POST-TRAINING WITH COMMUNICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how to post-train language models on tasks whose evidence no single model holds. We propose *post-training with communication*, a training paradigm in which identical copies of one model, the agents, each hold a different shard of the context, exchange free-text messages on a shared board, and then answer. No agent has a role, and no agent holds all the evidence, so the team earns full reward only by communicating. Training uses only the verifier of single-model RLVR, with no reward model and no hand-written collaboration reward. To further improve performance, we add two counterfactual credit terms, which the same verifier computes. We find that the trained team scales zero-shot. Trained with only four agents, it keeps most of its accuracy in teams of 32 and remains effective with 64, on inputs several times longer than the model's context window. At these scales, on long-context memory and multi-hop question answering, the team matches or outperforms the same model with a YaRN-extended context window or run as map-reduce. The team agent also stays a generalist: it has no fixed role, and communication training causes no measurable loss of general reasoning.

1 INTRODUCTION

RL post-training, and the multi-agent systems built on it, assume that every trained model sees all the evidence that the answer needs. Whether the reward comes from human preference (RLHF) or from a programmatic checker (RLVR), the model learns the task from the whole context. Many tasks break this assumption: their evidence is split across parties, or it is spread over more text than one model can read reliably. On long-context memory, for example, a model trained by GRPO on the whole input degrades as the input grows past its context window. We are interested in whether one model can instead be trained to solve such tasks as a team of its own copies, in which each copy holds part of the evidence and no copy holds a role.

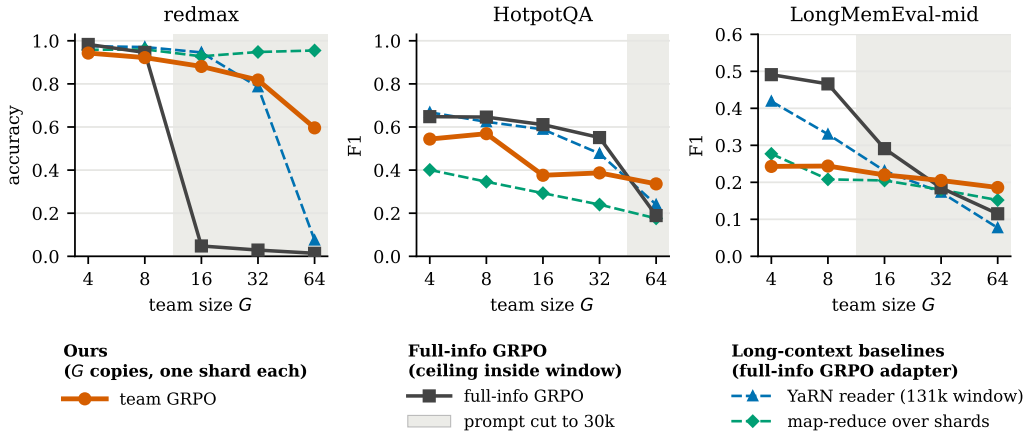

Figure 1: Zero-shot team scaling (eval-only; seed-0 adapters trained at $G = 4$). *Team GRPO*: G agents (copies of one model), one shard each. *Full-info GRPO*: one model reads the whole input; shaded: its prompt is cut to 30k ($G \geq 16$ on redmax and LongMemEval-mid, $G = 64$ on HotpotQA). *Dashed*: the same adapter with a 131k YaRN window and as map-reduce. Values in Table 7.

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

Existing approaches either extend a single reader or train teams on full information. Long-input methods give one reader a longer window, retrieval or memory (Peng et al., 2023; Lewis et al., 2020; Yu et al., 2025), or shard the input under a hand-written protocol (Zhang et al., 2024; Zhou et al., 2024). Unfortunately, a hand-written protocol fixes what the shard-holders share: each holder passes on only what the prompt tells it to, and the task cannot change that. Trained multi-agent LLM systems, in contrast, split a full-information problem into roles, with per-agent policies or one policy conditioned on a role prompt (Motwani et al., 2024b; Park et al., 2025; Hong et al., 2025; Chen et al., 2026; Xu et al., 2026). Because each agent there sees the whole problem, the team gains by dividing work, not by pooling evidence. Few methods train agents to communicate evidence that only they hold, and each differs from our setting: Optima trains offline (Chen et al., 2024), MARSHAL trains in games without free-text messages (Yuan et al., 2025), WideSeek-R1 gives a lead agent its own role over subagents that search (Xu et al., 2026), NeuroMAS gives every agent the full input (Lu et al., 2026), and in PI-Mem only one merge call writes the shared memory (Liu et al., 2026) (§2). None of them trains identical, role-free copies online to pool evidence that is split across them.

We introduce *post-training with communication*, a paradigm that makes the team, rather than one model, the unit of RLVR. The key insight is that the verifier of single-model RLVR scores a team's answers just as it scores one model's, so a team can be trained with no reward model and no hand-written collaboration reward. As Figure 2 shows, identical copies of one model, which we call *agents*, each hold a different shard of the context, post one free-text message to a shared board, and then each answer. No agent has a role, and no agent holds all the evidence, so the team earns full reward only by communicating. Crucially, a verifier is more important here than in single-model RLVR: a learned judge can be persuaded through the channel (§2), and the credit terms below score counterfactual rollouts (a no-board answer for every agent, and sibling boards) at every step, which is cheap only when the reward is a program. The verifier is needed only in training. Frozen agents that communicate at test time beat independent ones only with enough compute and a verifier they can query during the episode (Park et al., 2026), but the trained team runs with no verifier present.

Under team reward alone, however, a message receives little credit. A message changes the other agents' answers, not its sender's, so the gradient that reaches it is weak (§3.2). Frontier systems avoid this setting; Kimi K2.5, for example, freezes its subagents "to circumvent … credit assignment ambiguity and training instability" (Kimi Team, 2026). We therefore add *counterfactual channel credit*, two terms that the same verifier computes: *message credit* pays a message for its effect on the other agents' answers, and *answer credit* pays an answer for its gain from the board.

We evaluate post-training with communication on Qwen3-8B-Base (Yang et al., 2025a) across three tasks whose evidence is split across the agents by construction: numeric aggregation (reduction-max), multi-hop question answering (HotpotQA; Yang et al. 2018) and long-context memory (LongMemEval; Wu et al. 2024). We compare the trained team with the same model trained by GRPO on the whole context and, past the context window, with stronger long-context uses of that model. We find that the trained policy exhibits *zero-shot team scaling*: trained with four agents, it generalizes to teams of up to 64 agents and to inputs several times its context window, with no retraining (§5.1). On long-context memory, the team keeps most of its F1 from 4 to 64 agents as the input grows past one model's window, and from 32 agents on it is at least level with a YaRN-extended reader and map-reduce over the same model. On multi-hop question answering, the single-model baselines stay ahead while the input fits the window, and at 64 agents the team is above the YaRN reader and map-reduce. On numeric aggregation the team keeps most of its reward with up to 32 agents, but a fixed map-reduce, which is exact for a maximum, scores higher. Inside the window, full-info GRPO stays ahead on long-context memory, and an oracle evidence board places the remaining gap in how the agents read the board, not in what they send (§5.1). To our knowledge, no prior RL post-training work runs one trained team across a range of sizes larger than the one it trained with (§2).

In summary, we make four contributions:

- **Post-training with communication**, a training paradigm that trains one role-free policy as a team of its own copies on split context, with the verifier of single-model RLVR as the only reward. Its credit component, counterfactual channel credit, adds a message term and an answer term that the same verifier computes, and costs at most 5% more wall-clock time per step (§3).

Method	Training	Private shard	One policy, no role	Free-text messages	Verifier reward	Per-agent credit	Other team sizes
Long inputs, split across workers							
Chain of Agents (2024)	none	✓	×	✓	–	×	✓
LLM×MapReduce (2024)	none	✓	×	○	–	×	✓
Recursive LMs (2025a)	none, RL	○	×	✓	✓	×	✓
Trained multi-agent systems, full information							
MAPoRL (2025)	RL	×	○	✓	×	○	✓
MAGRPO (2025)	RL	×	×	×	✓	×	×
Stronger-MAS (2025)	RL	×	×	✓	✓	○	○
C3, SHARP, CCPO (2026)	RL	×	×	✓	✓	✓	×
Trained, information asymmetry							
Optima (2024)	offline	✓	○	✓	✓	×	○
MARSHAL (2025)	RL	○	○	×	✓	×	×
Ours	RL	✓	✓	✓	✓	✓	✓

Table 1: Comparison with related methods. *Training*: online RL, offline (SFT or DPO) or none. *Private shard*: no agent holds the whole input. *Other team sizes*: runs at sizes other than the trained one. ○: partial match (for credit: per-agent but not counterfactual); –: no reward.

- **Zero-shot team scaling.** Trained with four agents, the policy runs with up to 64 agents and on inputs past its context window; on long-context memory, from 32 agents on, it is at least level with a YaRN-extended reader and map-reduce over the same model (§5.1).
- **Retention.** Team training causes no measurable loss on MMLU-Pro, GSM8K or HumanEval (§5.2).
- **A credit ablation.** Each credit term matters on a different task: message credit on reduction-max, and answer credit on HotpotQA (§5.3).

2 Related work

Two lines of work are closest to ours: methods that split a long input across several workers, and methods that train teams of LLMs. Table 1 compares the closest methods in each line with ours on seven properties; ours is the only method in the table that has all seven.

Long inputs. There have primarily been two directions for inputs longer than one model can read reliably: 1) extending a single reader, with a longer window, retrieval, memory or long-input RL (Peng et al., 2023; Lewis et al., 2020; Yu et al., 2025; Wan et al., 2025), and 2) splitting the input across several workers. Our setting belongs to the second direction. Existing splitting methods either run a prompted pipeline (Zhang et al., 2024; Zhou et al., 2024), in which each worker shares only what its prompt tells it to, or give one root model the whole input to delegate (Zhang et al., 2025a). The closest trained design, PI-Mem (Liu et al., 2026), trains one policy with GRPO to read every chunk in parallel against a shared memory, but one merge call writes that memory, not the chunk readers. Our setting differs from these works in that no agent holds the whole input, and every shard-holder is the same trained policy, which learns from the task reward what to share.

Trained multi-agent LLM systems. Many methods train LLMs to collaborate on a task that every agent sees in full. They assign roles or seats, with per-role or per-agent policies or role prompts (Motwani et al., 2024b; Park et al., 2025; Liu et al., 2025; Hong et al., 2025; Zhao et al., 2025), so the team gains by dividing work, not by pooling evidence. Closer methods each differ from ours in one respect. Optima (Chen et al., 2024) trains under information asymmetry, but offline; MARSHAL (Yuan et al., 2025) trains one policy online across the seats of two-player games, but with no free-text messages; WideSeek-R1 (Xu et al., 2026) trains a lead agent and its subagents on one shared LLM, but the subagents gather evidence by search and the lead has its own role; and NeuroMAS (Lu et al., 2026) trains role-free agents that exchange text, but with one adapter per agent and the full input for every agent. Our agents, on the other hand, share one policy, as is the MARL default for homogeneous agents (Terry et al., 2020; Yu et al., 2021), differ only in their shard, and train online.

Team size. A shared policy transfers to unseen team sizes when each agent's job stays the same at every size, and fails when the coordination pattern must change with the size (Agarwal et al., 2019; Jiang et al., 2020; Long et al., 2020). Trained LLM teams that communicate in language have rarely been evaluated in this way. MAPoRL (Park et al., 2025) reports one cell past its trained size (trained with two agents, evaluated with three); Stronger-MAS (Zhao et al., 2025) and Optima (Chen et al., 2024) report larger teams without stating whether each size is retrained; WideSeek-R1 (Xu et al., 2026) varies its subagents only up to the ten it trains with; and NeuroMAS (Lu et al., 2026) trains each team size. To our knowledge, none of them runs one RL-trained team across a range of sizes past the one it trained with (Table 1). Our protocol keeps each agent's job the same at every size, the condition under which such transfer is expected.

Credit assignment. Under a shared team reward, agents learn to free-ride on their peers, the lazy-agent problem (Sunehag et al., 2018), which also appears in RL-trained LLM pairs (Zhang et al., 2025b). There are two standard remedies, and each of our credit terms follows one of them. First, critic-free RL uses the mean reward of sibling samples as the baseline (Ahmadian et al., 2024; Shao et al., 2024), and process advantages score a step by its gain in success probability (Setlur et al., 2024); answer credit is such a model-free process advantage. Second, a difference reward (Wolpert & Tumer, 2001) scores an agent by the team reward minus the team reward without its contribution, and COMA (Foerster et al., 2018) turns it into a counterfactual baseline; message credit is a difference reward restricted to the other agents. For LLM teams, concurrent C3 (Chen et al., 2026), SHARP (Li et al., 2026a) and CCPO (Li et al., 2026b) compute per-agent credit from the shared reward. Unlike COMA, message credit needs no learned critic; unlike these LLM-team methods, it scores a message only by its effect on the *other* agents' rewards, because a message cannot change its sender's own answer.

The verifier. We require a program reward because a learned judge can be exploited through the channel (Kenton et al., 2026; Wen et al., 2024; Motwani et al., 2024a).

Information asymmetry. Our tasks are hidden profiles (Stasser & Titus, 1985), in which the evidence that the answer needs is split across the team. Distributed-information benchmarks measure how LLM teams fail to pool such evidence (Zhang et al., 2026; Li et al., 2025); we instead train agents to pool it. Negative results for multi-agent LLMs, such as a single agent outperforming a team at matched compute (Tran & Kiela, 2026), assume shared information, so they do not cover this setting.

3 Post-training with communication

3.1 Setting

Given a policy π_θ and a question q whose context is split into G shards $c_1, \ldots, c_G$, post-training with communication trains π_θ to answer q as a team of G copies of itself, the *agents*, each of which sees only part of the context (Figure 2). The agent in seat $k \in \{1, \ldots, G\}$ (agent k) holds only c_k. Each agent first writes a message $m_k \sim \pi_\theta(\cdot \mid q, c_k)$, and the messages form a shared board $B = (m_1, \ldots, m_G)$. Each agent then answers from its shard and the board, $y_k \sim \pi_\theta(\cdot \mid q, c_k, B)$, and a verifier R scores the answer, $r_k = R(y_k)$. No agent has a role prompt: every seat runs the same weights (parameter sharing; Terry et al. 2020), and training is online GRPO. The team size is a property of the protocol, not of the task instance, so task difficulty (list length, distractor count) can change independently of it.

Team-verifiable reward. Each agent's reward depends on the whole team. The team's score is the mean of r_k over agents, and because agent k answers from its shard *and* the whole board, r_k is a function of the team's joint output, not of agent k's actions alone; we call such a reward *team-verifiable*. A task is *binding* when the evidence that the answer needs is split across the agents, so that no agent holds all of it. Post-training with communication is RL with team-verifiable rewards on binding tasks. An agent alone can still score above zero on a binding task, either by the *shard-luck chance* of holding the answer or through a shortcut (§4 gives them for redmax and HotpotQA).

The full-info ceiling. Inside one model's context window, the team's problem is harder than a single model's. The team must learn two skills, what to send and how to read what the other agents sent, whereas one model that sees everything learns only to answer. *Full-info GRPO* is that model: the same base model, trained by the same GRPO recipe on the whole context, with at least the team's budget of trained generation (§4). While the input fits its window we treat full-info GRPO as the ceiling, not as a competitor, and at the trained size we do not expect the team to reach it. Once the input overflows the window the ceiling no longer holds, and full-info GRPO becomes the baseline that the sharded team must pass (§5.1).

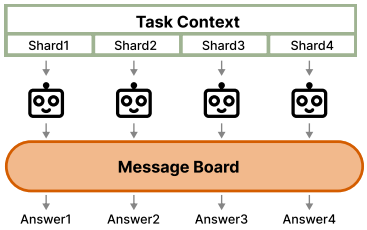

Figure 2: Post-training with communication ($G = 4$). Every agent runs the same policy on one shard, posts one message, and answers from its shard and the board. A verifier (not drawn) scores every answer.

3.2 Counterfactual channel credit

Under team reward, a message receives little credit for its effect on the other agents. GRPO samples a group of N rollouts of each question, indexed by i; the rollouts share the shards and draw different boards. Under vanilla team GRPO, the advantage of agent k normalizes r_k by the mean and standard deviation of agent k's rewards over the N rollouts, and it applies to every token that agent k produced, message and answer alike. This *slot-matched grouping* is the information-asymmetry analogue of per-role grouping: agents that hold different shards are never baselined against each other. This advantage is the right signal for the answer, but not for the message. Agent k's message adds nothing that its shard does not already hold, so it gives agent k no new information for its own answer. Its effect appears instead in the rewards of the other agents, the *peers* of agent k, and no term of agent k's advantage measures it. *Counterfactual channel credit* therefore makes two additions to vanilla team GRPO: one counterfactual term for the answer and one for the message, with weights δ and ω (1 when a term is on and 0 when it is off); §5.3 ablates them.

Answer credit. Answer credit pays an answer for using the board. It adds $\delta \cdot \left(r_k - r_k^{\text{solo}}\right)$ to agent k's advantage, where r_k^{solo} is the reward of agent k when it answers from c_k alone, with no board. Every variant already runs this no-board (*solo*) rollout, so the term requires no extra generation. It is added raw, outside the group normalization, so it stays non-zero even when every rollout in a group ties. The term descends from process advantages (Setlur et al., 2024) and Monte-Carlo step values, with "no board" as the counterfactual baseline.

Message credit. Message credit pays a message for its causal effect on the peers' answers. Let $R_k^{\text{peer}}(B) = \frac{1}{G-1}\sum_{j\neq k} r_j$ be the mean reward of the peers of agent k when they answer from board B, and let $B \setminus m_k$ be the board with m_k blanked. Message credit is the difference reward $\omega \cdot \left(R_k^{\text{peer}}(B) - R_k^{\text{peer}}(B \setminus m_k)\right)$, following the wonderful life utility (WLU; Wolpert & Tumer 2001) and COMA (Foerster et al., 2018); concurrent C3 (Chen et al., 2026) applies a related idea to team reward. Computed exactly, it costs $G-1$ regenerated answers per sender. We instead use a cross-rollout Monte-Carlo proxy over the group's $N \geq 2$ rollouts, which needs no extra generation: the rollouts share the shards, so the other rollouts give the baseline. In rollout i, the message m_k gets $\omega\hat{A}^{\text{peer}}$, with

$$\hat{A}^{\text{peer}} = \frac{R_{k,i}^{\text{peer}} - \frac{1}{N-1}\sum_{i'\neq i} R_{k,i'}^{\text{peer}}}{\sigma_k + \epsilon},$$

where $R_{k,i}^{\text{peer}}$ is the peers' mean reward in rollout i, σ_k is its population standard deviation over the N rollouts, and $\epsilon = 10^{-8}$. The proxy is not an estimate of the blanked-board difference. The baseline boards differ from rollout i's in every message, not only in m_k, so $\hat{A}^{\text{peer}}$ pays m_k when the peers of rollout i do better than the peers of its sibling rollouts, and part of that gain comes from the other messages. Every message-credit cell in the main results uses this proxy.

In sum, agent k's advantage is the vanilla advantage plus answer credit. Its answer receives this advantage, and its message receives it plus $\omega\hat{A}^{\text{peer}}$ (the *add* assembly), so answer credit also reaches the message tokens. The task verifier scores every credit term in the main results, and no other signal is used.

4 EXPERIMENTAL SETUP

Tasks. We design our evaluation around three tasks, one per domain, that are binding by construction: no agent holds all the evidence that the answer needs. Appendix G.1 gives the construction and Table 9 the settings that differ by task.

Reduction-max (redmax). A numeric task: the team reports the maximum of a list of 1600 distinct integers. Each of $G = 4$ agents holds a disjoint slice of 400 numbers. Here, the answer depends on every shard, and the shard-luck chance, that a given agent holds the maximum, is $1/G$.

HotpotQA (Yang et al., 2018). A multi-hop QA benchmark, used in its distractor setting. HotpotQA supplies exactly two gold paragraphs per question, so the team has $G = 2$ agents with one gold paragraph each; these are the only main-table adapters trained at a team size other than $G = 4$. Many questions can be answered from one gold paragraph (Trivedi et al., 2022), so a trained agent alone scores well above zero, and the two-agent team needs less communication than the $G = 4$ tasks.

LongMemEval-mid (Wu et al., 2024). A long-context memory task: the LongMemEval questions whose evidence spans at least two sessions, each kept with its evidence sessions and four distractor sessions. $G = 4$ agents each hold a disjoint slice of the multi-session history, and the evidence sessions go to different agents. Here, the answer needs facts from at least two agents. 212 questions train and 71 evaluate.

Training recipe. Every cell runs one training recipe: Qwen3-8B-Base, LoRA r64/α128, lr 10^{-4}, $N = 4$ rollouts per group with the cross-rollout message credit, 8 questions per step, 50 steps, and vLLM rollouts. We match the budget on training questions (400 per cell), not on epochs: redmax draws fresh instances every step, HotpotQA sees each question about once, and LongMemEval-mid cycles its 212 questions about twice. The credit terms cost almost no wall-clock time: with $N = 4$, the median time per training step of the message-credit, answer-credit and both-credit variants is at most 5% above vanilla on every task (Appendix E). We compare the following variants and baselines.

Team GRPO. The team variants ablate the credit terms of §3: *both credits*, *answer credit* (message credit off), *message credit* (answer credit off) and *vanilla* team GRPO (both terms off). Both credits are the default: unless a variant is named, *team GRPO* is the both-credit team, the team adapter of §5.1–§5.2 (§5.3 gives the reason).

Full-info GRPO. We compare directly to the ceiling, the same model trained by the same recipe on the whole context (on LongMemEval-mid its training prompt is cut at $30{,}000$ tokens, as in the eval), with at least the team's budget of trained generation (messages and answers) in GRPO samples per question (Appendix E). The match does not count the team's untrained no-board answers or prefill.

Untrained floor. The *untrained team*, the team before any RL step, gives the floor (Table 2). It is a prompted protocol, not a bare model: every agent's message instruction already names what to send (Appendix H), so training teaches the agents to carry the protocol out, not what the protocol is. The untrained team is also the prompted form of parallel divide-and-conquer pipelines for long inputs, in which each agent maps its shard to a message and reduces the board to an answer: the parallel variants of LLM$\times$MapReduce (Zhou et al., 2024) and Chain of Agents (Zhang et al., 2024), and a recursive language model (Zhang et al., 2025a) whose root only splits the input and sends sub-calls. Sequential chains and multi-level recursion are outside this set.

Metrics. The training variants and the retention cells run at three seeds; the scaling cells are one seed unless the text says otherwise. Every $\pm$ is the sample standard deviation over seeds. Every table reports the task's own metric: accuracy for redmax (the fraction of agents whose answer is exactly the true value; the reward is 1 for an exact answer and 0 otherwise), and token-overlap F1 averaged over agents for the QA and long-context tasks. Every number comes from an eval of the final adapter on a fixed held-out set, which scores every variant on the same questions as a team; full-info GRPO is scored as one model that sees everything. Appendix G gives the task construction, the eval protocol and the compute.

Team-size sweeps. Every adapter trains at one setting per task, and we ask, eval-only, whether the trained team generalizes beyond it. This is a generalization test, not a scaling law (one seed, one trained setting). Prior work finds that the gain from collaboration against difficulty is an inverted U (Almaatouq et al., 2021; Li et al., 2024; Yang et al., 2025b; Wu et al., 2025), so we report where

the team's score holds and where it falls. The team is the seed-0 team GRPO adapter (both credits) of its task; the sweeps were not repeated on seeds 1 and 2. On redmax this adapter is one of the two seeds that train well (Table 4 gives the three-seed spread), and message credit alone is the more stable redmax variant; we keep one team recipe on every task and did not sweep the message-credit adapter. We run three sweeps:

- *redmax, a fixed slice:* 400 numbers per agent at $G = 4, 8, 16, 32$ and 64 (list lengths 1600 to 25,600), so the team size and the total volume change together and the per-agent load does not.
- *LongMemEval-mid, a growing haystack:* the question's evidence sessions plus G distractor sessions at team size G, one distractor per agent ($G = 4, 8, 16, 32, 64$; at $G = 64$ the distractors come from LongMemEval-m, the larger haystack split, since LongMemEval-s, the smaller one, has too few).
- *HotpotQA, a growing haystack:* each question's 10 paragraphs (2 gold) plus paragraphs from other questions of the same split, $5G$ in total, so every agent holds about 5 paragraphs at $G = 4, 8, 16, 32$ and 64. The two gold paragraphs go to two different agents; the other agents hold only distractors.

The redmax and LongMemEval-mid sweeps use the main-table adapters. For HotpotQA, whose main-table adapters train at $G = 2$, the sweep uses team GRPO (both credits) and full-info GRPO adapters trained in this setting at $G = 4$ (full-info GRPO at the matched 21 samples per question). It scores $n = 200$ questions. The comparison throughout is the full-info GRPO adapter reading the whole input alone, scored as the mean over G independent samples. Past the 32k window its prompt is cut to the first 30,000 tokens, while every agent of a team still holds its whole slice. We therefore also run two stronger uses of the same adapter at every G. The *YaRN reader* applies static YaRN RoPE scaling (Peng et al., 2023) at factor 4 at load time, which extends the native 32,768 positions to a 131k window; the prompt is cut at 130k tokens, so only the redmax and LongMemEval-mid $G = 64$ prompts are cut. *Map-reduce* (Zhou et al., 2024) makes G independent map calls, each with the single-agent prompt over one shard, and then one reduce call over the G map outputs. On redmax the reducer gets the full-info prompt over the mappers' integers, which is exact when every mapper is right; on LongMemEval-mid it reads the question and every map output verbatim, so it can combine counts and sums across sessions. HotpotQA does the same, since a bridge question needs facts from two agents. At HotpotQA $G = 64$ the reduce input (64 notes of up to 512 tokens) can exceed the native window, so that call runs under the YaRN window; no other map-reduce call needs it. Neither baseline is trained for its setting; Appendix G.3 gives their full setup.

5 RESULTS

Figure 1 and Table 7 report our main result, zero-shot team scaling (§5.1): trained with four agents, the team keeps most of its score with up to 32 agents, still works with 64, and runs on inputs past its context window. Table 3 reports retention on general reasoning benchmarks, where team training causes no measurable loss (§5.2), and Table 4 ablates the two credit terms at the trained setting of each task (§5.3): message credit matters on redmax, and answer credit on HotpotQA.

5.1 THE TRAINED TEAM GENERALIZES TO LARGER TEAMS AND INPUTS

Trained with four agents, the team keeps most of its score with up to 32 agents and still works with 64, while full-info GRPO fails once the input overflows its window. Figure 1 grows the team from its trained $G = 4$ to $G = 64$, which takes the input past one model's window (values in Table 7, Appendix C; Appendix D grows the redmax list instead of the team). On redmax, the team goes from 0.943 at $G = 4$ to 0.818 at $G = 32$ and still scores 0.596 at $G = 64$, whereas full-info GRPO, level with the team through $G = 8$, falls to 0.048 at $G = 16$ once its prompt is cut. On LongMemEval-mid, the team goes only from 0.243 to 0.186, while full-info GRPO falls from 0.491 to 0.115.

Notably, on long-context memory the team is at least level with stronger long-context baselines once the input is several times the window (from $G = 32$ on). The baselines are the full-info adapter with a 131k YaRN window and as map-reduce over the shards. At $G = 16$, the truncated full-info GRPO and the YaRN reader are still ahead (0.291 and 0.231 against 0.220); at $G = 32$ the team is level with or above both (0.205 against 0.173 and 0.180; one seed, $n = 71$), and at $G = 64$ it leads (0.186 against 0.077 and 0.152).

Adapter	MMLU-Pro	GSM8K	HumanEval
base	0.252	0.600	0.354
trained on redmax			
team GRPO	$\mathbf{0.369 \pm 0.007}$	$\mathbf{0.725 \pm 0.048}$	0.508 ± 0.077
full-info GRPO	0.349 ± 0.038	0.687 ± 0.040	$\mathbf{0.565 \pm 0.020}$
trained on HotpotQA			
team GRPO	$\mathbf{0.451 \pm 0.031}$	$\mathbf{0.727 \pm 0.042}$	$\mathbf{0.642 \pm 0.025}$
full-info GRPO	0.356 ± 0.052	0.705 ± 0.031	0.528 ± 0.092
trained on LongMemEval-mid			
team GRPO	$\mathbf{0.383 \pm 0.020}$	0.681 ± 0.015	$\mathbf{0.632 \pm 0.019}$
full-info GRPO	0.377 ± 0.025	$\mathbf{0.697 \pm 0.039}$	0.577 ± 0.043

Table 3: Retention, zero-shot accuracy ($n = 500, 500, 164$). A one-adapter change against base has a 95% interval of about ± 0.05 (± 0.10 on HumanEval). Bold: the better trained adapter.

HotpotQA shows the same pattern: full-info GRPO and the YaRN reader stay ahead while the prompt fits the window (through $G = 32$), and at $G = 64$ the team (0.336) is above both the YaRN reader (0.240) and map-reduce (0.176).

On redmax, in contrast, map-reduce is stronger than the team (0.928–0.955 from $G = 16$ to 64): the maximum of the shard maxima is exact, so a fixed pipeline loses nothing. The trained team is thus most useful when the answer needs a few facts located in many shards, not for an associative aggregate; this rests on the largest team sizes of each task, with one seed each.

	board	
Adapter	own	gold evidence
base (untrained team)	0.039	0.054
team GRPO, vanilla	0.194 ± 0.017	0.232 ± 0.028
team GRPO, both credits	0.228 ± 0.011	0.252 ± 0.015

Table 2: Oracle-board control, LongMemEval-mid team F1 ($n = 71$; base: one run).

Inside the window, the team is limited by how its agents answer from the board, not by what they send. At the trained $G = 4$, full-info GRPO stays well ahead on LongMemEval-mid (0.491 against 0.243; Table 7). To locate this gap, an oracle-board control (Table 2) replaces the board with the gold evidence and regenerates only the answers. On their own boards, the trained teams are far above the untrained team (0.194 and 0.228 against 0.039; Appendix I shows their messages). With the gold evidence on the board, however, they gain only 0.02–0.04. If the messages were the bottleneck, a perfect board would close most of the gap; the remaining gap therefore lies in how the agents answer from the board: mostly multi-session (60%) and knowledge-update (35%) questions, which combine facts from different agents into one value, a step that neither credit term targets (Appendix B).

5.2 Collaboration does not measurably reduce general reasoning

We find that team training does not measurably reduce general reasoning. Table 3 scores the base model and each task's team and full-info adapters zero-shot on MMLU-Pro, GSM8K and HumanEval, one benchmark per main group of the Qwen3 report (Appendix F). Our plain prompt makes the absolute numbers incomparable to that report, so we compare each adapter with the base model.

Every adapter, at every seed, scores above the base model on every benchmark. We read this as retention rather than improvement: most of the gain is answer format (the base model answers in the required form on 0.69 of MMLU-Pro and 0.74 of HumanEval, the adapters on 0.82–0.98 and 0.90–0.99), and the table rules out only a loss larger than about 0.05 (0.10 on HumanEval). Team training also costs no more than single-agent RL: the team adapter is at least level with the full-info adapter on every benchmark except HumanEval after redmax training (0.508 against 0.565, inside the seed spread).

5.3 MESSAGE CREDIT MATTERS ON REDMAX, ANSWER CREDIT ON HOTPOTQA

We find that message credit matters on redmax and answer credit on HotpotQA, and that no single term is best on every task. Table 4 ablates counterfactual channel credit (§3) at the trained setting of each task, removing one or both terms from the both-credit team.

Adapter	redmax accuracy	HotpotQA F1	LongMemEval-mid F1
both credits	0.779 ± 0.274	0.650 ± 0.029	$\mathbf{0.228 \pm 0.011}$
answer credit	0.286 ± 0.025	$\mathbf{0.654 \pm 0.037}$	0.214 ± 0.008
message credit	$\mathbf{0.827 \pm 0.021}$	0.538 ± 0.023	0.215 ± 0.020
none (vanilla)	0.626 ± 0.080	0.554 ± 0.055	0.194 ± 0.017

Table 4: Ablating the team credit ($n = 100$, 200, 71). Each row names the credit terms kept; *vanilla* keeps none. Bold: the best variant per task.

On redmax, message credit is necessary: three of the four agents can answer correctly only from the holder's message, so most of the team's reward depends on messages. Removing message credit from the both-credit team drops redmax from 0.779 to 0.286, far below vanilla (0.626); removing answer credit instead lifts it to 0.827 (± 0.021), the best and most stable variant. Both effects hold on each of the three seeds. The both-credit team varies widely (± 0.274): one seed of three ends far below the other two.

On HotpotQA, in contrast, answer credit is necessary. Removing it drops HotpotQA from 0.650 to 0.538 F1, level with vanilla (0.554), on each of the three seeds; removing message credit leaves it level (0.654). A trained agent alone already answers many HotpotQA questions from one gold paragraph (§4), so the signal that matters is whether an answer uses the board, which answer credit measures.

On LongMemEval-mid, the two terms add up in the mean (0.228, against 0.214 and 0.215 with one term and 0.194 with none), within the seed spread; §5.1 shows that the main limit there is answering from the board, which neither term targets.

No single term is best on every task. We use the both-credit team in §5.1–§5.2 because it is close to the best variant on each task.

We also explored three approximations of message credit, each of which replaces one piece of its counterfactual (Appendix A). None of them improves on message credit in both cost and score: on redmax they are level with it, and on HotpotQA two have a lower mean (by 0.03–0.04, within seed noise) and the third, information gain, collapses.

6 CONCLUSION

We introduced post-training with communication, a training paradigm that trains one role-free policy, as a team of its own copies, to answer from evidence that no copy holds alone, with the task's verifier as the only reward. Trained with four agents, the team keeps most of its score at team sizes and input lengths it never saw; past the window it is at least level with YaRN and map-reduce on long-context memory and multi-hop QA, while a fixed map-reduce is better on an associative aggregate. Team training shows no measurable loss of general reasoning, and two verifier-scored credit terms each help on a different task. More broadly, we hope that training a team of one model on split context can complement longer context windows on tasks whose evidence no single model holds.

Limitations and future work. Every result uses one model, Qwen3-8B-Base. The clearest weak spot is answering from the board on LongMemEval-mid: questions that combine facts from different agents into one value need a training signal that neither credit term gives. We have no rule yet that predicts which credit term a task needs. The scaling cells have one seed (seed 0, not a seed mean), and on redmax we did not sweep the more stable message-credit adapter; all tasks use one message round; on HotpotQA a gold paragraph comes first in the shard that holds it; the baselines past the window do not include a retrieval or memory agent (Yu et al., 2025); compute matching excludes the team's untrained no-board answers and prefill; and retention rules out only a loss larger than about 0.05 (0.10 on HumanEval).

AI USE STATEMENT

In this work, we used generative AI tools to implement methods. We have not used generative AI tools to formulate mathematical claims or to write proofs, and translation and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to search and summarize related literature, to create figures and to edit the paper for readability. We reviewed all AI-assisted work: we reviewed the AI-assisted code and ran it to check it, checked every reported number against the run logs, checked literature summaries and citations against the original sources, and reviewed all AI-assisted text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains language models on a synthetic numeric task and on public benchmarks (HotpotQA, LongMemEval, MMLU-Pro, GSM8K, HumanEval). It does not involve human subjects or collect new personal data. We do not anticipate significant ethical concerns from the methods or findings.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 and Appendix G give the method, the training recipe, task construction (Appendix G.1), evaluation protocol (Appendix G.2) and compute (Appendix G.4), and Appendix H gives all prompts. Appendix E gives the compute matching between the team and the single-agent baselines. We will release code and trained adapters upon publication.

REFERENCES

Akshat Agarwal, Sumit Kumar, and Katia Sycara. Learning Transferable Cooperative Behavior in Multi-Agent Teams. arXiv:1906.01202, 2019.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. arXiv:2402.14740, 2024.

Abdullah Almaatouq, Mohammed Alsobay, Ming Yin, and Duncan J. Watts. Task complexity moderates group synergy. *Proceedings of the National Academy of Sciences*, 118(36):e2101062118, 2021. doi: 10.1073/pnas.2101062118.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System. arXiv:2410.08115, 2024.

Yanjun Chen, Yirong Sun, Hanlin Wang, Jinghan Wang, Xinming Zhang, Xiaoyu Shen, Wenjie Li, and Wei Zhang. Exact Is Easier: Credit Assignment for Cooperative LLM Agents. arXiv:2603.06859, 2026.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. arXiv:1705.08926, 2018.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-Agent Deep Research: Training Multi-Agent Systems with M-GRPO. arXiv:2511.13288, 2025.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph Convolutional Reinforcement Learning. arXiv:1810.09202, 2020. ICLR 2020.

Zachary Kenton, Lili Janzer, Rory Greig, Tian Huey Teh, Kirill Tyshchuk, Jonah Brown-Cohen, Harri Edwards, Senthooran Rajamanoharan, Noah Y. Siegel, Natasha Jaques, and Rohin Shah. Debate Training Reduces Reward Hacking in RLAIF. arXiv:2608.17776, 2026.

Kimi Team. Kimi K2.5: Visual Agentic Intelligence. arXiv:2602.02276, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401, 2020. NeurIPS 2020.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More Agents Is All You Need. arXiv:2402.05120, 2024. TMLR.

Yanming Li, Xuelin Zhang, WenJie Lu, Ziye Tang, Maodong Wu, Haotian Luo, Tongtong Wu, Zijie Peng, Hongze Mi, Yibo Feng, Naiqiang Tan, Chao Huang, Lian Peng, and Li Shen. Who Deserves the Reward? SHARP: Shapley Credit-based Optimization for Multi-Agent System. arXiv:2602.08335, 2026a.

Yuxuan Li, Aoi Naito, and Hirokazu Shirado. Systematic Failures in Collective Reasoning under Distributed Information in Multi-Agent LLMs. arXiv:2505.11556, 2025. ICML 2026.

Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. Counterfactual Credit Policy Optimization for Multi-Agent Collaboration. arXiv:2603.21563, 2026b.

Dawei Liu, Haixu Song, Shuang Cheng, Shijie Wang, Haozheng Hou, Kaifeng Liu, Ermo Hua, Zhonghang Yuan, Zhijie Zhong, Yuchen Fan, Biqing Qi, and Bowen Zhou. PI-Mem: Pushing Long-Context Reasoning to 3.6M Tokens with Parallel-Iterative Memory. arXiv:2608.03048, 2026.

Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM Collaboration with Multi-Agent Reinforcement Learning. arXiv:2508.04652, 2025. AAAI 2026.

Qian Long, Zihan Zhou, Abhibav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. arXiv:2003.10423, 2020. ICLR 2020.

Haoran Lu, Luyang Fang, Wenxuan Zhong, and Ping Ma. NeuroMAS: Multi-Agent Systems as Neural Networks with Joint Reinforcement Learning. arXiv:2605.16757, 2026.

Sumeet Ramesh Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H. S. Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. arXiv:2402.07510, 2024a.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti 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:2412.01928, 2024b. COLM 2025.

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:2502.18439, 2025. ACL 2025.

Jongho Park, Vasilis Kontonis, Shivam Garg, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Scaling Discovery through Test-Time Communication. arXiv:2609.21032, 2026.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models. arXiv:2309.00071, 2023.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. arXiv:2410.08146, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300, 2024.

Garold Stasser and William Titus. Pooling of Unshared Information in Group Decision Making: Biased Information Sampling During Discussion. *Journal of Personality and Social Psychology*, 48(6):1467–1478, 1985. doi: 10.1037/0022-3514.48.6.1467.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:1706.05296, 2018.

J. K. Terry, Nathaniel Grammel, Sanghyun Son, Benjamin Black, and Aakriti Agrawal. Revisiting Parameter Sharing in Multi-Agent Deep Reinforcement Learning. arXiv:2005.13625, 2020.

Dat Tran and Douwe Kiela. Single-Agent LLMs Outperform Multi-Agent Systems on Multi-Hop Reasoning Under Equal Thinking Token Budgets. arXiv:2604.02460, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. arXiv:2108.00573, 2022. TACL 2022.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. QwenLong-L1: Towards Long-Context Large Reasoning Models with Reinforcement Learning. arXiv:2505.17667, 2025. Technical Report.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. Language Models Learn to Mislead Humans via RLHF. arXiv:2409.12822, 2024.

David H. Wolpert and Kagan Tumer. Optimal Payoff Functions for Members of Collectives. *Advances in Complex Systems*, 4(2–3):265–279, 2001. doi: 10.1142/S0219525901000188.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. arXiv:2410.10813, 2024. ICLR 2025.

Haolun Wu, Zhenkun Li, and Lingyao Li. Can LLM Agents Really Debate? A Controlled Study of Multi-Agent Debate in Logical Reasoning. arXiv:2511.07784, 2025.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring Width Scaling for Broad Information Seeking via Multi-Agent Reinforcement Learning. arXiv:2602.04634, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. arXiv:2505.09388, 2025a.

Yongjin Yang, Euiin Yi, Jongwoo Ko, Kimin Lee, Zhijing Jin, and Se-Young Yun. Revisiting Multi-Agent Debate as Test-Time Scaling: A Systematic Study of Conditional Effectiveness. arXiv:2505.22960, 2025b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. arXiv:1809.09600, 2018. EMNLP 2018.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. arXiv:2103.01955, 2021. NeurIPS 2022 Datasets and Benchmarks.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping Long-Context LLM with Multi-Conv RL-based Memory Agent. arXiv:2507.02259, 2025. ICLR 2026.

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, Xiao-Ping Zhang, Chao Yu, and Yu Wang. MARSHAL: Incentivizing Multi-Agent Reasoning via Self-Play with Strategic LLMs. arXiv:2510.15414, 2025.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive Language Models. arXiv:2512.24601, 2025a.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arık. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks. arXiv:2406.02818, 2024. NeurIPS 2024.

Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and Wenyuan Jiang. Silo-Bench: A Scalable Environment for Evaluating Distributed Coordination in Multi-Agent LLM Systems. arXiv:2603.01045, 2026. ACL 2026.

Zhiwei Zhang, Xiaomin Li, Yudi Lin, Hui Liu, Ramraj Chandradevan, Linlin Wu, Minhua Lin, Fali Wang, Xianfeng Tang, Qi He, and Suhang Wang. Unlocking the Power of Multi-Agent LLM for Reasoning: From Lazy Agents to Deliberation. arXiv:2511.02303, 2025b.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-MAS: Multi-Agent Reinforcement Learning for Collaborative LLMs. arXiv:2510.11062, 2025. ICLR 2026.

Zihan Zhou, Chong Li, Xinyi Chen, Shuo Wang, Yu Chao, Zhili Li, Haoyu Wang, Rongqiao An, Qi Shi, Zhixing Tan, Xu Han, Xiaodong Shi, Zhiyuan Liu, and Maosong Sun. LLM×MapReduce: Simplified Long-Sequence Processing using Large Language Models. arXiv:2410.09342, 2024.

A Approximations of message credit

Message credit (§3) targets one quantity, the effect of a message on the peers' answers. Besides the cross-rollout proxy of the main results, it has a *counterfactual mode* that measures this effect from three pieces: a *reader* (the peers, regenerated), a *scorer* (the verifier), and a *counterfactual* (the realized message against one alternative message sampled from the policy, the peers regenerated for both). Each term in this appendix replaces exactly one piece (Table 5); reading the table down, supervision rises as generation falls. We explored these terms as alternatives that need fewer regenerated peer answers per sender; none of them is part of the method, because none improves on message credit in both cost and score. We report each against message credit under one shared recipe, so the question is always the same: does the cheaper term recover what message credit buys, and what does it cost? Table 6 answers it for all four terms on the same cells.

Term	Replaces	With	Cost
message credit, counterfactual mode (§3)	—	—	1 sampled message and $2(G-1)$ regenerated peer finals per covered sender; verifier only
self-counterfactual (§A.1)	the reader	the sender's own shardless self	2 finals per seat; verifier only; assumes one shared policy
info-gain message credit (§A.2)	the scorer	teacher-forced gold log-probability of the peers' answers	no generation; needs the gold answer; rewards dictating the answer
supervised sender credit (§A.3)	the counterfactual	token recall of the seat's own gold evidence in its message	no generation; needs a per-seat evidence annotation; says nothing about use

Table 5: One target, three approximations. Every row credits the sender's message; only message credit is scored by the verifier on the real readers.

Head-to-head. We train each term in place of message credit on redmax and HotpotQA under one recipe: Qwen3-8B-Base, 50 steps, answer credit off, message credit in its counterfactual mode (two messages, the realized one and one sampled alternative, with one peer regeneration each, for one covered sender per rollout), every other term at coefficient 1, seeds $\{0, 1, 2\}$ (Table 6). The counterfactual mode and a 32-list redmax eval set differ from the recipe of Table 4, so the message-credit cells here are a separate run, not its message-credit row. On HotpotQA the sender term scores token recall of the supporting sentences inside the seat's own paragraphs (the dataset's supporting-fact annotation), and the self-counterfactual term deletes the seat's paragraphs. On redmax every term matches message credit: the three means lie within one seed standard deviation of it, and the supervised sender term is the most stable variant (0.885 ± 0.012, a spread eight times smaller). On HotpotQA, where the final still has to compose two facts, message credit is best (0.600): the self-counterfactual and sender terms trail it by 0.03–0.04, within noise, and info-gain collapses. So no cheaper term beats message credit where the answer needs composition, and the one that needs

neither gold nor an annotation, the self-counterfactual term, stays close to it on both tasks at two finals per seat.

Task (metric)	message credit	self-counterfactual	info-gain	sender credit
redmax (accuracy)	0.839 ± 0.094	0.875 ± 0.064	0.823 ± 0.112	0.885 ± 0.012
HotpotQA (F1)	0.600 ± 0.038	0.571 ± 0.017	0.164 ± 0.272	0.561 ± 0.036

Table 6: Message credit and its three approximations head to head (answer credit off; $n = 32$, 200; three seeds).

A.1 Replacing the reader: self-counterfactual sender credit

Under one shared policy, a peer that lacks seat k's shard is the same model as seat k without its shard. The self-counterfactual term uses that twin as the reader: seat k answers from the realized board on a *shardless* view of its own prompt (the private shard deleted), once with its own message on the board and once with it blanked, both finals sharing one seed, and the credit is the difference in k's own verifier reward. It targets the same quantity as message credit at two finals per seat, every seat every step, against $2(G-1)$ regenerated peer finals per covered sender in message credit's counterfactual mode, and it needs no gold, only a task hook that strips the shard. What it gives up is the reader's own evidence: a real peer answers with its shard in hand, the shardless twin has none, so on a task where the reader's shard matters it is a stricter test of the message than message credit is.

Under the paper recipe (Table 6) the term alone, with message credit off, reaches 0.875 ± 0.064 on redmax against 0.839 ± 0.094 for message credit, and 0.571 ± 0.017 F1 on HotpotQA against 0.600 ± 0.038: it recovers what message credit buys on both tasks, within seed noise.

A.2 Replacing the scorer: gold-anchored information gain

Message credit (§3) scores a message by the *verifier*'s reward on the counterfactual board. An alternative scores it by information gain: the summed teacher-forced log-probability of the peers' gold answers *with* minus *without* the message (run under the counterfactual mode that message credit uses, at a matched budget). At three seeds the result is task-dependent and does not warrant replacing the verifier signal (Table 6). On the team task metric (the team accuracy or F1 that the rest of the paper reports), info-gain message credit gives no clear gain on redmax (0.823 against 0.839 for message credit) and *destabilizes* HotpotQA (0.164 ± 0.272 against 0.600 ± 0.038 F1; the large spread is training collapse on two of three seeds). The one place info-gain message credit leads is the *raw trained reward* on redmax, the training reward rather than the eval metric of Table 6 (0.919 ± 0.027 against 0.846 ± 0.084), but the gold-anchored uptake objective directly shapes that reward, so the lead is partly mechanical and does not carry through to the team's answer reward. A gold-anchored uptake reward is maximized by a message that simply dictates the answer, which shows up as a higher raw reward when a shard genuinely holds the answer (redmax) and as training collapse when the final still needs multi-hop composition (HotpotQA). We therefore keep the verifier-scored message credit as the reported message-credit term (matched counterfactual budget, final-step values, sample sd over three seeds).

A.3 Replacing the counterfactual: supervised sender credit

This term does not measure what a message did to a reader; it names what the message should contain.

Message credit and answer credit score a message only through the finals it produces, so the sender learns what to say only once some reader has turned a message into a correct answer. We add one direct sender term: a seat's message earns a bonus equal to the fraction of the tokens of its own *supporting evidence* that appear in the message, where the evidence is read from the *training* split's per-seat annotation (on LongMemEval-mid, the turns of the seat's evidence session that carry the answer). The term is a supervised routing signal, unlike message credit and answer credit, which see only the final answers, and we report it as such throughout. It names *what* a seat should surface and says nothing about how to phrase it or how the peers should use it; a seat that holds no evidence earns nothing. On LongMemEval-mid we use the *replace* assembly: with message credit on, a message receives only its credit terms (message credit and the sender bonus) instead of the *add* assembly's seat

advantage plus credit (§3); with message credit off, it receives the vanilla advantage plus the sender bonus. This removes an empty-message collapse we observed under the add assembly on a pilot task (a same-code both-credit control went to 100% empty messages at step 34; the per-token mean loss pushes a short message harder than a long one). Everything else is the canonical 50-step recipe.

redmax and HotpotQA. In the head-to-head of Table 6 the term replaces message credit ("add" assembly, answer credit off). On redmax the evidence is the seat's own slice maximum, credited on an exact match; on HotpotQA it is the supporting sentences in the seat's paragraphs, credited by token recall. The term is the fastest to learn on redmax (0.63–0.77 accuracy at step 10 against 0.13–0.46 for message credit) and ends at 0.885 ± 0.012, level with message credit (0.839 ± 0.094) and far more stable. On HotpotQA it ends at 0.561 ± 0.036 F1 against 0.600 ± 0.038: stating one's own supporting sentence does not by itself teach the team to compose the two facts.

LongMemEval-mid. The oracle-board control of §5.1 and the message audit of Appendix B place the main limit in the reader. The sender credit gives the sender a direct signal: a seat that holds an evidence session is paid for the fraction of its flagged evidence turns that its message states, before any reader can use them. Under the canonical 50-step recipe it fires from the first step (the messages' evidence coverage is 0.12–0.16 at step 1 and 0.18–0.23 over the last ten steps). Both credits + sender credit reaches 0.241 ± 0.032 mean F1 over three seeds, against 0.228 ± 0.011 for the plain both-credit team (Table 4; the two also differ in the assembly, replace against add): a paired gain of $+0.013$ that is not significant at three seeds ($+0.02$, -0.03, $+0.05$). Vanilla + sender credit reaches 0.213 ± 0.006, a paired $+0.019$ over the vanilla team ($+0.03$, $+0.02$, $+0.00$), level with the single-credit variants of Table 4. So the supervised term puts evidence on the board from step one, but the readers convert little of it, as the oracle board of §5.1 predicts: even the gold evidence adds only 0.02–0.04.

B LongMemEval-mid message audit

This section gives the message audit behind the oracle-board control of §5.1. The credit terms move some answer content onto the board: all of the gold answer's content words appear somewhere on the board in 39–42% of the questions for the message-credit, answer-credit and both-credit teams, against 34% for vanilla, 23% for the untrained team and 10% from the question alone.

C Team-size sweep

Table 7 gives the values of Figure 1.

D Scaling the list length

A second redmax sweep, eval-only like those of §5.1, keeps the team at $G = 4$ and grows the list from 1600 to 12.8k numbers, so each seat's slice grows from 400 to 3200 numbers (Table 8). The full-info GRPO reader is scored as the mean over four samples.

The team holds across list length; full-info GRPO falls at the window. The team scores 0.943 and 0.935 at 1600 and 3200 numbers and 0.877 at 6400 and 12.8k, although it trained at 1600 only. Up to 3200 numbers the full-info GRPO reader is level with it or above (0.958–0.983). From 6400 numbers on, the reader's prompt is cut and its mean falls to 0.055 and 0.045: the team keeps 0.877 at a volume where full-info GRPO answers almost nothing.

E Compute budget

The full-info agent gets at least the team's budget of trained generation: $N_{\text{full}} = \text{round}\big(G \cdot N \cdot (T \cdot \text{msg} + \text{final})/\text{final}\big)$ GRPO samples per question, where T is the number of message rounds and msg and final are the token caps of a message and an answer. This is 18 samples for redmax, 11 for HotpotQA and 24 for LongMemEval-mid. The match does not count the no-board answer that each agent also generates for answer credit (at most $G \cdot N$ further answers per question, never trained on), and prefill is not equalized.

	team deployment	full-info GRPO, alone		map-reduce
Setting	team GRPO	30k	YaRN 131k	full-info GRPO
redmax (accuracy), 400 numbers per seat				
$G = 4$ (trained)	0.943	**0.983**	0.975	0.958
$G = 8$	0.922	0.946	**0.971**	0.960
$G = 16$	0.881	0.048†	**0.946**	0.928
$G = 32$	0.818	0.029†	0.786	**0.948**
$G = 64$	0.596	0.014†	0.075†	**0.955**
HotpotQA (F1), about 5 paragraphs per seat				
$G = 4$ (trained)	0.544	0.648	**0.668**	0.401
$G = 8$	0.569	**0.646**	0.624	0.346
$G = 16$	0.376	**0.611**	0.589	0.293
$G = 32$	0.387	**0.551**	0.477	0.240
$G = 64$	**0.336**	0.189†	0.240	0.176
LongMemEval-mid (F1), one distractor session per seat				
$G = 4$ (trained)	0.243	**0.491**	0.419	0.277
$G = 8$	0.244	**0.466**	0.330	0.208
$G = 16$	0.220	**0.291**†	0.231	0.205
$G = 32$	**0.205**	0.185†	0.173	0.180
$G = 64^\ddagger$	**0.186**	0.115†	0.077†	0.152

Table 7: Values of Figure 1: eval-only, seed-0 adapters trained at $G = 4$ ($n = 100$ redmax lists, 200 HotpotQA questions, 71 LongMemEval-mid questions; the HotpotQA adapters are trained at $G = 4$); baselines as in Appendix G.3. †Prompt truncated at the reader's window. ‡Distractors from LongMemEval-m (LongMemEval-s has at most 63). The LongMemEval-mid sweep gives each seat one distractor, so its $G = 4$ row is not the eval of Table 4, which assigns distractors at random. Bold: the best cell per setting.

	team deployment	full-info deployment
List length	team GRPO	full-info GRPO
1600 (trained)	0.943	**0.983**
3200	0.935	**0.958**
6400	**0.877**	0.055†
12.8k	**0.877**	0.045†

Table 8: Redmax list length at $G = 4$, eval-only accuracy (seed 0, $n = 100$). †Prompt truncated at the 30k window.

With $N = 4$ rollouts per group the credit terms cost almost no wall-clock time: the median time per training step of the message-credit, answer-credit and both-credit variants is 111–114 s on redmax, 44–49 s on HotpotQA and 222–224 s on LongMemEval-mid, at most 5% above vanilla on every task (on HotpotQA two of them are faster than vanilla).

F Retention benchmarks

The Qwen3 base-model report (Yang et al., 2025a) groups its benchmarks by general knowledge, math and coding, and then adds multilingual; we take one benchmark from each of its three main groups. We use a fixed stratified 500-question slice of MMLU-Pro, 500 questions of GSM8K, and all 164 HumanEval problems. Our own harness scores every model with the same plain zero-shot think/answer prompt; HumanEval runs the unit tests for pass@1.

G Implementation details

G.1 Task construction

Every seat's prompt has the same scaffold: the seat's shard, the question, and an instruction to reason inside `<think>` tags and to answer inside `<answer>` tags. The message phase and the answer

phase each append a short instruction (Appendix H gives the message instructions). We use the base model, so the prompt is plain text with no chat template. The parser takes the last `<answer>` span, and it also accepts a span that the model does not close. The full-info prompt has the same scaffold, the whole context and no team framing.

redmax. Each instance draws 1600 distinct integers uniformly from $[10000, 99999]$, without replacement, so the maximum is unique. The list is cut into G contiguous slices of equal length (400 numbers at $G = 4$), one slice per seat. The reward is 1 when the parsed answer equals the maximum and 0 otherwise. Training draws 8 fresh instances at every step from the run seed. The eval instances come from generator seed 1; for the seed-0 adapters, 8 of the 100 eval lists are the lists of training step 1, seen once before any update. Seeds 1 and 2 do not overlap.

HotpotQA. We use the distractor setting: each question comes with 10 paragraphs, 2 gold and 8 distractors. The two gold paragraphs go to different seats, and each distractor goes to a seat chosen at random, with a fixed seed. A paragraph keeps all its sentences. Each seat's gold paragraph comes first in its shard. The full-info variant trains with all 10 paragraphs in one shard, so its prompt has both gold paragraphs first (§6). Training takes questions in order from the start of the `train` split, 400 questions in 50 steps. Evaluation uses the first 200 questions of the `validation` split. The reward is token-overlap F1 after SQuAD normalization (lower case, no punctuation, no articles); for a yes/no gold answer it is 1 when the answer contains the gold token and not its opposite, and 0 otherwise.

LongMemEval-mid. We start from the `s` variant of LongMemEval (Wu et al., 2024). We keep the question types whose evidence can span sessions (multi-session, temporal reasoning, knowledge update), and of these only the questions whose evidence spans at least two sessions. This gives 283 binding questions. A fixed split by question (seed 0) holds out 25% of them: 212 questions for training and 71 for evaluation. Each question keeps its evidence sessions plus four distractor sessions, chosen at random from its haystack. The evidence sessions go to different seats in turn, and each distractor session goes to a seat chosen at random. Each seat keeps its sessions in haystack order. The reward is the token-overlap F1 of HotpotQA.

The four distractor sessions set the *mid* setting between two failure modes. With only the evidence sessions the routing is trivial, and under an early message-credit variant the fraction of empty messages rises from 2% to ∼70%. With the full `s` haystack each seat's ∼30k-token slice, with the prompt and the answer budget, overflows the model's 32k window. With four distractors each seat's slice stays under the window *and* the message channel stays productive.

	redmax	HotpotQA	LongMemEval-mid
team size G	4	2	4
message / answer token cap	64 / 512	512 / 1536	128 / 256
KL coefficient to the base model	0.02	0.04	0.02
full-info rollouts per group N_{full}	18	11	24
eval set	100 instances	200 questions	71 questions

Table 9: The settings that differ by task. All other settings are the recipe of §4.

Common to all tasks: AdamW at a constant learning rate of 10^{-4}, gradient norm clipped at 1.0, PPO ratio clip 0.2, and a per-token KL penalty to the base model (the adapter disabled). LoRA adapts the attention projections (q, k, v, o) with no dropout. The loss of a trajectory is the mean over its tokens. Each run uses the adapter of its final step, with no checkpoint selection.

G.2 Evaluation protocol

The eval decodes by sampling at temperature 1.0 and top-p 1.0, one sample per seat, with a seed fixed per question. The seeds do not depend on the adapter, so every variant and every training seed sees the same questions under the same sampling seeds. The eval loads each final adapter and scores it on one fixed set (Table 9): a team adapter as a team (the team protocol of training), and a full-info adapter as one agent with the whole context (one sample; the mean over G samples in the scaling evals, Appendix G.3). A cell reports the mean over seats of the verifier score: exact match for redmax, token-overlap F1 for HotpotQA and LongMemEval-mid. In the LongMemEval-mid eval and in the scaling evals of §5.1, every prompt except the YaRN reader's (Appendix G.3) is cut to its first 30,000

tokens; at these lengths only the full-info prompt can reach that limit. Every $\pm$ is the sample standard deviation over the three training seeds.

G.3 Long-context baselines

The team-size sweep of §5.1 compares the team with three uses of one model: the seed-0 full-info GRPO adapter of the task, trained at $G = 4$ (on HotpotQA, in the padded setting). All three are eval-only and use the questions, the sampling (temperature 1.0, top-p 1.0) and the answer budget of the team cell at the same G. The team and every baseline see the same $n = 100$ redmax lists, 200 padded HotpotQA questions and 71 LongMemEval-mid haystacks.

Truncated reader. The adapter reads the whole input alone with the native window. We cut the prompt to its first 30,000 tokens, which leaves room for the answer. The cell is the mean over G independent samples, so it is matched in samples to a team of G seats.

YaRN reader. The same reader with static YaRN RoPE scaling (Peng et al., 2023) at factor 4 over the native 32,768 positions, which gives a 131,072-token window, the extension that the Qwen3 model card documents. We set it at load time in vLLM (RoPE type `yarn`, original window 32,768, the model's own RoPE base kept). The adapter never trained with YaRN. We cut the prompt to its first 130,000 tokens. The full-info prompt is about 38k, 77k and 154k tokens on redmax and 44k, 83k and 161k on LongMemEval-mid at $G = 16$, 32 and 64, so only $G = 64$ is cut; the HotpotQA prompt (at most 43k) is never cut. Static YaRN changes the positions at every length, so inside the 30k window ($G \leq 8$) the YaRN and truncated readers see the same prompt and differ only in the RoPE scaling. The cell is the mean over G samples, as for the truncated reader.

Map-reduce. A fixed two-step pipeline with the same adapter at both steps, in the style of LLM×MapReduce (Zhou et al., 2024). *Map:* G independent calls, one per shard. Each call gets the single-agent prompt over that shard alone (the full-info prompt of redmax over the shard's numbers, or the question and the shard's sessions for LongMemEval-mid), so the adapter maps in the format it trained on. No call sees a message or another shard. *Reduce:* one call over the G map outputs. On redmax the reducer gets the full-info prompt over the integers that the mappers answered; a map output without a parseable integer drops out. This is the standard map-reduce for an associative operation, and it is exact when every mapper is right. On LongMemEval-mid a bare per-shard answer cannot carry a count or a sum across sessions, so the reducer gets the question and every map output verbatim (the reasoning and the answer), each labelled with its reader. HotpotQA does the same, since a bridge question needs facts from two shards. We draw G independent reduce samples over the same map outputs and report their mean. No call sees more than one shard or the G map outputs, so the native 32,768-token window is sufficient, except for the HotpotQA $G = 64$ reduce call, which runs under the YaRN window.

The three baselines are strong uses of a trained model, not new training: none of them trains the adapter at the new G or for the new pipeline, just as the team is not trained past $G = 4$. We do not run retrieval or memory baselines (§6).

G.4 Compute

Each training run and each eval uses one NVIDIA H200 or B300 GPU, 8 CPU cores and 96 GB of host memory. vLLM generates the rollouts in bfloat16 with the current LoRA adapter; the policy gradient and the KL term are computed with the PyTorch model in bfloat16. The vLLM context limit is 8192 tokens for HotpotQA training and 32,768 tokens for redmax, LongMemEval-mid and every team-size sweep (131,072 for the YaRN reader).

H Message instructions

Every seat gets the same message instruction after its shard, so the untrained team already runs a prompted protocol that names what to send. redmax:

```
State ONLY the single largest number in your section, then
stop.
Your message:
```

HotpotQA and LongMemEval-mid:

```
You have only PART of the evidence.  Share the specific
facts from your information that could help answer the
question (names, dates, relations).  Be concise; do not
guess the final answer yet.
```

I COMMUNICATION EXAMPLES

Two held-out questions on which the trained team (the seed-0 adapter in every seat) answers from facts that no single seat holds. We quote the messages verbatim but cut them to the lines that carry facts ([. . .] marks a cut). The full messages also hold text that continues the prompt, such as invented dialogue turns, which the readers must read past.

HotpotQA, two seats. *"When was the British author who wrote the novel on which 'Here We Go Round the Mulberry Bush' was based born?"* Gold: 7 January 1936. Seat 0 holds the paragraph on the film, which names the author; seat 1 holds the paragraph on the author, which gives the birth date.

> **Seat 0:** The British author who wrote the novel "Here We Go Round the Mulberry Bush" is Hunter Davies. [. . .] In the information provided, there is no specific mention of Hunter Davies' birth date.
>
> **Seat 1:** Hunter Davies: Edward Hunter Davies, OBE (born 7 January 1936) is a British author, journalist and broadcaster. [. . .]

Seat 0 resolves the first hop and states that its shard lacks the second; seat 1 sends the second. Both seats answer *7 January 1936*. On the same question the untrained team's seat 1 sends the birth date of an unrelated actor, and neither untrained seat gives a scored answer.

LongMemEval-mid, four seats. *"How much have I made from selling eggs this month?"* Gold: $120. The two facts sit in two different evidence sessions of the chat history: the number of eggs sold and the price per dozen.

> **Seat 0:** 1. 40 dozen eggs sold this month. [. . .]
>
> **Seat 1:** I've been selling eggs to my neighbor for $3 a dozen. [. . .]

Seats 2 and 3 send no fact about eggs. Seats 0 and 2 answer *$120*, seat 3 *120 USD*, and seat 1 *$40 dozen x $3 a dozen = $120* (per-seat F1 for seats 0–3: 1, 0.29, 1 and 0.67; the token F1 scores correct answers in another form below 1). Seat 2 holds neither fact and answers from the board alone. All four seats of the untrained team score 0; their finals drift into unrelated text.