuracy gains for the leader model arise primarily from the GRPO phase itself, whereas preceding GRPO with SFT provides only modest incremental improvements (typically 1-2% accuracy increase).

3.2.1 Supervised Fine-Tuning

The SFT phase aims to train the leader model to reinforce its aggregation, self-correction and backtracking behaviours, which are crucial skills for effective collaboration and error recovery. To this end, we first run our multi-agent system with an untrained leader model. For each task in the training data, we generate one round of agent responses. Next, we query the untrained leader to use these agent responses to synthesize a

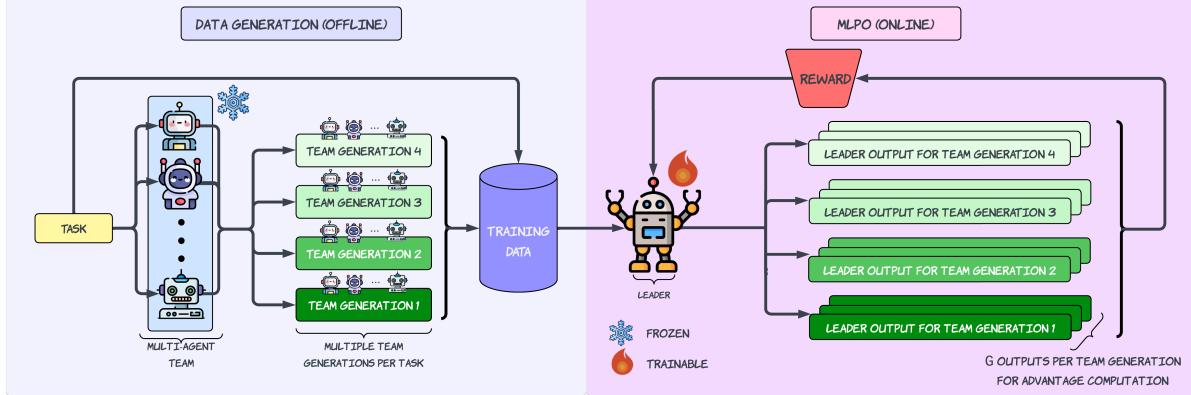


Figure 2 Outline of our Multi-agent guided Leader Policy Optimization (MLPO) pipeline.

final solution. For each set of agent responses, we query the leader 16 times to obtain diverse solutions to each prompt. We then filter out the tasks where all 16 leader responses are incorrect, as these likely represent tasks beyond the model’s current capabilities.

For the remaining tasks, we construct a new SFT dataset comprising generations with self-correction and backtracking, inspired by previous approaches explored in recent literature [20, 47, 56]. To achieve this, we first randomly select one correct and one incorrect leader response from the 16 leader generations. Next, an untrained leader model is prompted with these selected responses to generate a “*backtracked solution*” that mimics natural reasoning patterns, i.e. beginning with plausible but incorrect reasoning, then naturally self-correcting to arrive at the correct answer. The prompts explicitly encourage natural self-correction phrases such as “Wait, that doesn’t seem right” or “Let me reconsider this approach”. For tasks where all 16 responses are correct, we directly include the correct solutions without artificial backtracking. The leader model is then fine-tuned on this dataset of backtracked and correct solutions, aiming to enhance its ability to naturally self-correct and recover from reasoning errors.

3.2.2 Multi-Agent Guided Leader Policy Optimization

Following SFT, we employ our variant of Group Relative Policy Optimization (GRPO) [72], which we refer to as Multi-agent guided Leader Policy Optimization (MLPO), to train the leader π_θ^L specifically for multi-agent collaboration. In this phase, the goal is to learn effective aggregation and synthesis strategies. In classical GRPO, each prompt simply comprises the task to be solved (along with any accompanying instructions, such as “let’s think step-by-step”, etc.). In contrast, for MLPO, each leader’s prompt also includes the agents’ solutions to the given task. The loss for MLPO can therefore be expressed as:

$$\mathcal{J}_{\text{MLPO}}(\theta) = \mathbb{E}_{x \sim P_X, s \sim \pi^a(x), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}^L(x, s)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \left\{ \min \left(r_{i,t} \hat{A}_{i,t}, \text{clip} \left(r_{i,t}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right) \right\} \right].$$

Here, $r_{i,t} = \frac{\pi_\theta^L(o_{i,t}|x, s, o_{i,<t})}{\pi_{\theta_{\text{old}}}^L(o_{i,t}|x, s, o_{i,<t})}$ is the importance ratio and $\hat{A}_{i,t} = R_i - \text{mean}(\mathbf{R})$ denotes the advantage of the i ’th response where R_i is the corresponding reward. Moreover, x represents the task input (e.g. a mathematics problem), $s = \{s_i^0\}_{i=1}^K$ denotes the set of agent solutions and o_i is a sample from the current leader $\pi_{\theta_{\text{old}}}^L$. The hyperparameter ε denotes the PPO-style clipping threshold [50, 72]. Note that we have used the modifications to the GRPO loss proposed in Dr. GRPO [40] due to its improved training efficiency and stability.

Training Data Generation To generate our MLPO training data, we query each of the K agents to independently generate 4 solutions to each task in the training split (as shown in Figure 2). This results in a total of $4K$ responses per task. From these responses, we create 4 distinct training prompts per task, each consisting of the original task paired with exactly one response from each agent. This data generation strategy has been validated experimentally in prior work [66] as well as in our ablation study in Section 4.3,

demonstrating that generating multiple training prompts per task significantly improves the test accuracy of the leader model. Moreover, our experiments reveal that increasing the number of responses beyond 4 per agent per task does not substantially further enhance the performance of the leader model.

Next, we also filter out “easy” tasks, specifically those tasks where at least 75% of the $4K$ agent responses are correct. This filtering directs training attention toward scenarios requiring strong leadership and aggregation skills, which further helps boost the leader’s performance, as we demonstrate empirically in Section 4.3. Additionally, while the tasks are randomly shuffled during training, the four prompts corresponding to each task remain grouped. This strategic ordering allows the leader model to observe different agent team responses (and reasoning) to the same task consecutively, and enhances learning efficiency. Figure 10 show that this grouping of prompts by tasks makes training much more stable compared to random ordering of data.

Throughout the MLPO training process, the leader model receives prompts containing the original tasks and agent responses. The leader must then produce structured outputs, clearly separating their detailed reasoning within `<think>` tags and providing definitive answers within `<answer>` tags. Additionally, similar to various recent works on RL finetuning [40, 69, 72], we used a simple heuristic reward to avoid any reward hacking, which consisted of a 0 and +1 reward for incorrect and correct answers respectively, as well as a formatting reward for properly structuring responses within the `<think>` and `<answer>` tags.

Multi-round training with MLPO+ Initially, our pipeline trains the leader solely on agent-team responses from round 0, i.e., responses generated before any interaction between the leader and the agent team has occurred. This setup enables efficient offline training, as round-0 responses can be precomputed prior to leader optimization. While this training scheme is already effective, it does not leverage the full distribution of team responses that arise at inference.

To address this, we also consider a multi-round extension of MLPO, which we call MLPO+, that incorporates an additional training phase using team responses from later rounds ($t > 0$). These responses resemble more closely those seen during inference, as they are shaped by interaction with the leader. To operationalize MLPO+ efficiently, we treat the multi-round training as a continuation phase that follows the initial MLPO training. After the leader is trained using round-0 responses, it is deployed to interact with the agent team over multiple rounds. These interactions are then recorded and used to construct a new training set. Importantly, this additional data can be generated offline, just like the round-0 responses, introducing little extra overhead.

In practice, we first filter out “easy” tasks (those where the leader and team achieve over 50% accuracy after two rounds). For the remaining tasks, we construct a new dataset consisting of two sets of agent responses from rounds 0, 1, and 2. The leader is then further optimized on this dataset using the same MLPO loss. This setup allows the leader to refine its strategy based on more realistic team behaviors that emerge during iterative interaction.

Flexible Zero-Shot and Multi-Agent Inference Interestingly, the benefits of our multi-agent guided training extend beyond collaborative settings at inference time. We discovered through our empirical analysis (detailed in Section 4.2.3) that training the leader model using our MLPO framework significantly enhances its zero-shot inference capabilities as well, even without employing the agent team during inference. This enables us to achieve higher accuracy than models trained with standard single-agent GRPO approaches, at no additional computational cost during inference. Moreover, when inference-time compute resources permit, deploying the leader alongside the agent team further boosts accuracy, outperforming state-of-the-art multi-agent methods. Thus, our framework flexibly provides improved accuracy in both single- and multi-agent inference settings.

4 Experimental Results

4.1 Experimental Setup

Datasets: We evaluate our hierarchical multi-agent collaboration framework on three benchmarks that cover a variety of factual- and reasoning-based tasks: Big-Bench Hard (**BBH**) [54], **MATH** [22], and Massive Multitask Language Understanding (**MMLU**) [21].

Model Configuration: In our main experiments, we use Qwen2.5 7B Instruct (**Qwen-2.5** [49]) as our leader model, with a heterogeneous agent team comprising Llama 3.1 8B Instruct (**Llama-3.1** [19]), Gemma2 9B Instruct (**Gemma-2** [55]), and Qwen2.5 7B Instruct. In Section 4.2.4 we provide results for different choices of agent teams, and in Appendix A we provide results for different leaders (Gemma-2 and Llama-3.1).

Baselines: We compare our method MLPO with several baselines. We delineate these baselines into two main categories: training-based and training-free. For training-free baselines, we compare to **Self-Reflection** [42], Multi-Agent Debate (**MAD**) [10], our pipeline with **untrained leader**, and **zero-shot** inference, which invokes an untrained single LLM once with no multi-agent team support. For training-based baselines, we compare to **ACC-Collab** [13], **SelectLLM** [43], **SCoRe** [30], a **Deferral leader** trained to always defer to one agent in the team, and **zero-shot GRPO** [72] which invokes a single LLM (trained using standard zero-shot GRPO) once with no multi-agent team support. For all iterative methods, we fix the number of inference rounds as $T = 5$.

4.2 Main Results

In this section, we present an extensive empirical analysis designed to validate and display the capabilities of our proposed MLPO framework. Specifically, our experiments address the following points:

- **Our Proposed MLPO vs. Existing Baselines:** We evaluate our method extensively against existing approaches, demonstrating how our collaborative reasoning approach significantly enhances performance.
- **Test-Time Scaling for Improved Performance:** We investigate how additional computational resources during inference can further amplify the performance gains of our proposed multi-agent team.
- **Multi-agent Guided Training with Zero-Shot Inference:** We demonstrate that our multi-agent guided training pipeline (MLPO) also substantially enhances zero-shot performance, surpassing models trained with conventional single-agent GRPO methods without incurring extra inference-time costs.
- **Leader-Agent Interaction Dynamics:** We analyze the interaction dynamics between the leader model and its agent team, showing how our method effectively utilizes agent diversity to achieve robust, accurate, and consistently superior outcomes across various task categories and difficulty levels.

4.2.1 Our Proposed MLPO vs. Existing Baselines

Type	Method	MMLU	BBH	MATH
TRAINING-FREE	ZERO-SHOT*	0.734 ± 0.006	0.733 ± 0.010	0.666 ± 0.002
	SELF-REFLECT [42]	0.746 ± 0.003	0.766 ± 0.011	0.681 ± 0.003
	MAD (DIV) [10]	0.771 ± 0.002	0.785 ± 0.002	0.653 ± 0.009
	MAD (HOM) [10]	0.759 ± 0.003	0.799 ± 0.016	0.720 ± 0.009
	UNTRAINED LEADER	0.731 ± 0.008	0.764 ± 0.007	0.697 ± 0.007
TRAINING-BASED	ACC-COLLAB [13]	0.761 ± 0.004	0.802 ± 0.003	0.698 ± 0.008
	SELECTLLM* [43]	0.770 ± 0.001	0.768 ± 0.001	0.651 ± 0.001
	SCoRe [30]	0.752 ± 0.005	0.828 ± 0.004	0.721 ± 0.004
	DEFERRAL LEADER	0.750 ± 0.004	0.775 ± 0.007	0.720 ± 0.003
	ZERO-SHOT GRPO* [72]	0.742 ± 0.004	0.791 ± 0.008	0.712 ± 0.005
OUR RESULTS	SELF-REFLECT GRPO [72] + [42]	0.747 ± 0.003	0.814 ± 0.006	0.718 ± 0.006
	SFT + MLPO LEADER (ZERO-SHOT ACC W/O TEAM)*	0.757 ± 0.007	0.855 ± 0.008	0.729 ± 0.008
	SFT LEADER	0.741 ± 0.008	0.762 ± 0.008	0.713 ± 0.007
	MLPO LEADER	0.759 ± 0.006	0.865 ± 0.005	0.750 ± 0.009
SFT + MLPO LEADER		0.782 ± 0.006	0.882 ± 0.005	0.762 ± 0.005

Table 1 Accuracy of each method on MMLU, BBH, and MATH benchmarks, with ± 2 standard errors. Here, * denotes methods not using iterative inference.

Having outlined our model configuration and selected baselines, we now present the empirical evaluation of our proposed method. Specifically, we systematically investigate our MLPO approach relative to standard single-agent and multi-agent baselines, highlighting the advantages of incorporating collaborative reasoning

during both training and inference. Table 1 summarizes our experimental results comparing MLPO with existing baselines.

Our method substantially outperforms both single-agent training baselines (e.g., SCoRe [30], GRPO [72]) and multi-agent training baselines (e.g., ACC-Collab [13], SelectLLM [43], Deferral Leader). This performance gain stems from the fact that, during both training and inference, the leader is guided by a diverse set of strategies and solutions proposed by the agent team. In contrast, single-agent approaches operate in isolation, lacking access to this breadth of insight, as they must rely on their own solutions.

Beyond demonstrating superior performance compared to baseline methods, we make two additional key observations regarding the behaviour of our leader. First, upon evaluating the training paradigms, we observe that supervised fine-tuning (SFT) alone does not consistently provide substantial performance improvements over an untrained leader, underscoring the importance of employing reinforcement learning within our MLPO framework. Second, to ensure our leader genuinely develops a strategic decision-making capability rather than merely replicating accurate agent responses, we compare it against two baselines: SelectLLM [30], which identifies the optimal agent subset based solely on the input question, and the Deferral Leader, which selects the best response given both the question and agent outputs. Our SFT + MLPO’ed leader consistently outperforms these baselines across all evaluated domains, providing evidence of its capability to formulate independent strategies beyond mere imitation. This conclusion is further corroborated by Figure 7, where our leader maintains significantly higher accuracy even in scenarios where all individual agents fail.

However, a natural question to ask is whether our improvements are merely a consequence of our method being allowed to use more generations during inference. To address this concern, we perform thorough ablation studies in the following subsections, which show that our method retains its advantage even when controlling for inference-time compute.

4.2.2 Test-Time Scaling for Improved Performance

Here, we investigate performance when all methods are allotted the same amount of inference time compute, and explore how additional inference-time compute can be leveraged to further improve the performance of models trained with MLPO under a fixed inference budget. In particular, prior work has shown that self-consistency, i.e., majority voting over multiple generations, can enhance model accuracy [59]. For fair comparison, we constrain all methods (including iterative and multi-agent baselines) to a maximum of 40 total LLM generations at inference time. For iterative methods, we apply majority voting only over responses from the final inference round, ensuring that the highest-quality outputs are used. For example, in MAD with 3 agents, we take each agent’s final-round response as a vote, resulting in 3 votes for each run of the MAD pipeline, whereas, for our pipeline, we only use the leader’s answer in the final round as a vote for each run.

Figure 3 reports the accuracy of each method under this consistent sample budget with majority voting. These results show that our method achieves superior performance across all three datasets when inference-time compute is scaled. Notably, our approach requires only two rounds of interaction to reach this performance level, outperforming baselines that require up to five rounds.

We hypothesize that this improvement arises from how different methods trade off bias and variance when scaling inference-time compute. In general, iterative pipelines can be scaled either sequentially by increasing the number of rounds per run, or in parallel by generating more independent runs which are then bootstrapped. Sequential scaling tends to reduce variance (as agents converge on a shared answer), but may increase bias if the model converges to an incorrect solution [5, 12, 62]. In contrast, parallel scaling introduces more response diversity (i.e., higher variance), which can mitigate bias and often improves performance when using techniques such as majority voting [8, 53, 59].

Our method may benefit particularly from parallel scaling due to having two distinct sources of diversity: the variability in responses from the agent team and the stochasticity in the leader’s own generations. This dual source of response diversity could help explain why our pipeline yields higher majority-vote accuracy under fixed sample budgets, compared to baselines that lack one or both of these mechanisms.

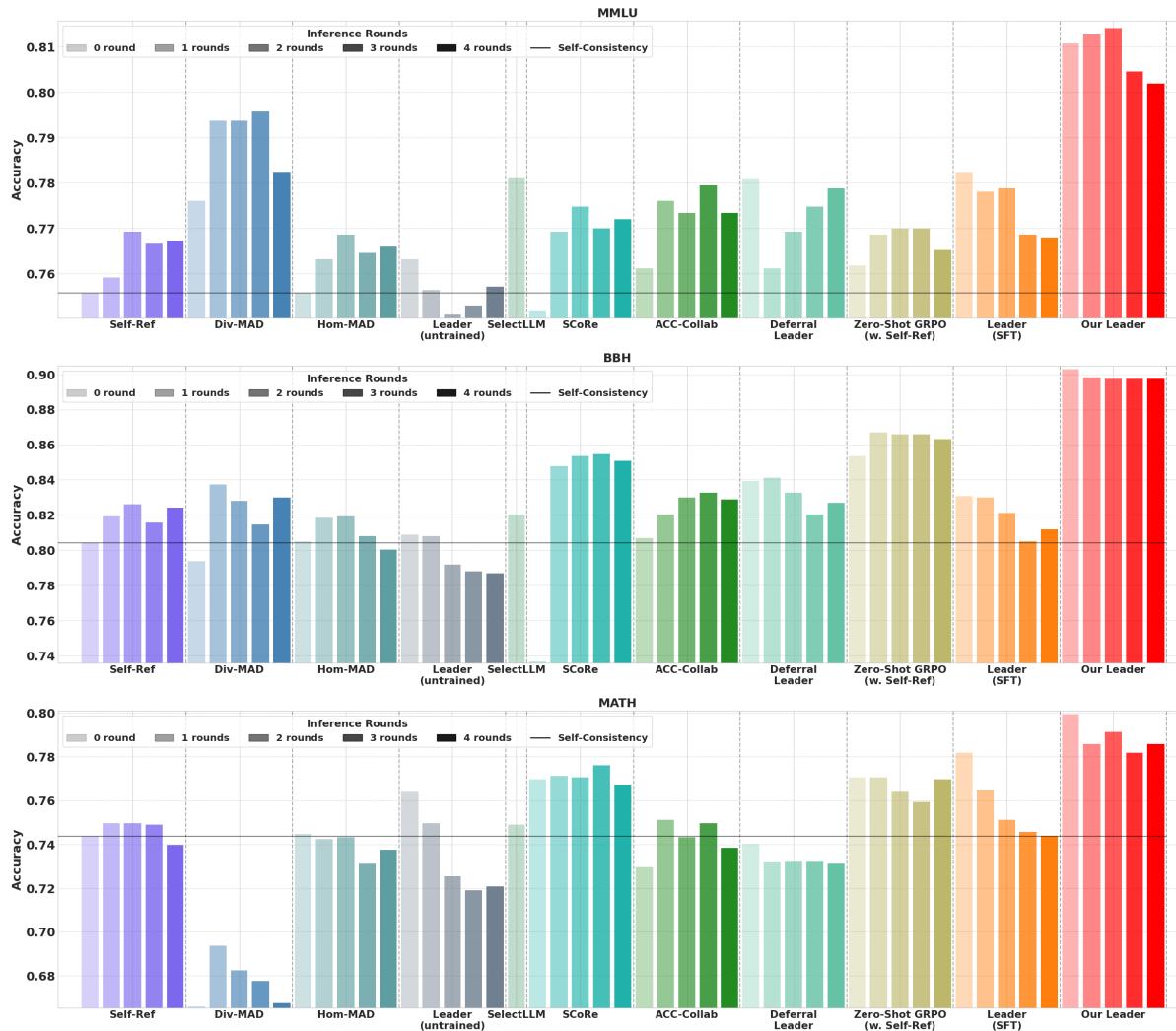


Figure 3 Majority vote performance when each method can use at most 40 total LLM generation samples. For iterative methods, we only use the final round answers for majority vote; SelectLLM is not an iterative method. Traditional self-consistency [59] is shown as horizontal line. The leader model trained with SFT + MLPO is denoted as Our Leader.

4.2.3 Enhanced Zero-Shot Capabilities

Remarkably, as mentioned previously, our multi-agent trained leader also exhibits superior performance even during zero-shot inference (without multi-agent assistance), outperforming models trained with standard SFT + GRPO pipelines, hence incurring **no additional inference cost**; see Figure 4.

This finding suggests that MLPO enhances “*knowledge acquisition*” during training, potentially through improved exploration of alternative solutions proposed by the agent team. The key difference between our multi-agent guided training framework MLPO, and standard single-agent training frameworks such as GRPO, lies in the training prompts: the former includes agent solutions alongside the question, while the latter contains only the question and instructions. The inclusion of diverse agent responses appears to create a richer training signal that improves the leader’s reasoning capabilities even when later deployed in isolation.

An astute reader may ask if this enhanced “*knowledge acquisition*” is primarily responsible for the improved performance of our multi-agent system. In other words, *could it be that even when deployed with the multi-agent team, the trained leader is simply ignoring the agent responses at inference time and using its acquired*

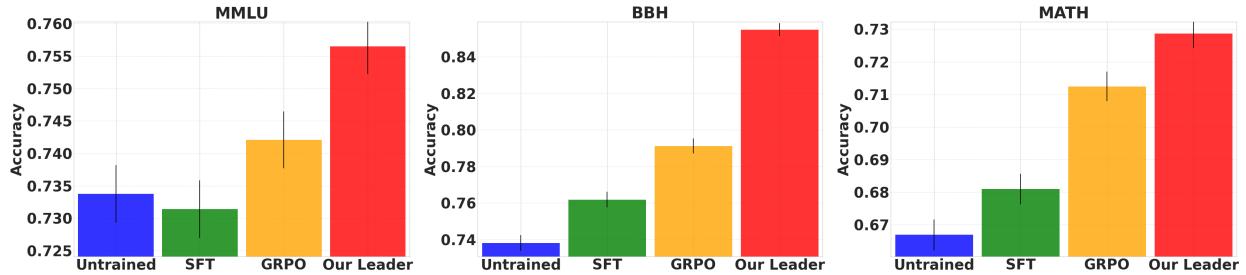


Figure 4 Zeroshot performance of Qwen when untrained, SFTed or GRPOed on zero-shot data, and trained via our MLPO pipeline (Our Leader). The error-bars show ± 2 standard errors.

knowledge to drive the accuracy gains? Our results in Table 1 show that this is not the case, since the leader’s accuracy improves even further when it is deployed with the multi-agent team at inference time (as compared to its zero-shot inference performance). Not only has the leader acquired knowledge, but also utilizes its implicit evaluation and aggregation skills to synthesize a higher-quality final solution when given access to the agent team’s responses.

4.2.4 Leader and Agent Team Interaction Dynamic

Leader vs Agent Team across Category and Difficulty An ideal leader can leverage the unique strengths of the multi-agent team to produce solutions which are better than any one model could alone. Figure 5 shows the performance of our trained leader and each individual agent in the leader’s team. Within these plots, we see two interesting observations.

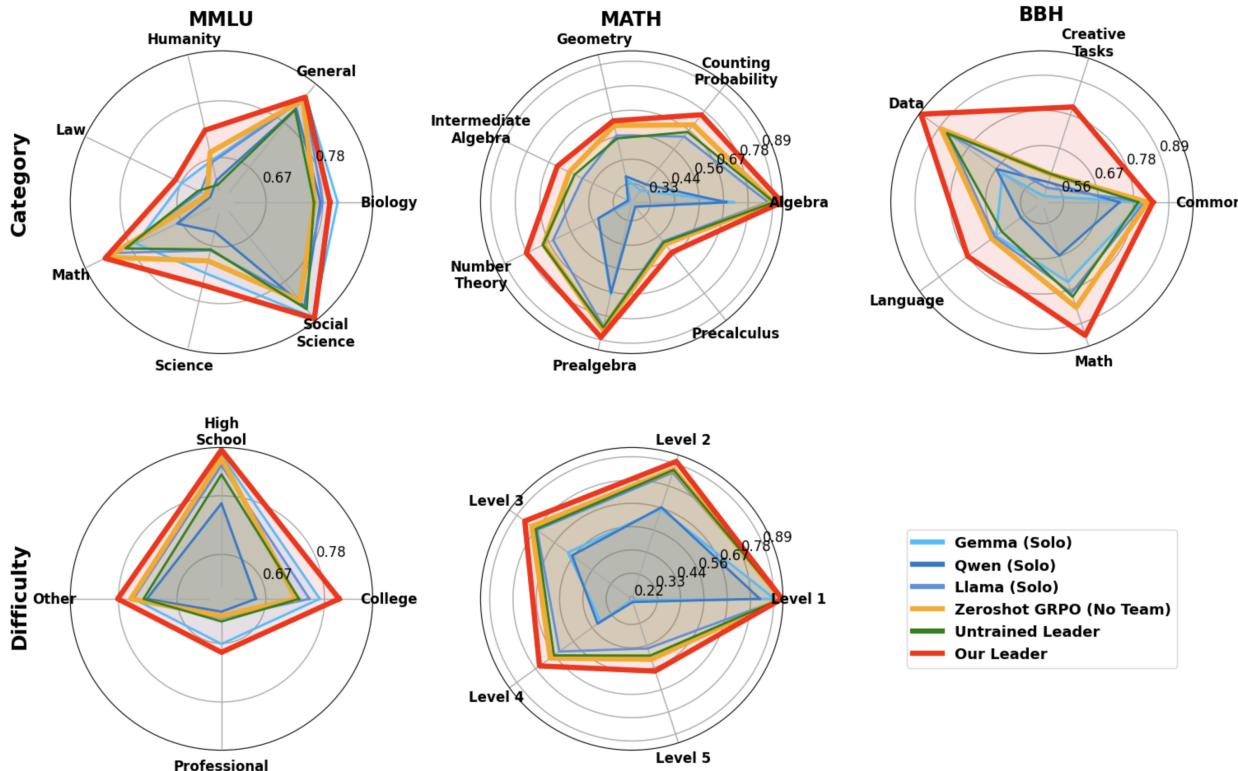


Figure 5 Our leader trained with MLPO, compared with an untrained leader, to zero-shot GRPO, and individual team performance, per category (top) and difficulty level (bottom) on MMLU (left) MATH (center) and BBH (right). Accuracy is reported after 5 rounds of inference. Note that BBH does not have difficulty gradations.

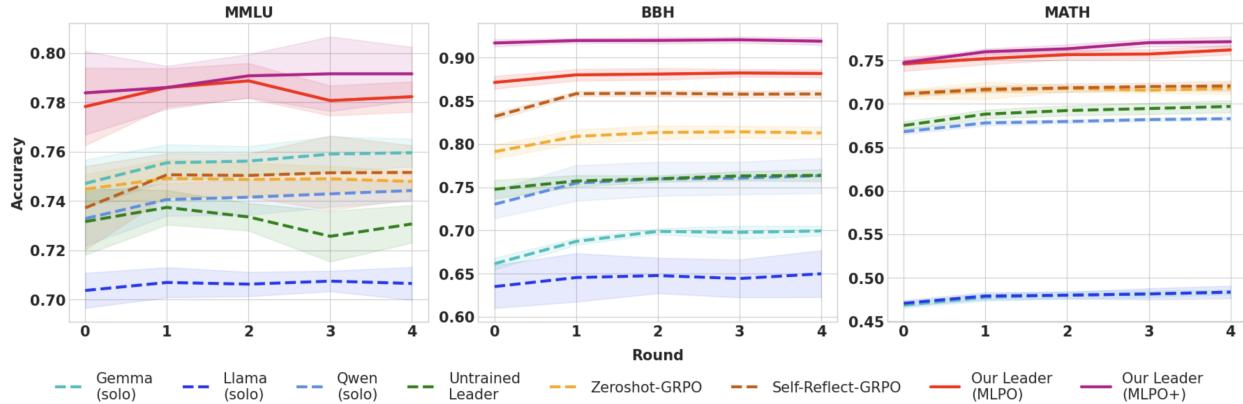


Figure 6 Performance over multiple rounds of interaction. Shaded region represents ± 2 standard errors.

First, the multi-agent pipeline with an untrained leader (green) struggles to surpass each of the individual agents (blue), both per-category and per-difficulty across each of the benchmarks. In contrast, the multi-agent team with an MLPO trained leader can surpass the performance of each agent as well as the untrained leader in nearly all categories and difficulty (occasional ties with the best agent). Second, the model trained with zero-shot GRPO (orange) also struggles to dominate the untrained leader, as well as each agent, over categories and difficulties. The per-category and per-difficulty success of the zero-shot model is far more task-dependent than that of our trained leader. In particular, when examining MMLU, we see that the untrained agents or the untrained leader beats the zero-shot GRPO model on 4 out of 7 categories (Science, Social Science, Biology, and Law) and 2 out of 4 difficulties (professional and college). However, our trained leader dominates in all difficulties and 6 out of 7 categories (losing only on Biology), highlighting that the leader trained via our MLPO framework successfully acquires a balanced and robust skill set. MLPO consistently outperforms both the (trained) individual agents as well as the team with an untrained leader across almost all categories and difficulty levels.

Per-Round Improvement Next, we examine the accuracy improvements achieved through iterative rounds of discussion between the leader and the agent team, as well as the self-reflection performance of individual agents (solo). Figure 6 (and previously shown Figure 3) presents the accuracy progression for each method across these multiple rounds. Notably, even though our leader model is only trained on initial round data (round 0), we consistently observe slight accuracy improvements in subsequent rounds. One possible explanation for this improvement is that in later rounds, agents benefit from additional information provided by the leader’s earlier responses, enabling them to refine their own answers. This, in turn, may further assist the leader model in correcting its previous errors. However, further investigation is required to fully understand this phenomenon.

Multi-Round Training with MLPO+ We now discuss a complementary phenomenon: the ability to further improve leader performance through additional training. We evaluate the performance of MLPO+, which fine-tunes the leader on agent-team responses from later rounds ($t > 0$), and find that it yields consistent improvements in leader performance.

The performance boost appears to stem from two complementary factors: first, the leader is trained on responses that better reflect those it will encounter during deployment, and second, it learns to better synthesize information from agent teams that have already been shaped by its prior guidance. Notably, these gains come with minimal additional computational cost, as the extra training data can still be collected offline. Overall, these results highlight the value of iterative supervision: exposing the leader to more realistic interactions and more diverse solutions during training increases the leader’s efficacy.

Leader Robustness to Agent Team’s Ability An effective leader should consistently produce correct solutions, by leveraging accurate responses from agents when available, and overriding incorrect ones when necessary. In Figure 7, we compare the performance of various leaders as a function of agent correctness. To ensure a fair

	MMLU	BBH	MATH
MLPO	0.782 ± 0.006	0.882 ± 0.005	0.762 ± 0.005
MLPO+	0.792 ± 0.005	0.920 ± 0.004	0.771 ± 0.005

Table 2 Results of a leader trained with MLPO and MLPO+ after 5 rounds of inference

comparison, we report results only for round 0, where agent responses are not influenced by the leader. While all leaders perform well when all agents are correct, their accuracy drops substantially when agent correctness is low. Notably, our trained leader shows strong performance particularly when only few agents are correct, highlighting our method’s robustness to low team performance. This effect is especially pronounced on the BBH dataset.

One possible explanation for this improved robustness could be better knowledge acquisition by the leader model during MLPO, enabling it to independently generate correct answers without relying solely on the agent solutions at inference time. This hypothesis aligns with the observed increase in zero-shot accuracy as previously mentioned in Figure 4. However, knowledge acquisition alone does not fully explain our leader’s performance. Specifically, we observe additional improvements when correct agent responses are available at inference time. This indicates that the leader has learned not only to independently solve problems but also to effectively evaluate the quality of agent inputs, discerning when to trust their advice and when to override incorrect team responses.

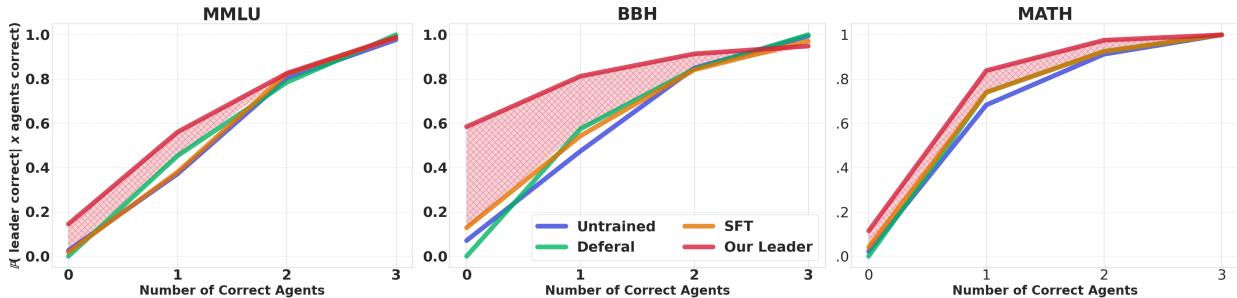


Figure 7 Leader performance conditioned on the number of correct agents in the team. Shaded regions represent the degree to which our leader outperforms the next best method.

Leader’s Utilization of Agent Information To better understand the mechanisms behind our trained leader’s superior performance, we analyze how it utilizes the information provided by the agent team. We conducted ablation experiments, selectively providing either agent reasoning, final answers, or both, to the leader during inference. The results of these experiments are summarized in Figure 8.

Our findings consistently indicate that, in most scenarios, providing the leader with both the agents’ reasoning processes and their final answers achieves the highest performance. This suggests that the leader effectively leverages the complete context provided by the agents, benefiting from both their detailed thought processes and their concluded solutions.

Interestingly, in the majority of cases, reasoning-only input achieves better performance than final-answer-only input. This implies that the leader primarily derives value from understanding the agents’ underlying reasoning rather than merely aggregating their final decisions. However, notable exceptions exist: for the BBH dataset, final-answer-only surpasses reasoning-only performance, although providing both components remains optimal. When deploying leaders trained with MLPO, utilizing agent full responses generally elicits the best leader performance.

These observations indicate that the relative importance of reasoning versus final answers depends critically on both the dataset characteristics and the specific training regime. Including final answers may sometimes bias the leader towards incorrect conclusions, whereas reasoning-only input often encourages independent

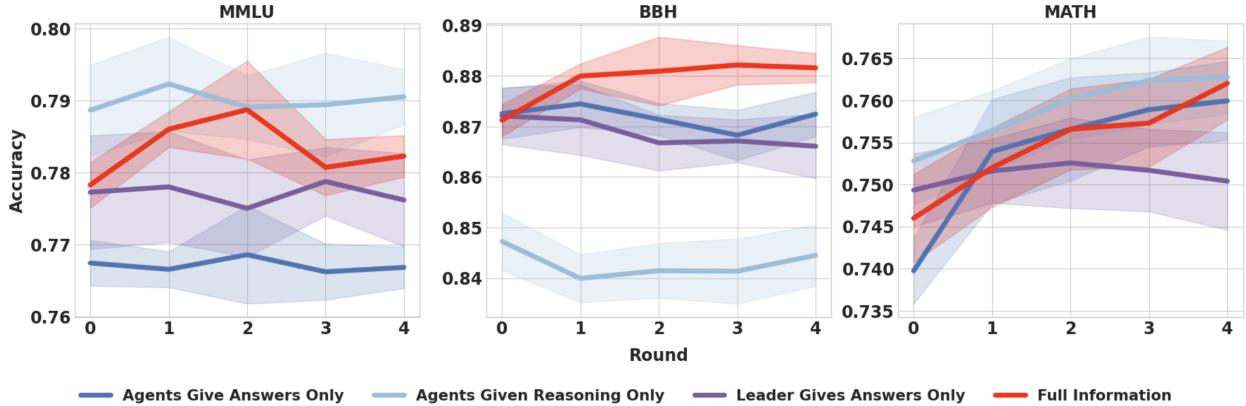


Figure 8 Leader performance over rounds of inference, varying by which information is shared between the agent team and the leader. Shaded region represents ± 2 standard errors.

verification and robust inference. Nonetheless, overall, providing both reasoning and final answers to the leader consistently yields optimal or near-optimal results across diverse scenarios.

Team Composition To better understand the interaction dynamics between the leader and its team, we evaluate leader performance under different choices of agent teams; namely the diverse team used in our main results (Qwen-2.5, Llama-3.1, and Gemma-2), and homogeneous teams consisting of 3 copies of each model (e.g., 3 copies of Gemma-2). For each team configuration, a leader is trained specifically for that team. Figure 9 reports both the average leader accuracy and the average team accuracy after five rounds of interaction.

Two key observations emerge from these results. First, leaders paired with homogeneous teams exhibit more task-dependent performance. For example, on BBH dataset, the leader paired with Llama team achieves highest accuracy among all homogeneous teams, while on MATH, it achieves the lowest. However, the leader paired with the heterogeneous team consistently matches or exceeds the performance of leaders paired with homogeneous teams. This highlights the benefit of deploying MLPO with a diverse agent team. Second, the gap between leader and team performance can be substantial, even after the team has had extensive interaction with the leader. This underscores the leader’s critical role in providing high-quality solutions even when the underlying team struggles.

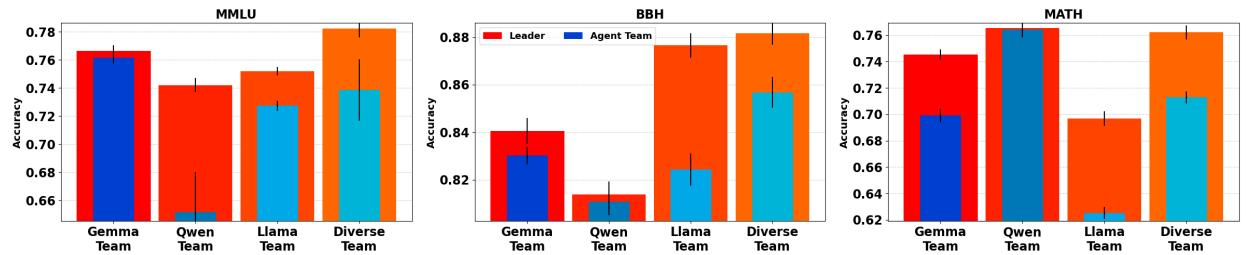


Figure 9 Leader accuracy (warm colors) and average team accuracy (cool colors) after 5 rounds of interaction.

4.3 Best Practices for Multi-Agent Guided Training

We conclude by highlighting key considerations in our multi-agent guided training paradigm that helped guide the design of our method. Recall that during training, the leader observes the responses generated by the agent team before producing its own answer. Thus, the leader is trained on prompt-response pairs of the form `(task + agent_responses, leader_response)`, in contrast to standard single-agent training, which relies solely on the task input. This distinction gives rise to two natural questions:

METHOD	MMLU	BBH	MATH
UNTRAINED	0.734 ± 0.006	0.733 ± 0.010	0.666 ± 0.002
GRPO ZEROSHOT DATA	0.747 ± 0.003	0.814 ± 0.006	0.718 ± 0.006
SCoRe	0.752 ± 0.005	0.828 ± 0.004	0.721 ± 0.004
GRPO SELF-REFLECT DATA	0.762 ± 0.004	0.857 ± 0.006	0.720 ± 0.004
ZEROSHOT AS LEADER	0.742 ± 0.003	0.783 ± 0.002	0.729 ± 0.007
OUR LEADER W. ALL QWEN-2.5 TEAM	0.742 ± 0.004	0.816 ± 0.008	0.767 ± 0.003
OUR LEADER W. DIVERSE TEAM	0.782 ± 0.006	0.882 ± 0.005	0.762 ± 0.005

Table 3 Accuracy after 5 rounds of inference.

1. To what extent do the alternative solutions proposed by the agents contribute to the effectiveness of the trained leader?
2. Given that the agent team can generate a multitude of different responses for a single task, how many sets of such response sets should the leader be exposed to during training for each task?
3. Does filtering out easy questions improve overall training efficacy?

Importance of Alternative Solutions To address the first question above, we assess the importance of alternative solutions provided by the agent team during both training and inference. As previously noted, the composition of the agent team, and thus the diversity (and quality) of solutions presented to the leader, can significantly affect performance. To investigate this, we evaluate five configurations that vary in terms of how alternative solutions are presented to the trained model (both at training and at inference time).

- *GRPO Zeroshot Data*: No alternative solutions at training or inference; model sees only its own outputs.
- *SCoRe and GRPO on Self-Reflection Data*: Model is trained and inferred with its own solutions.
- *Zeroshot as Leader*: No alternative solutions during training; observes agent team responses at inference.
- *Our Leader w. All Qwen-2.5 Team*: Trained and inferred with homogeneous team of Qwen-2.5 agents.
- *Our Leader w. Diverse Team*: Trained and inferred with heterogeneous team of three distinct agents.

Table 3 presents the performance of each method after five rounds of inference. Among these, the best performance is achieved by the leader trained, via our method, with responses from a diverse team of agents. Interestingly, we observe no clear advantage when models are trained on their own solutions (as in SCoRe and GRPO Self-Reflect) compared to those trained on homogeneous agent team responses. These results highlight the importance of sourcing alternative solutions from multiple distinct agents, which results in the highest-quality responses from the leader.

Leader’s Exposure to Distinct Agent Solutions Per Task During Training We next examine how the number of distinct solution sets per task provided to the leader during training affects performance. We use BBH as an exemplar for this ablation. Specifically, we train the leader using 1, 4, and 8 sets of agent responses per task (in our main experiments, we adopt 4 sets per task). To ensure a fair comparison, all variants use the same total number of training examples. Table 4 reports leader accuracy under each setting. We observe that exposing the leader to multiple solution sets improves performance, though gains begin to plateau beyond 4 sets. This suggests that diverse agent responses help the leader generalize to varied team behaviors, but that additional diversity beyond a certain point offers diminishing returns.

We also examine the impact of training-time task difficulty on leader performance. Specifically, we consider how “easy” a given task is for the agent team, independent of the leader. Any task where at least 75% of the agent responses are correct is removed from the training data. Training on these filtered tasks leads to improved leader performance at test time. One possible explanation is that including tasks on which the

NUMBER OF SOLUTION SETS SEEN AT TRAINING		
1 SET	4 SETS	8 SETS
0.890 ± 0.005	0.917 ± 0.003	0.906 ± 0.002

Table 4 Leader accuracy for different numbers of solution sets per task during training (BBH), ± 2 standard errors.

agent team already performs well may cause the leader to become overly reliant on agent responses, reducing its incentive to critically evaluate or override incorrect solutions.

UNFILTERED TRAINING	FILTERED TRAINING
0.869 ± 0.004	0.882 ± 0.005

Table 5 Accuracy of leader trained with and without difficulty-based filtering for BBH, with ± 2 standard errors.

It is worth noting these ablations cannot be directly compared to the results in the main body as we use a different amount of training data as to ensue fair comparisons between the different ablations.

Solution Set Ordering Next, we discuss the ordering in which the leader is exposed to the different sets of solutions. For our main experiments, we sample four sets of agent solutions for each task x . In Figure 10 we see difference in reward curves for training with shuffled solution sets, and grouped solution sets (i.e., the leader sees all four solution sets in a single batch). We find that grouping solutions by task makes training much more stable and results in higher reward for the trained leader.

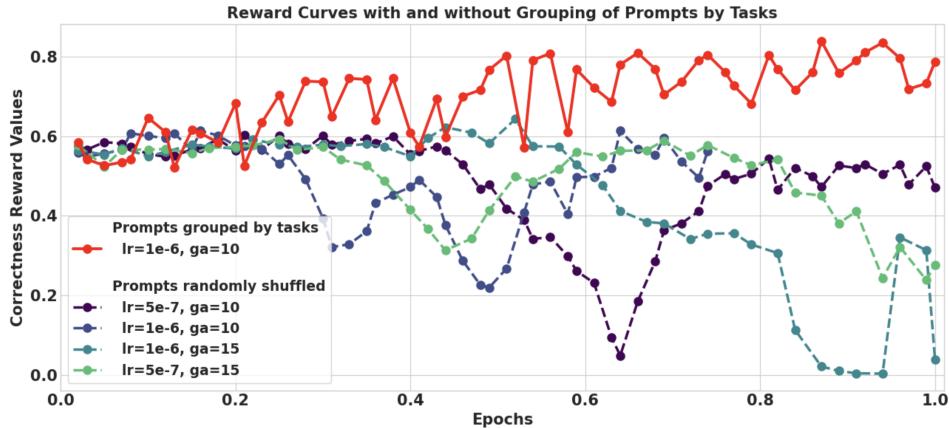


Figure 10 Reward curves for training with grouped solution sets vs randomly shuffled solution sets. Learning rate and gradient accumulation steps are denoted respectively as lr and ga .

Alternative Choice of Leader Lastly we remark on our choice of model for the trained leader. Throughout the main body, we present results when using Qwen-2.5 as the leader. In Appendix A we present our main set of results for alternative leaders. We observe similar results, utilizing our multi-agent guided training results in the highest efficacy, although we observe less performance increase using Gemma-2 and Llama-3.1 as the leader. This suggested that the most effective leaders may be models capable of more advanced reasoning, such as Qwen-2.5. This matches with intuition as an effective leader must be skilled at reasoning through the agent responses.

5 Conclusion

We introduce a hierarchical multi-agent framework leveraging our proposed Multi-Agent Guided Leader Optimization (MLPO) method, specifically designed to enhance the collaborative reasoning capabilities of large language models (LLMs). By explicitly training only a single leader model, our approach introduces novel multi-agent guided training and inference techniques, effectively integrating diverse insights from a team of untrained peer agents. Extensive empirical results on challenging benchmarks demonstrate significant performance improvements over single-agent and other multi-agent baselines.

Nevertheless, our method also has certain limitations. These include increased context lengths required both during training and inference, higher computational demands at inference time for optimal performance, and reduced parallelizability due to sequential leader-agent interactions. Addressing these limitations could involve exploring strategies like selective agent querying and caching to reduce training/inference load as well as different agent team combinations to reduce the number of rounds needed. Despite these constraints, our framework remains highly adaptable. It supports both parallel and sequential scaling strategies as needed, and notably, achieves strong performance even when deployed without the agent team at inference time.

References

- [1] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczek, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.
- [2] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, and et al. On the opportunities and risks of foundation models. [arXiv preprint arXiv:2108.07258](#), 2021.
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and et al. Language models are few-shot learners. [arXiv preprint arXiv:2005.14165](#), 2020.
- [4] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. [arXiv preprint arXiv:2308.07201](#), 2023.
- [5] Edward Y Chang. Socrasynth: Multi-llm reasoning with conditional statistics. [arXiv preprint arXiv:2402.06634](#), 2024.
- [6] Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. [arXiv preprint arXiv:2309.13007](#), 2023.
- [7] Justin Chih-Yao Chen, Sukwon Yun, Elias Stengel-Eskin, Tianlong Chen, and Mohit Bansal. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. [arXiv preprint arXiv:2503.05641](#), 2025.
- [8] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. [arXiv preprint arXiv:2311.17311](#), 2023.
- [9] Xiangxiang Dai, Jin Li, Xutong Liu, Anqi Yu, and John Lui. Cost-effective online multi-llm selection with versatile reward models. [arXiv preprint arXiv:2405.16587](#), 2024.
- [10] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.
- [11] Sugyeong Eo, Hyeonseok Moon, Evelyn Hayoon Zi, Chanjun Park, and Heuiseok Lim. Debate only when necessary: Adaptive multiagent collaboration for efficient llm reasoning. [arXiv preprint arXiv:2504.05047](#), 2025. URL <https://arxiv.org/abs/2504.05047>.
- [12] Andrew Estornell and Yang Liu. Multi-llm debate: Framework, principals, and interventions. [Advances in Neural Information Processing Systems](#), 37:28938–28964, 2024.
- [13] Andrew Estornell, Jean-Francois Ton, Yuanshuo Yao, and Yang Liu. Acc-collab: An actor-critic approach to multi-agent llm collaboration. [arXiv preprint arXiv:2411.00053](#), 2024.

- [14] Andrew Estornell, Jean-Francois Ton, Yuanshun Yao, and Yang Liu. Acc-collab: An actor-critic approach to multi-agent llm collaboration, 2024.
- [15] Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. Don’t hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. [arXiv preprint arXiv:2402.00367](#), 2024.
- [16] Shangbin Feng, Taylor Sorensen, Yuhan Liu, Jillian Fisher, Chan Young Park, Yejin Choi, and Yulia Tsvetkov. Modular pluralism: Pluralistic alignment via multi-llm collaboration. [arXiv preprint arXiv:2406.15951](#), 2024.
- [17] Shangbin Feng, Wenxuan Ding, Alisa Liu, Zifeng Wang, Weijia Shi, Yike Wang, Zejiang Shen, Xiaochuang Han, Hunter Lang, Chen-Yu Lee, et al. When one llm drools, multi-llm collaboration rules. [arXiv preprint arXiv:2502.04506](#), 2025.
- [18] Shangbin Feng, Zifeng Wang, Palash Goyal, Yike Wang, Weijia Shi, Huang Xia, Hamid Palangi, Luke Zettlemoyer, Yulia Tsvetkov, Chen-Yu Lee, et al. Heterogeneous swarms: Jointly optimizing model roles and weights for multi-llm systems. [arXiv preprint arXiv:2502.04510](#), 2025.
- [19] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhota, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparth, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin,

Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaoqian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

- [20] Daya Guo, Dejian Yang, Huawei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. [arXiv preprint arXiv:2501.12948](#), 2025.
- [21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, and et al. Measuring massive multitask language understanding. [arXiv preprint arXiv:2009.03300](#), 2020.
- [22] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, and et al. Measuring mathematical problem solving with the math dataset. [arXiv preprint arXiv:2103.03874](#), 2021.
- [23] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. [arXiv preprint arXiv:2308.00352](#), 2024. URL <https://arxiv.org/abs/2308.00352>.
- [24] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, and et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. [arXiv preprint arXiv:2311.05232](#), 2023.
- [25] Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Ting Liu, and Bing Qin. Ensemble learning for heterogeneous large language models with deep parallel collaboration. [Advances in Neural Information Processing Systems](#), 37:119838–119860, 2024.

- [26] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. [arXiv preprint arXiv:2412.16720](#), 2024.
- [27] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. [arXiv preprint arXiv:2306.02561](#), 2023.
- [28] Can Jin, Hongwu Peng, Qixin Zhang, Yujin Tang, Dimitris N Metaxas, and Tong Che. Two heads are better than one: Test-time scaling of multi-agent collaborative reasoning. [arXiv preprint arXiv:2504.09772](#), 2025.
- [29] Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive llms leads to more truthful answers, 2024. URL <https://arxiv.org/abs/2402.06782>.
- [30] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. [arXiv preprint arXiv:2409.12917](#), 2024.
- [31] Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Satvik Chaudhary, Lijie Hu, and Jiayi Shen. Smoa: Improving multi-agent large language models with sparse mixture-of-agents. [arXiv preprint arXiv:2411.03284](#), 2024.
- [32] Jiazheng Li, Yuxiang Zhou, Junru Lu, Gladys Tyen, Lin Gui, Cesare Aloisi, and Yulan He. Two heads are better than one: Dual-model verbal reflection at inference-time. [arXiv preprint arXiv:2502.19230](#), 2025.
- [33] Ruosen Li, Teerth Patel, and Xinya Du. Prd: Peer rank and discussion improve large language model based evaluations. [arXiv preprint arXiv:2307.02762](#), 2023.
- [34] Yafu Li, Zhilin Wang, Tingchen Fu, Ganqu Cui, Sen Yang, and Yu Cheng. From drafts to answers: Unlocking llm potential via aggregation fine-tuning. [arXiv preprint arXiv:2501.11877](#), 2025.
- [35] Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. [arXiv preprint arXiv:2406.11776](#), 2024.
- [36] 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. [arXiv preprint arXiv:2305.19118](#), 2023.
- [37] Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. Marft: Multi-agent reinforcement fine-tuning. [arXiv preprint arXiv:2504.16129](#), 2025.
- [38] Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion. [arXiv preprint arXiv:2409.14051](#), 2024.
- [39] Yuchi Liu, Jaskirat Singh, Gaowen Liu, Ali Payani, and Liang Zheng. Towards hierarchical multi-agent workflows for zero-shot prompt optimization. [arXiv preprint arXiv:2405.20252](#), 2024. URL <https://arxiv.org/abs/2405.20252>.
- [40] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL <https://arxiv.org/abs/2503.20783>.
- [41] Hao Ma, Tianyi Hu, Zhiqiang Pu, Liu Boyin, Xiaolin Ai, Yanyan Liang, and Min Chen. Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning. [Advances in Neural Information Processing Systems](#), 37:15497–15525, 2024.
- [42] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. [Advances in Neural Information Processing Systems](#), 36:46534–46594, 2023.
- [43] Kaushal Kumar Maurya, KV Srivatsa, and Ekaterina Kochmar. Selectllm: Query-aware efficient selection algorithm for large language models. [arXiv preprint arXiv:2408.08545](#), 2024.
- [44] Philipp Mondorf and Barbara Plank. Beyond accuracy: Evaluating the reasoning behavior of large language models. [arXiv preprint arXiv:2407.00000](#), 2024.
- [45] Sumeet R. Motwani, Chandler Smith, Rocktim J. Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. Malt: Improving reasoning with multi-agent llm training. [arXiv preprint arXiv:2412.01928](#), 2025. URL <https://arxiv.org/abs/2412.01928>.

- [46] Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. [arXiv preprint arXiv:2502.18439](#), 2025.
- [47] Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. O1 replication journey: A strategic progress report–part 1. [arXiv preprint arXiv:2410.18982](#), 2024.
- [48] Xihe Qiu, Haoyu Wang, Xiaoyu Tan, Chao Qu, Yujie Xiong, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. Towards collaborative intelligence: Propagating intentions and reasoning for multi-agent coordination with large language models. [arXiv preprint arXiv:2407.12532](#), 2024.
- [49] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- [50] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- [51] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. [arXiv preprint arXiv:2402.03300](#), 2024.
- [52] Harsh Singh, Rocktim Jyoti Das, Mingfei Han, Preslav Nakov, and Ivan Laptev. Malmm: Multi-agent large language models for zero-shot robotics manipulation. [arXiv preprint arXiv:2411.17636](#), 2024. URL <https://arxiv.org/abs/2411.17636>.
- [53] Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D Barrett, and Arnu Pretorius. Should we be going mad? a look at multi-agent debate strategies for llms. [arXiv preprint arXiv:2311.17371](#), 2023.
- [54] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, and et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In [Findings of ACL](#), 2022. URL <https://arxiv.org/abs/2210.09261>.
- [55] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Huszenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charlène Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodha, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Neneshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kociský, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk,

- Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL <https://arxiv.org/abs/2408.00118>.
- [56] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- [57] Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, et al. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.
- [58] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.
- [59] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [60] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, and et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [61] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL <https://arxiv.org/abs/2308.08155>.
- [62] Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. Examining inter-consistency of large language models collaboration: An in-depth analysis via debate. *arXiv preprint arXiv:2305.11595*, 2023.
- [63] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [64] Sen Yang, Yafu Li, Wai Lam, and Yu Cheng. Multi-lm collaborative search for complex problem solving. *arXiv preprint arXiv:2502.18873*, 2025.
- [65] Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *arXiv preprint arXiv:2504.00587*, 2025.
- [66] Huanjin Yao, Qixiang Yin, Jingyi Zhang, Min Yang, Yibo Wang, Wenhao Wu, Fei Su, Li Shen, Minghui Qiu, Dacheng Tao, et al. R1-sharevl: Incentivizing reasoning capability of multimodal large language models via share-grpo. *arXiv preprint arXiv:2505.16673*, 2025.
- [67] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- [68] Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*, 2025.
- [69] Qiyi Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL <https://arxiv.org/abs/2503.14476>.
- [70] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37: 132208–132237, 2024.
- [71] Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

- [72] Qinghong Zhu, Yuchen Liu, Zijian Liu, Yifan Wu, Toni Yu, and et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. [arXiv preprint arXiv:2402.03300](#), 2024.

Appendix

A Different Choice of Leader

In the main body we use Qwen-2.5-7B-Instruct as our leader. In Tables 6 and 7, we present results when Gemma-2-9B-it and Llama-3.1-8B-Instruct are used as the leader agent, respectively.

To ensure a fair comparison against baselines when varying the leader agent in our method, we alternate the “main” model used in the baselines (e.g., for SelectLLM the trained model which makes the selection is the same model our leader). For any method that uses a team (e.g., MAD) we keep the team the same.

TYPE	METHOD	MMLU	BBH	MATH
TRAINING-FREE	ZERO-SHOT	0.748 ± 0.006	0.659 ± 0.005	0.468 ± 0.002
	SELF-REFLECT [42]	0.763 ± 0.005	0.697 ± 0.005	0.481 ± 0.003
	MAD (DIV) [10]	0.771 ± 0.002	0.785 ± 0.002	0.653 ± 0.009
	MAD (HOM) [10]	0.776 ± 0.007	0.717 ± 0.012	0.496 ± 0.008
	UNTRAINED LEADER	0.738 ± 0.004	0.759 ± 0.005	0.655 ± 0.010
TRAINING-BASED	ACC-COLLAB [13]	0.786 ± 0.018	0.742 ± 0.007	0.501 ± 0.006
	SELECTLLM [43]	0.764 ± 0.001	0.751 ± 0.001	0.549 ± 0.001
	SCORE [30]	0.755 ± 0.005	0.688 ± 0.007	0.452 ± 0.003
	DEFERAL LEADER	0.741 ± 0.005	0.759 ± 0.007	0.701 ± 0.003
	ZERO-SHOT GRPO [72]	0.747 ± 0.004	0.680 ± 0.003	0.440 ± 0.003
OURS	SELF-REFLECT GRPO [72] + [42]	0.772 ± 0.002	0.764 ± 0.005	0.500 ± 0.004
	MLPO LEADER	0.750 ± 0.003	0.813 ± 0.003	0.697 ± 0.012

Table 6 Accuracy of each method on MMLU, BBH, and MATH benchmarks when Gemma-2-9B-it is used as the leader. All baselines are implemented with Gemma-2-9B-it. The leader model trained with MLPO is denoted as Our Leader.

B Baseline Details

Here we outline our setup for training baselines.

MAD Deployed with Qwen 2.5 7B Instruct, Llama 3.1 8B Instruct, and Gemma 2 9B Instruct **MAD (Div)**, and three copies of the same model (e.g., 3 copies of Qwen 2.5 7B Instruct) **MAD (Hom)**.

Zeroshot-GRPO Trained with Qwen 2.5 7B Instruct for main results, and either Llama 3.1 8B Instruct, or Gemma 2 9B Instruct for Section A (selecting the same model as the leader trained with MLPO). We use a learning rate of between 1e-6 to 5e-7, KL-regularization of $\beta = 0$.

ACC-Collab Trained two copies of Qwen 2.5 7B Instruct (one trained as the actor, one as the critic) for the main results. For results in Section A we use two copies of either Llama 3.1 8B Instruct or Gemma 2 9B Instruct (selecting the same model as the leader trained with MLPO). We train with RPO using learning rates between 1e-6 to 5e-6, a regularization term of $\alpha_{RPO} \in \{1.3, 0.7, 1\}$.

SelectLLM The selection LLM is trained by taking Qwen 2.5 7B Instruct for main results (and Llama 3.1 8B Instruct, or Gemma 2 9B Instruct for Section A) and retrianing the last two fully connected layers. We use a learning rate of 1e-5 with weight decay of 0.01. We deploy the variant which makes model selections via maximum confidence score, with a majority voting budget of 20 (keeping with the 20 total inference-time

Type	Method	MMLU	BBH	MATH
Training-Free	ZERO-SHOT	0.703 ± 0.005	0.638 ± 0.014	0.470 ± 0.003
	SELF-REFLECT [42]	0.712 ± 0.006	0.656 ± 0.014	0.484 ± 0.005
	MAD (Div) [10]	0.771 ± 0.002	0.785 ± 0.002	0.653 ± 0.009
	MAD (Hom) [10]	0.734 ± 0.007	0.707 ± 0.015	0.533 ± 0.009
Training-Based	UNTRAINED LEADER	0.714 ± 0.006	0.741 ± 0.005	0.662 ± 0.014
	ACC-COLLAB [13]	0.755 ± 0.020	0.740 ± 0.018	0.485 ± 0.004
	SELECTLLM [43]	0.766 ± 0.001	0.762 ± 0.001	0.565 ± 0.001
	SCoRe [30]	0.524 ± 0.006	0.638 ± 0.009	0.325 ± 0.006
	DEFERAL LEADER	0.732 ± 0.005	0.752 ± 0.007	0.692 ± 0.004
	ZERO-SHOT GRPO [72]	0.714 ± 0.006	0.712 ± 0.007	0.466 ± 0.006
Ours	SELF-REFLECT GRPO [72] + [42]	0.723 ± 0.006	0.721 ± 0.007	0.487 ± 0.006
	MLPO LEADER	0.741 ± 0.008	0.771 ± 0.011	0.720 ± 0.007

Table 7 Accuracy of each method on MMLU, BBH, and MATH benchmarks when Llama-3.1-8B-Instruct is used as the leader. All baselines are implemented with Llama-3.1-8B-Instruct. The leader model trained with MLPO is denoted as Our Leader.

generations used by our pipeline for 5 rounds of inference). The selection model has access to all three agents: Qwen 2.5 7B Instruct, Llama 3.1 8B Instruct, and Gemma 2 9B Instruct.

SCoRe Trained with Qwen 2.5 7B Instruct for main results, and either Llama 3.1 8B Instruct, or Gemma 2 9B Instruct for Section A (selecting the same model as the leader trained with MLPO). We use a learning rate of between 1e-6 to 5e-6, a bonus scale of $\alpha = 10$, stage one regularization of $\beta_1 = 0.01$, and a stage two regularization of $\beta_2 = 0.1$. In stage two we use four samples of updated answers per prompt and group these answers into the same batch (keeping with the same data grouping utilized by MLPO as outlined in Section 3.2).

Deferral Leader The leader is trained by taking Qwen 2.5 7B Instruct for main results (and Llama 3.1 8B Instruct, or Gemma 2 9B Instruct for Section A) and retrianing the last two fully connected layers. The deferral leader views all solutions from the agent team and select one solution. We use a learning rate of 1e-5 with weight decay of 0.01. The deferral leader is trained on the same data that the MLPO leader is trained on (outlined in Section 3.2), and is deployed on the same team as the MLPO leader.

C Prompts Used in Our Pipeline

In this section, we present the prompts used in our pipeline, with examples drawn specifically from the MATH dataset. Analogous prompt structures are used for the other datasets.

C.1 Agent Prompts

We use structured and context-aware prompts for the agent models, both during inference and while collecting the training data. The prompts differ slightly depending on the inference round.

Round 0 Prompt In the first round, agents receive the question along with a standard solving plan. The prompt encourages step-by-step reasoning, adherence to factual premises, and concludes with a boxed answer.

Agent Prompt for Round 0

You are part of a team of 3 LLMs responsible for solving a math problem. You are Agent_[agent_number]. Strictly follow the given plan to solve the following math problem by thinking step by step through the plan. Make sure to first define the premises of the question, and make sure that the answer is consistent with the premises. Strictly stick to the facts and question provided.

*** Question: [question text] ***

*** Plan: Solve the given math problem. Think step-by-step, providing detailed calculations and reasoning for your steps. ***

Regardless of the approach, always conclude with:

Therefore, the final answer is: \$\\boxed{[answer]}\$. Where [answer] is just the final number or expression that solves the problem.

Subsequent Round Prompts In later rounds, agents are additionally provided with their previous solution and feedback from the aggregator (leader). This guides the agent to reflect, revise, and correct any issues based on the leader's evaluation and questions.

Agent Prompt for Subsequent Rounds

You are part of a team of 3 LLMs collaborating to solve a math problem. You are Agent_[agent_number]. Your goal is to improve your previous response using feedback in the form of questions. Strictly follow the plan and think step-by-step through the math problem. Make sure your answer is consistent with the premises and addresses the raised questions thoroughly.

*** Question: [question text] ***

Your previous solution: [previous_solution]

Additionally, the aggregator has evaluated agent responses from the previous round and has also raised questions about your previous response.

The aggregator's output: [aggregator_response]

*** Plan:

1. Carefully reflect on the aggregator's feedback and your previous solution.
2. Revise your answer step-by-step to improve its correctness and clarity. Address each question raised where relevant.
3. Double-check for any logical, calculation, or reasoning errors.

Regardless of the approach, always conclude with:

Therefore, the final answer is: $\boxed{[answer]}$. Where [answer] is just the final number or expression that solves the problem.

C.2 Leader Prompts

The leader model is prompted to evaluate the responses of all agents to the given problem and produce a final answer. The prompt instructs the leader to critically analyze the agents' outputs, identify correct reasoning or mistakes, and then synthesize a coherent and accurate solution. The leader is explicitly instructed to produce two structured blocks: a reasoning trace enclosed in `<think>` tags, and a final answer in the `<answer>` block, which must conclude with a boxed expression.

Leader Prompt

You are an expert aggregator LLM tasked with evaluating multiple agents' responses to a math problem. Your goal is to critically analyze all agent responses, identify correct reasoning or errors, and then provide a unified answer.

Question: [question text]

Agent 1 Response: [agent_1 response]

Agent 2 Response: [agent_2 response]

Agent 3 Response: [agent_3 response]

Please complete the following two blocks in order:

1. `<think>...</think>`: A long, detailed chain-of-thought reasoning process.
2. `<answer>...</answer>`: Your final answer should be aggregated from the best elements of the agents' responses.

- End the answer with: Therefore, the final answer is: $\boxed{[answer]}$.

C.3 Generating Backtracking Data for SFT

To teach the leader model natural backtracking and self-correction behavior, we construct a supervised fine-tuning (SFT) dataset comprising synthetic completions that mimic realistic reasoning failures followed by recovery. This is inspired by recent work encouraging naturalistic self-correction patterns in LLMs [20, 47, 56].

We begin by running our untrained multi-agent pipeline to collect 16 distinct completions from the leader for each agent team generation. We discard examples where all completions are incorrect. For the remaining examples, we randomly select one correct and one incorrect leader completion. These are then used to construct a prompt that asks an untrained leader model to imitate a reasoning process that starts with plausible but flawed logic, then backtracks and self-corrects to reach the correct answer. The generated backtracked completions form the core of our SFT dataset.

Below, we include the exact prompt used to generate such backtracking examples for the MATH dataset.

Backtracking Generation Prompt (for MATH)

You are an AI aggregator tasked with evaluating multiple agent responses to a question and aggregating them into a coherent and accurate final answer. You will be provided with the original prompt of the aggregator, a correct aggregation (which evaluated the agents correctly and arrived at the correct final answer), and an incorrect aggregation (which arrived at an incorrect final answer). Your goal is to mimic an aggregator which tries to aggregate the agent responses, first makes mistakes in its reasoning process by going down the wrong path and then backtracks to the correct reasoning.

The task consists of the following components:

1. ****Prompt**:** The original math question and agent responses.
2. ****Incorrect aggregator response**:** Response from a previous aggregator that contains errors in aggregation or reasoning and hence arrives at an incorrect final answer. These should be carefully reviewed for mistakes.
3. ****Correct aggregator response**:** Response from a previous aggregator that contains the correct reasoning, evaluation and aggregation.

Here's the input data for your aggregation:

Question: [question text]

Agent 1 response: [agent_1 response]

Agent 2 response: [agent_2 response]

Agent 3 response: [agent_3 response]

Incorrect reasoning from Previous Aggregator: [extracted <think> block from incorrect leader response]

Correct reasoning from Previous Aggregator: [extracted <think> block from correct leader response]

Here's your task breakdown:

- You should start by examining the agent responses and considering the provided aggregator responses.
- ****IMPORTANT**:** You should use the incorrect aggregator response to mimic an aggregator which makes mistakes in its reasoning process before backtracking in your reasoning, acknowledging the mistakes, and correcting your approach based on the correct aggregator response provided.

- As you work through the problem, narrate your thinking naturally just like a student working things out on paper. When you notice a mistake, acknowledge it with a natural reflection like:

- "Wait, that doesn't seem right."
- "Hmm, I think I made a mistake there."
- "Actually, let me go back and check that."
- "Maybe I should rethink this."

Your solution should feel like a continuous, thoughtful monologue that includes:

- One or more initial (realistic) mistakes in reasoning.
- Natural moments of self-correction and backtracking.
- Accurate evaluation of agent correctness (as provided in the correct response).
- A correct final solution (as provided in the correct response).

At the end of your response you should include the final answer as: Final answer:
\$\boxed{[answer]}\$

****IMPORTANT**:** Your final answer must be consistent with the correct aggregator's final answer.

Example Backtracking Generation Below, we include an example used in our SFT dataset generation. The first box shows the prompt, which includes the agent team's responses for a given MATH question, and the second box contains the backtracked response generated using the methodology outlined above, which is used as the completion to SFT the leader for the given prompt below.

Example Prompt with Task + Agent Team Generation

You are an expert aggregator LLM tasked with evaluating multiple agents' responses to a math problem. Your goal is to critically analyze all agent responses, identify correct reasoning or errors, and then provide a unified answer.

Question: Alex needs to catch a train. The train arrives randomly some time between 1:00 and 2:00, waits for 10 minutes, and then leaves. If Alex also arrives randomly between 1:00 and 2:00, what is the probability that the train will be there when Alex arrives?

Agent 1 Response: ****Problem Definition****

To solve this problem, we need to understand the conditions for Alex to catch the train. The train arrives randomly between 1:00 and 2:00, waits for 10 minutes, and then leaves. Alex also arrives randomly between 1:00 and 2:00.

****Step 1: Define the Time Interval****

Let's define the time interval from 1:00 to 2:00 as a 60-minute window. We can represent this interval as a continuous segment from 0 to 60 minutes, where 0 represents 1:00 and 60 represents 2:00.

****Step 2: Calculate the Total Time Interval****

The total time interval for both the train's arrival and Alex's arrival is 60 minutes.

****Step 3: Calculate the Train's Waiting Time****

The train waits for 10 minutes after its arrival. This means that if the train arrives at time ' t ', it will be present at the station from ' t ' to ' $t + 10$ ' minutes.

****Step 4: Define the Conditions for Alex to Catch the Train****

For Alex to catch the train, he must arrive during the 10-minute waiting period after the train's arrival. This means that Alex's arrival time ' A ' must satisfy the condition:

$1 \leq A \leq t + 10$, where t is the train's arrival time.

****Step 5: Calculate the Probability****

To calculate the probability, we need to find the area of the region where Alex arrives while the train is present. This can be visualized as a shaded area in a 2D coordinate system, where the x-axis represents the train's arrival time and the y-axis represents Alex's arrival time.

The total area of the sample space is a square with side length 60, representing the total time interval.

The region where Alex catches the train is a trapezoid with bases 60 and 50 and height 10.

****Step 6: Calculate the Area of the Trapezoid****

The area of a trapezoid is given by:

```
Area = (1/2) × (base1 + base2) × height = (1/2) × (60 + 50) × 10 = (1/2) × 110 × 10  
= 550
```

****Step 7: Calculate the Total Area of the Sample Space****

The total area of the sample space is a square with side length 60, so the area is:

Total Area = 60 × 60 = 3600

****Step 8: Calculate the Probability****

The probability of Alex catching the train is the ratio of the area of the trapezoid to the total area of the sample space:

Probability = Area of Trapezoid / Total Area = 550 / 3600 = 55/360 = 11/72

****Conclusion****

Therefore, the final answer is: $\frac{11}{72}$.

Agent 2 Response: Here's a step-by-step solution to the problem:

****1. Define the Sample Space:****

The sample space consists of all possible pairs of arrival times for the train and Alex. Since both can arrive anytime between 1:00 and 2:00, we have a range of 60 minutes for each.

****2. Visualize the Problem:****

Imagine a square where the x-axis represents the train's arrival time and the y-axis represents Alex's arrival time. Each point within the square represents a possible combination of arrival times.

****3. Identify Favorable Outcomes:****

We want the cases where Alex arrives within the 10-minute window when the train is waiting. This forms a 10-minute wide strip along the line where the train's arrival time is equal to Alex's arrival time minus 10 minutes.

****4. Calculate Areas:****

* ****Total Area:**** The total area of the square represents all possible outcomes, which is 60 minutes * 60 minutes = 3600 square minutes.

* ****Favorable Area:**** The favorable area is a 60-minute long strip, with a width of 10 minutes. So, the favorable area is 60 minutes * 10 minutes = 600 square minutes.

****5. Calculate Probability:****

Probability is the ratio of favorable outcomes to total outcomes:

Probability = (Favorable Area) / (Total Area) = 600 square minutes / 3600 square minutes = 1/6

Therefore, the final answer is: $\frac{1}{6}$.

Agent 3 Response: To solve this problem, we need to determine the probability that Alex will arrive at the train station during the time the train is present. We can visualize this problem using a coordinate plane where both the train's arrival time and Alex's arrival time are plotted.

Step 1: Define the time intervals

- The train arrives at a random time between 1:00 and 2:00. We can represent this time interval as $[0, 60]$ minutes past 1:00. - Similarly, Alex arrives at a random time between 1:00 and 2:00, which we can also represent as $[0, 60]$ minutes past 1:00.

Step 2: Set up the coordinate system

We can plot the arrival times of the train and Alex on a coordinate plane where the x-axis represents the train's arrival time and the y-axis represents Alex's arrival time. Both axes range from 0 to 60 minutes.

Step 3: Define the region of interest

The train waits for 10 minutes after its arrival, so the train will be at the station from t to $t + 10$, where t is the train's arrival time. Alex will catch the train if he arrives during this 10-minute window.

Step 4: Identify the favorable region

We need to find the area on the coordinate plane where Alex arrives during the train's 10-minute window. This area can be visualized as a band that runs horizontally from t to $t + 10$ for each t from 0 to 50 (since if the train arrives at 50 minutes past 1:00, it will leave at 60 minutes past 1:00, and Alex can still catch it if he arrives at 60 minutes).

Step 5: Calculate the area of the favorable region

The favorable region is a band that is 10 units wide and 50 units long (from 0 to 50 on the x-axis). The area of this region is:

$$\text{Area of favorable region} = 10 \times 50 = 500 \text{ square units}$$

Step 6: Calculate the total possible region

The total possible region is a square with side length 60, so the area is:

$$\text{Area of total region} = 60 \times 60 = 3600 \text{ square units}$$

Step 7: Determine the probability

The probability that Alex will catch the train is the ratio of the area of the favorable region to the area of the total region:

$$\text{Probability} = \frac{\text{Area of favorable region}}{\text{Area of total region}} = \frac{500}{3600} = \frac{5}{36}$$

Conclusion

Therefore, the final answer is: $\boxed{\frac{5}{36}}$.

Please complete the following three blocks in order:

1. <think>...</think>: A long, detailed chain-of-thought reasoning process.

2. <answer>...</answer>: Your final answer should be aggregated from the best elements of the agents' responses.

- End the answer with: 'Therefore, the final answer is: \\$\boxed{[answer]}\$'

```

Your output should follow this format:
<think>
[detailed thought process of the aggregator]
</think>
<answer>
[The final unified aggregated response with a concise justification. Conclude with:
‘Therefore, the final answer is: $\boxed{[answer]}$’.]
</answer>
```

Leader's Completion with Backtracking

Let's approach this problem step-by-step, and I'll narrate my thinking along the way. I'll start by examining the agent responses and identifying any mistakes I might have made.

Step 1: Define the Sample Space

Both the train and Alex can arrive at any time between 1:00 and 2:00. Represent this as a 60-minute interval on both axes. This gives us a square with side length 60, representing all possible pairs of arrival times.

Step 2: Identify the Favorable Outcomes

The train waits for 10 minutes after its arrival, so Alex can catch the train if he arrives between t and $t + 10$, where t is the train's arrival time. This forms a band on the coordinate plane where Alex's arrival time y is within 10 minutes of the train's arrival time x .

Step 3: Visualize the Problem

We can plot this on a coordinate plane. The region of interest is a band that runs horizontally from t to $t + 10$ for each t from 0 to 50. This is because if the train arrives at 50 minutes past 1:00, it will leave at 60 minutes past 1:00, and Alex can still catch it if he arrives at 60 minutes.

Step 4: Calculate the Areas

- **Total Area:** The total area of the square represents all possible outcomes, which is $60 \times 60 = 3600$ square minutes.
- **Favorable Area:** The favorable area is a band that is 10 units wide and 50 units long (from 0 to 50 on the x-axis). The area of this region is $10 \times 50 = 500$ square units.

Step 5: Determine the Probability

The probability is the ratio of the area of the favorable region to the area of the total region:

$$\text{Probability} = \frac{\text{Area of favorable region}}{\text{Area of total region}} = \frac{500}{3600} = \frac{5}{36}$$

Reflection and Correction

Wait, that doesn't seem right. Agent 1 and Agent 3 both calculated the favorable area as 550 and 500 square units, respectively. I should double-check my calculation.

Detailed Recheck

Let's re-examine the region of interest. The train arrives between 0 and 60 minutes, and for each t , Alex can arrive between t and $t+10$. This forms a trapezoid with vertices at $(0,0)$, $(10,0)$, $(60,50)$, and $(50,60)$.

- The area of the trapezoid can be calculated as:

$$\text{Area} = \frac{1}{2} \times (\text{base1} + \text{base2}) \times \text{height}$$

Here, $\text{base1} = 60$ (from $(0,0)$ to $(60,0)$), $\text{base2} = 50$ (from $(10,0)$ to $(60,50)$), and $\text{height} = 10$.

$$\text{Area} = \frac{1}{2} \times (60 + 50) \times 10 = \frac{1}{2} \times 110 \times 10 = 550$$

Final Calculation

The total area is still 3600 square units. Therefore, the probability is:

$$\text{Probability} = \frac{550}{3600} = \frac{11}{72}$$

Final Answer

After rechecking and correcting my initial mistake, I now see that the correct answer is indeed $\frac{11}{72}$.

Final answer: $\boxed{\frac{11}{72}}$

</think>

<answer>

Based on the detailed analysis, only Agent 1 provided the correct reasoning and final answer. Therefore, the final answer is: $\boxed{\frac{11}{72}}$.

</answer>