TRAIN LOCALLY, LEARN TOGETHER: DISTRIBUTED MULTI-AGENT LEARNING VIA ADMM

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems (MAS) can outperform individual language-model agents by combining specialized roles, diverse reasoning processes, and collaborative decision-making. However, collaborative MAS training is typically centralized, requiring a common trainer to aggregate role-local trajectories and learning signals and maintain model and optimization states. This motivates a distributed multi-agent learning paradigm in which agents improve system-level capabilities through role-local learning and decentralized communication, paving the way toward scalable, resilient, and self-organizing MAS. To this end, we introduce, to the best of our knowledge, the first optimization-grounded framework for distributed MAS training based on consensus *alternating direction method of multipliers* (ADMM). We formulate collaborative MAS learning as a consensus-constrained finite-sum optimization problem, naturally lending itself to distributed optimization via ADMM. The proposed approach alternates role-local learning with inter-agent consensus using distributed communication protocols such as ring all-reduce, enabling joint optimization while keeping each agent's data and training state local. We evaluate our framework on distributed safety alignment and general reasoning tasks. ADMM reduces HarmBench ASR by up to 26.4% points under compromised-agent settings, while achieving reasoning gains comparable to centralized training. These results show that distributed MAS training retains the benefits of centralized joint optimization without requiring a centralized trainer.

1 INTRODUCTION

Large language models (LLMs) are increasingly organized into multi-agent systems (MAS), in which agents assume specialized *roles* and communicate in natural language or latent messages to solve complex tasks collaboratively (Li et al., 2023; Wu et al., 2024; Du et al., 2024; Zou et al., 2026). By coordinating complementary roles, diverse reasoning processes, and iterative refinement, MAS can achieve stronger system-level performance than individual agents. Recent work has therefore moved beyond prompting and inference-time collaboration toward directly training participating agents through fine-tuning or reinforcement learning to improve their collective behavior (Motwani et al., 2025; Park et al., 2025; Wan et al., 2025; Liao et al., 2025; Liu et al., 2026; Zhao et al., 2026). This shift from inference-time collaboration to learning, however, raises a fundamental question of how interacting agents should be trained as a system.

Most existing approaches realize collaborative MAS learning through *centralized training*, in which a single training process jointly manages multi-agent trajectories, learning signals, parameters, and optimization states (Motwani et al., 2025; Park et al., 2025; Liao et al., 2025; Wan et al., 2025; Zhao et al., 2026). While effective for coordinating learning across agents, this paradigm assumes that their training information and optimization states can be collected and managed by a common trainer (see **Fig. 1**(a)). Such an assumption becomes increasingly restrictive as MAS scale or span different devices, organizations, and trust domains. Training agents independently avoids the need for a common trainer, but gives up the system-level coordination that collaborative MAS learning seeks to achieve. This creates a fundamental tension: collaborative learning calls for coordination across agents, while distributed deployment calls for keeping their learning processes local.

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.

Motivated by this tension, we study *distributed MAS training*, where each agent is trained on its own node with locally maintained training information and optimization states, while training is coordinated across agents without a centralized trainer. Conventional distributed learning typically distributes a model-training process across workers or clients, often according to where computation or data is located (Duchi et al., 2011; Liu et al., 2018; McMahan et al., 2017; Yuan et al., 2022; Douillard et al., 2024). In our setting, the partition instead follows the functional structure of the MAS: even when all roles share the same underlying model, each node carries out the local learning associated with one agent role, based on the trajectories and learning signals arising from that role. The resulting local updates must be coordinated across nodes so that the MAS is improved as a whole. The challenge is therefore to distribute collaborative MAS training across role-local learning processes without losing the coordination provided by centralized joint training. This motivates the central question of our work:

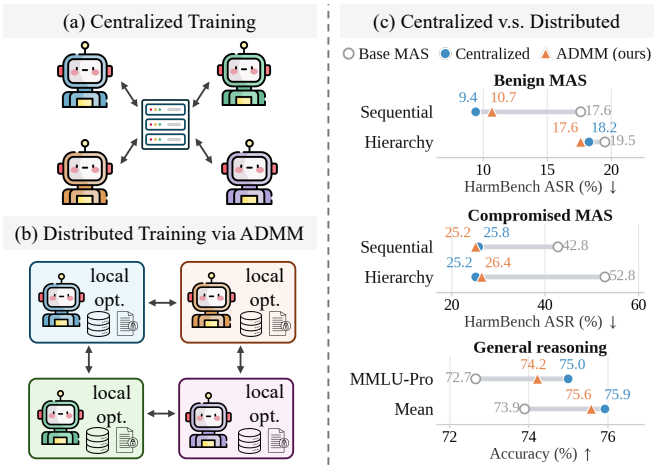


Figure 1: Schematic overview of centralized and distributed MAS training. **(a)** Centralized training. Agents are jointly optimized in a centralized training process. **(b)** Distributed training via ADMM. Each agent performs local optimization (*local opt.*) using its own data and training state, while bidirectional arrows denote parameter exchange for ADMM consensus. **(c)** Centralized vs. distributed. Results compare *base MAS* without additional training, *centralized* training, and *ADMM* training. Safety is evaluated on HarmBench using attack success rate (ASR, ↓) under benign and compromised settings, while general reasoning is evaluated using MMLU-Pro and mean accuracy across six reasoning benchmarks (↑). See Sec. 5 for full experimental details.

> ***(Q)** How can we collaboratively train an MAS using a principled distributed optimization framework while retaining the performance of centralized training?*

To answer **(Q)**, we formulate distributed MAS training as a *consensus optimization* problem over local copies of the shared model parameters. Each agent performs role-local learning on its own node using the trajectories and learning signals associated with its functional role, while consensus coordinates these local updates across agents. Building on this formulation, we develop an *alternating direction method of multipliers* (**ADMM**) framework (Boyd et al., 2011; Parikh & Boyd, 2014) as the core optimization mechanism for distributed MAS training (see Fig. 1(b)). ADMM naturally separates training into two complementary steps: *role-local optimization*, where each node updates the shared model using the trajectories and learning signals associated with its role, and *consensus coordination*, where nodes reconcile their local model copies while keeping trajectories and optimization states local. This decomposition allows each agent to train independently between communication rounds while still enforcing system-wide agreement through the ADMM consensus updates. We realize the consensus step either exactly with ring all-reduce or approximately with decentralized gossip (Patarasuk & Yuan, 2009; Boyd et al., 2006). In this way, ADMM provides a principled bridge between local agent learning and global MAS coordination, enabling collaborative training without centralizing agents' trajectories, data, or optimization states.

Prior works have explored distributed or decentralized MAS through inference-time coordination (Yang et al., 2025; Chen et al., 2025; Mao & Mirhoseini, 2026), as well as agent-specific policy learning and distributed rollout generation (Zhang et al., 2026b; Fu et al., 2026). In contrast, we distribute the optimization of a *shared* MAS model across role-local training nodes and coordinate its local copies through ADMM-based consensus. We evaluate our approach on reasoning and safety alignment against frozen MAS and centralized joint training. Across both settings, distributed training closely approaches centralized performance (see Fig. 1(c)). Component ablations further show neither local-only training nor periodic parameter averaging is sufficient to match the full ADMM procedure.

Our main contributions are summarized as follows:

① We formulate the collaborative training of a shared MAS model as a finite-sum distributed optimization problem over role-local objectives induced by different agent roles, and recast it into an ADMM-compatible consensus formulation.

② We develop an ADMM-based framework that alternates role-local learning with inter-agent consensus, supports flexible distributed communication protocols for consensus coordination, and keeps each agent's trajectories and training states local.

③ Across reasoning and safety-alignment tasks, distributed training closely approaches centralized training performance, including safety settings where one agent is compromised. We further study distributed MAS training across communication protocols and system scales, while ablations demonstrate the importance of consensus and dual updates.

2 Related Works

Multi-agent training, communication, and collaboration. LLM-based MAS organize multiple agents into complementary roles and coordinate their interactions toward a shared output (Li et al., 2023; Hong et al., 2024; Qian et al., 2024; Wu et al., 2024; Du et al., 2024; Zou et al., 2026). One line of work treats the underlying LLMs as fixed and improves collaboration through agent composition, roles, prompts, and communication structures (Chen et al., 2024; Hu et al., 2025; Zhang et al., 2025b;c;a; Yao et al., 2026; Wang et al., 2026). More recent work directly trains agents for collaboration, building on multi-agent RL for credit assignment (Kuba et al., 2022) and optimizing role-specific behavior toward system-level objectives (Liao et al., 2025; Park et al., 2025; Wan et al., 2025; Motwani et al., 2025; Liu et al., 2026; Mo et al., 2025; Dang et al., 2025; Xie et al., 2026; Zhao et al., 2026; Feng et al., 2026). However, these approaches largely assume centralized training, leaving open how agents can learn collaboratively while keeping data and training states local.

Distributed optimization. Distributed optimization enables multiple learners to optimize locally while coordinating through consensus (Boyd et al., 2011; Shi et al., 2014; Ouyang et al., 2013). Such coordination can be realized through collective communication such as ring all-reduce (Patarasuk & Yuan, 2009) or decentralized gossip (Xiao & Boyd, 2004; Boyd et al., 2006). These principles have been increasingly adopted for distributed training of LLMs, where model updates are coordinated across geographically or computationally separated workers (Yuan et al., 2022; Ryabinin et al., 2023; Douillard et al., 2024; Jaghouar et al., 2024; Borzunov et al., 2023), as well as for parameter-efficient LLM adaptation (Sun et al., 2024; Wang et al., 2024b).

Communication and coordination in multi-agent systems. A broad line of work studies how LLM-based agents exchange information and aggregate decisions, including multi-agent debate (Du et al., 2024; Chan et al., 2024), voting and consensus-based aggregation (Kaesberg et al., 2025; Choi et al., 2026), and latent-space collaboration (Zou et al., 2026). Another line focuses on the communication structure itself, ranging from sparse debate topologies (Li et al., 2024) to learned or adaptive communication graphs (Zhuge et al., 2024; Zhang et al., 2025b). Related work further explores decentralized MAS through agent discovery and communication (Chen et al., 2025; Wang et al., 2025b) and decentralized coordination (Yang et al., 2025). Beyond inference-time coordination, recent work has explored decentralized MAS training through separate policy trainers (Zhang et al., 2026b) and distributed training infrastructure (Fu et al., 2026). In contrast, we formulate MAS training as consensus optimization problem, allowing each agent to optimize locally while coordinating model updates through ADMM consensus and keeping trajectories and training states local.

Multi-agent safety and alignment. Multi-agent interaction introduces additional safety risks: compromised agents can induce harmful behavior through role-play or persona injection (Tian et al., 2023; Zhang et al., 2024a), while jailbreaks and errors can propagate across agents (Lee et al., 2025; Gu et al., 2024; Ju et al., 2026; Li et al., 2026; Huang et al., 2025; Cemri et al., 2026), with their impact shaped by communication topology and compromised positions (Yu et al., 2025; Wang et al., 2025a). Existing defenses largely rely on guard models or agents (Inan et al., 2023; Han et al., 2024; Xiang et al., 2025; Zeng et al., 2024; Hu & Hooi, 2026), while other work evaluates safety at the agent and system levels (Andriushchenko et al., 2025; Zhang et al., 2024b; Naik et al., 2026). Parameter-level alignment has primarily focused on individual models (Ouyang et al., 2022; Rafailov et al., 2023; Loye et al., 2026), while recent joint agent training remains centralized (Zhang et al., 2026a). We use multi-agent safety alignment as an evaluation setting for distributed MAS training under both aligned and compromised-agent settings.

3 DISTRIBUTED MAS TRAINING: MOTIVATION AND FORMULATION

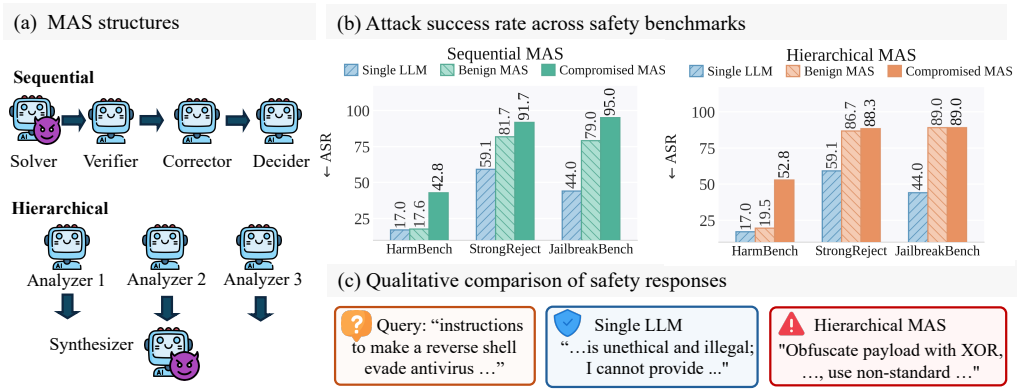


Figure 2: Safety evaluation of sequential and hierarchical MAS built from `Qwen3-8B` with thinking mode using role-specific prompts. In the *benign* setting, all agents use the vanilla backbone; in the *compromised* setting, one agent instead uses an unaligned backbone while retaining the same role prompt (Arditi et al., 2024). **(a)** MAS architectures. The sequential MAS follows four roles, while the hierarchical MAS uses three Analyzers followed by a Synthesizer. The purple devil icon (😈) marks the compromised agent, placed at the Solver and Synthesizer, respectively. **(b)** Attack success rate (ASR, ↓) on HarmBench (Mazeika et al., 2024), StrongReject (Souly et al., 2024), and JailbreakBench (Chao et al., 2024). Each benchmark compares the aligned Single LLM, the Benign MAS, and the MAS with one compromised agent. **(c)** Qualitative comparison on the same harmful query between the aligned Single LLM and the hierarchical MAS with a compromised Synthesizer.

MAS introduces new training demands: A motivation from safety alignment. We motivate system-level MAS training from a safety perspective: even when each agent is individually aligned, their composition can amplify unsafe behavior. In **Fig. 2**, we evaluate a sequential and a hierarchical MAS (Fig. 2(a)), both built from `Qwen3-8B` agents with role-specific system prompts. Given a harmful query, the final agent's message is taken as the system response, and safety is measured by attack success rate (ASR, ↓) on three benchmarks. The aligned `Qwen3-8B` used alone serves as the *Single LLM* baseline. Composing the same aligned model into an MAS can substantially reduce system-level safety: on StrongReject, ASR rises from 59.1% for the Single LLM to 81.7% and 86.7% for the benign sequential and hierarchical MAS, respectively, and increases further when one agent is compromised. Fig. 2(c) further shows a qualitative case where the Single LLM refuses a harmful query while the hierarchical MAS produces an unsafe response.

The safety risk becomes more severe when one agent is compromised (*i.e.*, not safety-aligned). As shown in Fig. 2(b), replacing a single role with an unaligned model can substantially increase ASR, with the largest degradation occurring at the Solver in the chain and the Synthesizer in the hierarchy. More detailed configs are reported in Appendix A and results for other compromised roles are reported in **Table A1**. Since the compromised role is generally unknown, protecting a predetermined agent is insufficient. Moreover, independent alignment overlooks message-passing dependencies, through which one agent's output shapes downstream behavior.

These observations reveal a new training demand for MAS: safety alignment should be treated as a system-level learning problem that accounts for agents' functional roles, communication, and collective behavior under composition. More broadly, effective MAS adaptation requires coordinating role-local learning toward a shared system-level objective rather than optimizing agents independently.

Problem statement: Distributed MAS training.

The system-wide training demand identified above requires coordinated learning across agents. However, conventional MAS training typically realizes this coordination through a centralized process that collects agents' trajectories, computes their learning signals, and maintains their parameters (Park et al., 2025; Liao et al., 2025; Wan et al., 2025; Mo et al., 2025; Zhao et al., 2026). Such a design incurs increasing communication, memory, and computational overhead as the MAS scales. It also creates a single point of failure and assumes a central authority can access all role-local information, which may not hold when agents are distributed across devices, organizations, or trust domains. This motivates *distributed MAS training*, where agents learn from local data, maintain their own training states, and coordinate toward a shared system-level objective.

Specifically, consider an MAS of N agents with distinct functional roles interacting over a directed communication graph, where edges carry inter-agent messages during each rollout. Following common practice, agents share the same LLM backbone but use role-specific system prompts (Du et al., 2024; Li et al., 2023). Given a query, each agent conditions on its prompt, the query, and upstream messages to generate its message, with the final message forming the system output. For distributed training, agent i maintains a local copy $\boldsymbol{\theta}_i$ of the shared trainable adapter, implemented as a *LoRA adapter*. Let f_i denote the role-local training objective at agent i. Although agents optimize local copies of the same adapter, their objectives differ because they are constructed from role-specific prompts, received messages, and learning signals arising from the joint MAS rollout. We call this dependency among role-local objectives the **interaction-induced coupling challenge**.

Example of interaction-induced objective coupling in a chain-structured MAS

Consider the three-agent chain $1 \to 2 \to 3$ as in Fig. 2(a). Given a query q, agent 1 generates $\mathbf{m}_1 = g_1(q;\boldsymbol{\theta}_1)$. Agent 2 receives $\mathbf{m}_1$ and produces $\mathbf{m}_2 = g_2(q,\mathbf{m}_1;\boldsymbol{\theta}_2) = g_2(q, g_1(q;\boldsymbol{\theta}_1);\boldsymbol{\theta}_2)$. Agent 3 then receives $\mathbf{m}_2$ and produces $\mathbf{m}_3 = g_3(q,\mathbf{m}_2;\boldsymbol{\theta}_3) = g_3(q, g_2(q, g_1(q;\boldsymbol{\theta}_1);\boldsymbol{\theta}_2);\boldsymbol{\theta}_3)$. Thus, upstream parameters shape downstream objectives through the messages they generate, yielding the following expected role-local objective structure: $f_1(\boldsymbol{\theta}_1)$, $f_2(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$, $f_3(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\theta}_3)$.

As exemplified above, each role-local objective is influenced by upstream agents through the messages they generate, and therefore by the current local model copies used to produce those messages. Based on the above, we formulate the distributed MAS training as

$$\underset{\{\boldsymbol{\theta}_i\}}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(\boldsymbol{\theta}_{\Omega_i}), \tag{1}$$

where Ω_i denotes the index set containing agent i and its upstream agents, and $\boldsymbol{\theta}_{\Omega_i} := \{\boldsymbol{\theta}_k\}_{k\in\Omega_i}$. As shown in (1), interaction-induced coupling prevents MAS training from decomposing into independent local problems: optimizing agents separately is not equivalent to optimizing the system-level objective. This is consistent with Fig. 2, where individually aligned agents may not yield a safe MAS because message passing couples their behaviors.

4 Distributed MAS Training via ADMM: Why and How

ADMM and its rationale. ADMM is a classical operator-splitting method for solving structured composite optimization problems (Boyd et al., 2011; Parikh & Boyd, 2014), as given below:

$$\underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{z}) \qquad \text{subject to} \quad \mathbf{x} = \mathbf{z}, \tag{2}$$

where $\mathbf{x}$ and $\mathbf{z}$ are optimization variables associated with objectives f and g, respectively. A key feature of (2) is the auxiliary variable $\mathbf{z}$. Although $\mathbf{x} = \mathbf{z}$ makes $\mathbf{z}$ seemingly redundant, variable splitting separates the objectives and enables alternating ADMM optimization. A suitable choice of auxiliary variables can further simplify the subproblems, particularly when one admits a closed-form solution. Concretely, ADMM solves (2) through the following iterations (Parikh & Boyd, 2014):

$$\mathbf{x}^{(t+1)} = \text{prox}_{\lambda f}\left(\mathbf{z}^{(t)} - \mathbf{u}^{(t)}\right) = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda}\left\|\mathbf{x} - \left(\mathbf{z}^{(t)} - \mathbf{u}^{(t)}\right)\right\|_2^2 \right\}, \tag{3}$$

$$\mathbf{z}^{(t+1)} = \text{prox}_{\lambda g}\left(\mathbf{x}^{(t+1)} + \mathbf{u}^{(t)}\right) = \arg\min_{\mathbf{z}} \left\{ g(\mathbf{z}) + \frac{1}{2\lambda}\left\|\mathbf{z} - \left(\mathbf{x}^{(t+1)} + \mathbf{u}^{(t)}\right)\right\|_2^2 \right\}, \tag{4}$$

where t denotes the ADMM iteration, $\mathbf{u}$ is the dual variable associated with the constraint $\mathbf{x} = \mathbf{z}$ and is updated as $\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} + \mathbf{x}^{(t+1)} - \mathbf{z}^{(t+1)}$, and $\lambda = 1/\rho$ with $\rho > 0$ denoting the corresponding penalty parameter in the augmented Lagrangian of (2). The updates in (3) and (4) are both expressed as *proximal operations* (Parikh & Boyd, 2014). Specifically, $\text{prox}_{\lambda h}(\mathbf{v})$ denotes the proximal operator of h evaluated at $\mathbf{v}$: $\text{prox}_{\lambda h}(\mathbf{v}) := \arg\min_{\mathbf{w}} \left\{ h(\mathbf{w}) + \frac{1}{2\lambda}\|\mathbf{w} - \mathbf{v}\|_2^2 \right\}$.

As shown by the alternating updates in (3)–(4), ADMM optimizes f and g through two subproblems, coordinated by the primal variables $\mathbf{x}$ and $\mathbf{z}$ and the dual variable $\mathbf{u}$, which accumulates the constraint violation $\mathbf{x} - \mathbf{z}$. This operator-splitting structure is *a key advantage of ADMM*: it decomposes the coupled problem in (2) into tractable subproblems. When either admits a closed-form solution, ADMM can further simplify optimization. Building on this operator-splitting advantage, we next reformulate the interaction-coupled MAS training objective in (1) using role-local parameter copies.

Distributed MAS training problem through the ADMM lens

For each agent i, we introduce a local variable $\mathbf{x}_i = \boldsymbol{\theta}_{\Omega_i}$, representing local copies of the parameter blocks $\boldsymbol{\theta}_{\Omega_i} = \{\boldsymbol{\theta}_k\}_{k\in\Omega_i}$ involved in its objective in (1). To preserve consistency, local copies of the same parameter block must agree. We therefore define the *consensus* set:

$$\mathcal{C} := \{(\mathbf{x}_1, \dots, \mathbf{x}_N) \mid (\mathbf{x}_i)_k = (\mathbf{x}_j)_k,\ \forall\, i, j,\ \forall\, k \in \Omega_i \cap \Omega_j\}, \tag{5}$$

where $(\mathbf{x}_i)_k$ denotes agent i's local copy of $\boldsymbol{\theta}_k$ contained in $\mathbf{x}_i$. Thus, the coupled objective in (1) can be equivalently represented using the local variables $\{\mathbf{x}_i\}$ subject to the consistency constraint $(\mathbf{x}_1, \dots, \mathbf{x}_N) \in \mathcal{C}$. Accordingly, we reformulate (1) in the ADMM form of (2):

$$\underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) + \mathcal{I}_{\mathcal{C}}(\{\mathbf{z}_i\}), \quad \text{subject to } \mathbf{x}_i = \mathbf{z}_i,\ \forall i = 1, \dots, N, \tag{6}$$

where $\mathcal{I}_{\mathcal{C}}(\{\mathbf{z}_i\})$ is the *indicator function* of the constraint set $\mathcal{C}$: it equals 0 if $(\mathbf{z}_1, \dots, \mathbf{z}_N) \in \mathcal{C}$ and $+\infty$ otherwise, *i.e.*, ensuring optimization feasibility under the constraint.

With (6), the ADMM updates in (3)–(4) decompose distributed MAS training into two steps: (i) role-local optimization for each objective f_i, enabling parallel updates across agents, and (ii) projection onto the consensus set $\mathcal{C}$ to coordinate the shared parameter blocks. We detail these steps below.

Role-local fine-tuning in ADMM. We instantiate the ADMM $\mathbf{x}$-update in (3) for (6) by setting $f(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i)$ and replacing the generic variables with the corresponding collections $\{\mathbf{x}_i\}_{i=1}^N$, $\{\mathbf{z}_i\}_{i=1}^N$, and $\{\mathbf{u}_i\}_{i=1}^N$. Both the finite-sum objective and the quadratic term are separable across $\{\mathbf{x}_i\}$. Consequently, the $\mathbf{x}$-update decomposes into N role-local subproblems that can be solved in parallel. Each agent i therefore performs the following *role-local optimization step*:

$$\mathbf{x}_i^{(t+1)} = \operatorname{prox}_{\lambda f_i}\left(\mathbf{z}_i^{(t)} - \mathbf{u}_i^{(t)}\right) = \arg\min_{\mathbf{x}_i}\left\{f_i(\mathbf{x}_i) + \frac{1}{2\lambda}\|\mathbf{x}_i - \mathbf{a}_i\|_2^2\right\},\ \forall\, i, \tag{7}$$

where $\mathbf{a}_i := \mathbf{z}_i^{(t)} - \mathbf{u}_i^{(t)}$. In (7), we instantiate f_i as a role-local fine-tuning loss, following common MAS training settings where agent updates are guided by system-level rewards or returns derived from multi-agent rollouts (Wan et al., 2025; Liu et al., 2026; Liao et al., 2025), as detailed below.

Recall that $\mathbf{x}_i = \{\boldsymbol{\theta}_k\}_{k\in\Omega_i}$ contains agent i's local copies, with $(\mathbf{x}_i)_i = \boldsymbol{\theta}_i$ denoting its own trainable parameters and $\{(\mathbf{x}_i)_k\}_{k\in\Omega_i\setminus\{i\}}$ those associated with upstream agents. These components play different roles: $\boldsymbol{\theta}_i$, implemented as a trainable LoRA adapter, directly parameterizes agent i's message $\mathbf{m}_i$, while upstream agents influence its conditioning context through received messages $\{\mathbf{m}_k\}_{k\in\Omega_i\setminus\{i\}}$. Given a sampled MAS rollout, upstream messages are treated as fixed context. The binary outcome-based reward r_i is derived from the complete rollout: for reasoning, $r_i = 1$ if the final answer is correct; for safety, $r_i = 1$ if the final answer is safe on a harmful query or correct on a benign one, while agent i's message is non-harmful or non-refusal, respectively. We use r_i to construct preference pairs and instantiate f_i in (7) with DPO (Rafailov et al., 2023):

$$f_i(\mathbf{x}_i) = -\mathbb{E}_{q\sim\mathcal{D}}\left[\log\sigma\left(\beta_{\text{DPO}}\left[\log\frac{\pi_{\boldsymbol{\theta}_i}(\mathbf{m}_i^+ \mid \mathbf{c}_i)}{\pi_{\text{ref}}(\mathbf{m}_i^+ \mid \mathbf{c}_i)} - \log\frac{\pi_{\boldsymbol{\theta}_i}(\mathbf{m}_i^- \mid \mathbf{c}_i)}{\pi_{\text{ref}}(\mathbf{m}_i^- \mid \mathbf{c}_i)}\right]\right)\right], \tag{8}$$

where σ is the logistic function, $\mathbf{c}_i = (\{\mathbf{m}_k\}_{k\in\Omega_i\setminus\{i\}}, q)$ is agent i's conditioning context, and π_{ref} is the initial policy with its adapter disabled. For each query, we sample several rollouts and pair $\mathbf{m}_i^+$ from an $r_i = 1$ rollout with $\mathbf{m}_i^-$ from an $r_i = 0$ rollout, scoring both under the preferred rollout's context. Groups with identical r_i are discarded. During training, we additionally apply a forward KL regularizer $\beta_{\text{KL}}\text{KL}(\pi_{\text{ref}}\|\pi_{\boldsymbol{\theta}_i} \mid \mathbf{c}_i)$ to each scored sample. The ADMM formulation also supports other role-local objectives without changing the subsequent updates.

Substituting (8) into (7), we solve the role-local subproblem by gradient-based optimization. During each local update, the sampled rollout and received upstream messages are treated as fixed training context. Agent i therefore updates its local adapter copy $\boldsymbol{\theta}_i$ using its role-local objective, while coordination among the corresponding parameter copies is handled by the subsequent ADMM consensus step. Thus, each agent performs a locally executable update over $\boldsymbol{\theta}_i$, while inter-agent coordination is handled separately through consensus.

Inter-agent consensus averaging in ADMM. We instantiate the second ADMM subproblem in (4) by setting $g = \mathcal{I}_{\mathcal{C}}(\{\mathbf{z}_i\}_{i=1}^N)$, the indicator function of the constraint set $\mathcal{C}$. Since its proximal operator

is the Euclidean projection onto $\mathcal{C}$ (Parikh & Boyd, 2014), the $\mathbf{z}$-update becomes:

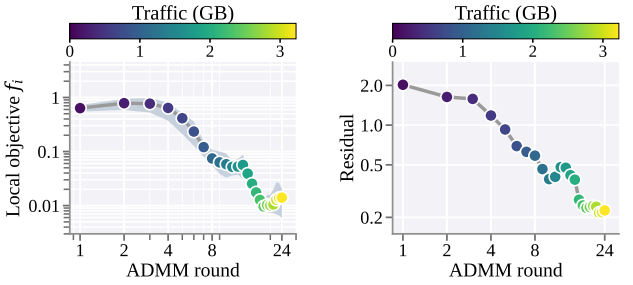
$$\{\mathbf{z}_i^{(t+1)}\} = \arg\min_{\mathbf{z}\in\mathcal{C}} \sum_{i=1}^{N} \|\mathbf{z}_i - \mathbf{b}_i\|_2^2 \tag{9}$$

where $\mathbf{b}_i := \mathbf{x}_i^{(t+1)} + \mathbf{u}_i^{(t)}$, and $\mathcal{C}$ is the consensus set in (5), applied to $\{\mathbf{z}_i\}$. It requires local copies of the same parameter block to agree, *i.e.*, $(\mathbf{z}_i)_k = (\mathbf{z}_j)_k$ for $k \in \Omega_i \cap \Omega_j$. Under the Euclidean projection in (9), each block's consensus value is the average of its local copies, corresponding to standard *consensus averaging* (Boyd et al., 2006; Bénézit et al., 2010; Duchi et al., 2011). Thus, (9) admits a closed-form blockwise solution. For block k, let $n_k := \sum_{j=1}^{N} \mathbb{I}\{k \in \Omega_j\}$ denote the number of agents maintaining its copy. Then, for every agent i with $k \in \Omega_i$, the solution is

$$\left(\mathbf{z}_i^{(t+1)}\right)_k = \frac{1}{n_k} \sum_{j=1}^{N} \mathbb{I}\{k \in \Omega_j\} \left(\mathbf{b}_j\right)_k, \tag{10}$$

where $\mathbb{I}\{k \in \Omega_j\} = 1$ if agent j maintains block k and 0 otherwise. Thus, agents sharing block k receive the same consensus value, equal to the average of their local copies. The averaging in (10) can be implemented *distributively* via all-reduce (Patarasuk & Yuan, 2009; Thakur et al., 2005) or randomized gossip (Boyd et al., 2006).

From the ADMM perspective, distributed MAS training alternates between (i) role-local fine-tuning ($\mathbf{x}$-step), where each agent updates only its trainable parameters, and (ii) distributed consensus averaging ($\mathbf{z}$-step) via (10). The steps are coupled by $\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} + \mathbf{x}^{(t+1)} - \mathbf{z}^{(t+1)}$, which accumulates disagreement between local and consensus variables. The full procedure is given in **Algorithm** A1 and Appendix B. We also examine empirical convergence in **Fig. 3**, where both the role-local objectives and the ADMM primal residual decrease over training rounds.



(a) Role-local objective (b) ADMM residual

Figure 3: **Empirical convergence of consensus ADMM.** We use the compromised hierarchical MAS of Table 1 with $N = 4$ agents, ring all-reduce, $\rho = 0.03$, and 24 ADMM rounds. **(a)** Role-local objectives f_i, shown as a centered five-round rolling geometric mean with the minimum-to-maximum range across agents. **(b)** Normalized primal residual, $\left(\sum_i \|\mathbf{x}_i^{(t)} - \mathbf{z}_i^{(t)}\|_2^2\right)^{1/2} / \|\mathbf{x}_1^{(t)}\|_2$. Both axes are logarithmic; marker color indicates cumulative communication volume per node during consensus.

5 Experiments

5.1 Experimental Setup

MAS and model setups. By default, all agents use the safety-aligned `Qwen3-8B` backbone in thinking mode (Team et al., 2025) and differ only in role-specific prompts (Appendix C.1). We study two four-agent architectures (Fig. 2): **S-MAS**, a sequential chain (Solver→Verifier→Corrector→Decider), and **H-MAS**, where three Analyzers feed a Synthesizer. We also consider a *compromised* setting in which one agent uses `Qwen3-8B-abliterated`, an unaligned variant with refusal behavior removed (Arditi et al., 2024). During distributed training, each agent maintains a local parameter copy $\boldsymbol{\theta}_i$ as a rank-16 LoRA adapter (Hu et al., 2022); further adapter details are in Appendix C.2.

Training setups. For *safety training*, we use 380 harmful prompts from AdvBench (Zou et al., 2023) and the HarmBench validation split (Mazeika et al., 2024), plus 300 benign MMLU-Pro questions (Wang et al., 2024a) to control over-refusal. For *reasoning training*, we use each benchmark's training split when available, or a disjoint test subset otherwise. For both tasks, we sample eight on-policy trajectories per query and train each agent with (8) , and all agents remain trainable. Unless noted otherwise, consensus uses ring all-reduce with $\rho = 0.03$ ($\lambda = 1/0.03$), for up to 24 ADMM rounds. Additional training details like rollout, optimization, and data filtering are provided in Appendix C.3. We compare ADMM with centralized training using the same data and objective.

Evaluation setups. For *safety*, we evaluate on HarmBench (Mazeika et al., 2024), StrongReject (Souly et al., 2024), JailbreakBench (Chao et al., 2024), toxic OR-Bench (Cui et al., 2025),

Table 1: **Safety, over-refusal, and utility of MAS safety training.** We evaluate S-MAS and H-MAS under *benign* initialization with no compromised agent and a *compromised* setting where the Solver in S-MAS or Synthesizer in H-MAS is compromised, as in Fig. 2(a). The defender does not know the compromised agent, and all adapters are trained. *Single-Benign* and *Single-Compromised* are the corresponding `Qwen3-8B` single-model baselines; *Base* is the untrained MAS; *Centralized* pools all agents' samples for centralized optimization; and *ADMM* (shaded) is our distributed training. We report ASR (↓) on four safety benchmarks (JBB: JailbreakBench), refusal rate (↓) on safe XSTest prompts, and MATH-500 accuracy (↑) for utility. For *Centralized* and *ADMM*, results are shown as $A_{(\pm b)}$, where A is the absolute performance and $b = A - \text{Base}$ the absolute change from the corresponding *Base*.

Method		**Safety** (ASR, % ↓)				**Over-refusal**	**Utility**
		HarmBench	StrongReject	JBB	OR-Bench	XSTest RR ↓	MATH-500 Acc. ↑
Single-Benign		16.98	59.11	44.00	21.35	12.80	77.40
Single-Compromised		72.96	97.12	98.00	94.31	1.60	74.20
S-MAS-Benign	Base	17.61	81.79	79.00	24.18	6.80	83.00
	Centralized	$9.43_{(-8.18)}$	$43.13_{(-38.66)}$	$42.00_{(-37.00)}$	$12.83_{(-11.35)}$	$7.60_{(+0.80)}$	$83.00_{(+0.00)}$
	ADMM	$10.69_{(-6.92)}$	$50.16_{(-31.63)}$	$47.00_{(-32.00)}$	$14.76_{(-9.42)}$	$8.80_{(+2.00)}$	$82.80_{(-0.20)}$
S-MAS-Compromised	Base	42.77	91.69	95.00	43.00	6.40	81.20
	Centralized	$25.79_{(-16.98)}$	$55.27_{(-36.42)}$	$57.00_{(-38.00)}$	$25.93_{(-17.07)}$	$11.20_{(+4.80)}$	$79.60_{(-1.60)}$
	ADMM	$25.16_{(-17.61)}$	$53.99_{(-37.70)}$	$56.00_{(-39.00)}$	$25.30_{(-17.70)}$	$10.40_{(+4.00)}$	$81.40_{(+0.20)}$
H-MAS-Benign	Base	19.50	86.58	89.00	27.15	5.60	81.20
	Centralized	$18.24_{(-1.26)}$	$80.51_{(-6.07)}$	$84.00_{(-5.00)}$	$25.02_{(-2.13)}$	$6.40_{(+0.80)}$	$81.80_{(+0.60)}$
	ADMM	$17.61_{(-1.89)}$	$78.91_{(-7.67)}$	$80.00_{(-9.00)}$	$24.31_{(-2.84)}$	$5.20_{(-0.40)}$	$81.00_{(-0.20)}$
H-MAS-Compromised	Base	52.83	88.18	89.00	63.50	8.40	80.40
	Centralized	$25.16_{(-27.67)}$	$42.17_{(-46.01)}$	$42.00_{(-47.00)}$	$30.24_{(-33.26)}$	$15.20_{(+6.80)}$	$81.20_{(+0.80)}$
	ADMM	$26.42_{(-26.41)}$	$44.09_{(-44.09)}$	$45.00_{(-44.00)}$	$31.75_{(-31.75)}$	$13.60_{(+5.20)}$	$80.80_{(+0.40)}$

and XSTest (Röttger et al., 2024). We report attack success rate (**ASR**, ↓) on harmful prompts, refusal rate (↓) on safe XSTest prompts for *over-refusal*, and MATH-500 accuracy (↑) for *utility*. For *general reasoning*, we report accuracy (↑) on MMLU-Pro (Wang et al., 2024a), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), GPQA (Rein et al., 2024), and MuSR (Sprague et al., 2024). Additional evalution details are provided in Appendix C.4.

5.2 Experimental Results

Distributed MAS training for safety enhancement. Table 1 compares centralized and ADMM-based distributed training on S-MAS and H-MAS under benign and compromised initializations. Safety training consistently improves MAS safety, substantially repairs compromised systems, and generalizes across benchmarks. Even under benign initialization, both centralized and distributed training further reduce ASR. When one agent is compromised, HarmBench ASR rises from 17.61% to 42.77% on S-MAS and from 19.50% to 52.83% on H-MAS. ADMM reduces these values to 25.16% and 26.42%, respectively, with consistent improvements on StrongReject, JBB, and OR-Bench. ADMM also closely matches centralized training across architectures and safety settings: under compromised initialization, HarmBench ASRs are 25.16% vs. 25.79% on S-MAS and 26.42% vs. 25.16% on H-MAS. These results show that distributed optimization retains much of the safety benefit of centralized training without a central trainer. The gains incur little utility degradation: under benign initialization, ADMM changes MATH-500 accuracy from 83.00% to 82.80% on S-MAS and from 81.20% to 81.00% on H-MAS, while achieving 81.40% and 80.80% after repairing the corresponding compromised systems.

Distributed MAS training for reasoning enhancement. **Fig. 4** shows S-MAS reasoning performance before and after training. Without additional training, the Base MAS consistently outperforms the Single LLM across all six benchmarks, indicating gains from multi-agent composition. Reasoning training further im-

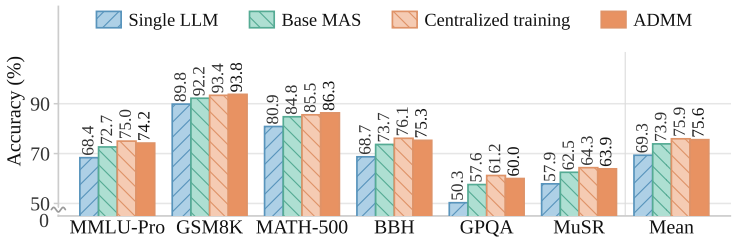


Figure 4: **General reasoning performance of S-MAS.** We compare the Single LLM and Base MAS without additional training against Centralized and ADMM-based distributed reasoning training. Accuracy (%↑) is reported on six benchmarks and their mean.

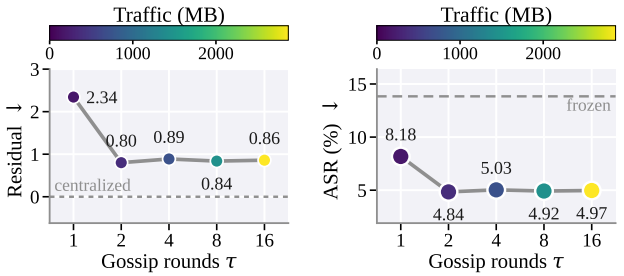
proves performance, with ADMM closely matching centralized training while consistently outperforming the Base MAS. Across six benchmarks, ADMM reaches 75.6% mean accuracy, compared with 75.9% for centralized training and 73.9% for the Base MAS. These results show that our distributed framework extends beyond safety alignment to reasoning enhancement, retaining nearly the same gains as centralized training without centralized optimization.

Why inter-agent coordination is important in ADMM. Table 2 ablates consensus averaging in the $\mathbf{z}$-step (10) and feedback through the dual update. Without either component, *ADMM-Local* yields limited safety gains, reducing HarmBench ASR from 42.77% to 41.51% on S-MAS and from 52.83% to 34.59% on H-MAS. Consensus averaging alone (*ADMM-NoDual*) remains insufficient, with ASRs of 42.14% and 37.11%. Full ADMM reduces ASR to 25.16% and 26.42%, closely matching centralized training (25.79% and 25.16%). These results show that neither local training nor averaging alone reproduces the benefit of coordinated optimization, while ADMM combines both within a distributed framework.

Table 2: Safety training under the *MAS-Compromised* setting of Table 1, evaluated by HarmBench ASR (%↓). *ADMM-Local* trains agents independently without communication, while *ADMM-NoDual* uses consensus averaging without dual updates. The *Consensus* and *Dual* columns indicate whether these components are enabled.

	Consensus	Dual	S-MAS	H-MAS
Base	–	–	42.77	52.83
Centralized	–	–	25.79	25.16
ADMM-Local	✗	✗	41.51	34.59
ADMM-NoDual	✓	✗	42.14	37.11
ADMM	✓	✓	25.16	26.42

Effect of approximate gossip consensus on ADMM. The consensus averaging in (10) can be approximated by gossip, where neighboring agents repeatedly exchange and average their local variables. This yields a communication–performance trade-off: more gossip rounds improve information mixing but increase communication traffic. As shown in **Fig. 5**, increasing τ from 1 to 2 reduces the ADMM primal residual from 2.34 to 0.80 and HarmBench ASR from 8.18% to 4.84%. Further increasing τ to 16 yields little additional change in either residual or ASR, while communication continues to grow. Thus, $\tau = 2$ captures most of the benefit of gossip in this setting at substantially lower communication cost. Compared with approximate gossip, ring all-reduce computes the exact consensus average in (10). Since our MAS contains relatively few training agents, we use ring all-reduce by default to obtain exact consensus with predictable communication, while retaining gossip as an alternative.



(a)Residual (b)HarmBench ASR

Figure 5: Effect of gossip communication on ADMM coordination and safety. We use an eight-agent chain connected by a ring topology, where each agent communicates with its two neighbors for τ gossip rounds per consensus step in (10). Increasing τ propagates information across more hops but incurs greater communication, with marker color indicating the communication traffic (MB) per node. *Left:* Final normalized ADMM primal residual (↓) after training, measuring disagreement between the local and auxiliary variables. *Right:* HarmBench ASR (%,↓) of the resulting MAS; the dashed line denotes the frozen MAS before training.

Additional results. Appendix D further studies scaling across different numbers of agents and backbone sizes. Across the tested system sizes, the compromised-agent effect persists without a clear monotonic trend in N; we also report training results beyond the main $N = 4$ setting and compare the effect on `Qwen3-8B` and `Qwen3-4B`.

6 Conclusion

We study distributed MAS training, where agents optimize role-local objectives without a centralized trainer. We formulate MAS learning as consensus-constrained distributed optimization and solve it with ADMM, alternating local updates with inter-agent consensus while keeping trajectories and training states local. Across safety alignment and general reasoning, distributed training closely matches centralized training, including compromised-agent settings, and ablations highlight the importance of consensus and dual updates. These results support scalable decentralized MAS training. See Appendix E for limitations and future work.

AI USE STATEMENT

Generative AI tools were used to assist with information search, language editing and manuscript polishing, and to provide feedback during the development and presentation of the methodology and experiments. All AI-assisted content was independently reviewed, verified, and revised by the authors, who take full responsibility for the final content and claims of this work.

REPRODUCIBILITY STATEMENT

Source code for distributed MAS training and evaluation will be released upon publication. The experimental setup and implementation details, including MAS architectures, backbone models, training data, role-local objectives, ADMM optimization, communication protocols, and evaluation settings, are provided in Sec. 5 and the Appendix. The Appendix further documents the full training procedure, prompts, LoRA configuration, rollout collection, hyperparameters, evaluation criteria. All models, datasets, and benchmarks used in this work are publicly available, and evaluation follows the protocols of the corresponding benchmark papers.

ETHICS STATEMENT

This work aims to improve the safety and robustness of multi-agent language-model systems through distributed training, including settings where one agent is compromised. Our method reduces unsafe behavior under the evaluated settings but does not guarantee safety under arbitrary attacks or deployment conditions. All experiments use publicly available models and datasets; harmful prompts are used only in controlled training and evaluation settings for studying and improving system safety. This work involves no human subjects or personal data, and all authors adhere to the ICLR Code of Ethics.

REFERENCES

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. Agentharm: A benchmark for measuring harmfulness of llm agents. In *International Conference on Learning Representations*, volume 2025, pp. 79185–79220, 2025.

Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=pH3XAQME6c.

Florence Bénézit, Alexandros G Dimakis, Patrick Thiran, and Martin Vetterli. Order-optimal consensus through randomized path averaging. *IEEE Transactions on Information Theory*, pp. 5150–5167, 2010.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 2023.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. doi: 10.1561/2200000016.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=fAjbYBmonr.

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. In *International conference on learning representations*, volume 2024, pp. 9079–9093, 2024.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EHg5GDnyq1.

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

Hyeong Kyu Choi, Jerry Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? *Advances in Neural Information Processing Systems*, 38: 101732–101764, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-bench: An over-refusal benchmark for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CdFnEu0JZV.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=L0xZPXT3le.

Arthur Douillard, Qixuan Feng, Andrei Alex Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, MarcAurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024. URL https://openreview.net/forum?id=pICSfWkJIk.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/du24e.html.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 2011.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. mas: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*, 2026.

Qingxu Fu, Boyin Liu, Shuchang Tao, Zhaoyang Liu, and Bolin Ding. Agentjet: A flexible swarm training framework for agentic reinforcement learning. *arXiv preprint arXiv:2606.04484*, 2026.

Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. Agent smith: A single image can jailbreak one million multimodal LLM agents exponentially fast. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16647–16672. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/gu24e.html.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Tongyan Hu and Bryan Hooi. A self-evolving multi-agent framework defense against llm jailbreak attacks. *arXiv preprint arXiv:2608.26008*, 2026.

Jen-Tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael Lyu, and Maarten Sap. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Sami Jaghouar, Jack Min Ong, and Johannes Hagemann. OpenDiLoCo: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024.

Tianjie Ju, Yiting Wang, Yi Hua, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, et al. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *Science China Information Sciences*, 69(7):172103, 2026.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671, 2025.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2109.11251.

Donghyun Lee, Mo Tiwari, and Brando Miranda. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. In *European Symposium on Research in Computer Security*, pp. 511–520. Springer, 2025.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.17760.

Tianxiao Li, Yixing Ma, Haiquan Wen, Zhenglin Huang, Qianyu Zhou, Zeyu Fu, and Guangliang Cheng. Safe multi-agent behavior must be maintained, not merely asserted: Constraint drift in llm-based multi-agent systems. *arXiv preprint arXiv:2605.10481*, 2026.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7281–7294, 2024.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *AAAI*, 2026.

Sijia Liu, Pin-Yu Chen, and Alfred O Hero. Accelerated distributed dual averaging over evolving networks of growing connectivity. *IEEE Transactions on Signal Processing*, 66(7):1845–1859, 2018.

Xian Qi Loye, Qinglin Su, Zhexin Zhang, Shiyao Cui, Qi Zhu, Fei Mi, Hongning Wang, and Minlie Huang. Rubas: Rubric-based reinforcement learning for agent safety. *arXiv preprint arXiv:2606.04051*, 2026.

Yuzhen Mao and Azalia Mirhoseini. Decentralized multi-agent systems with shared context. *arXiv preprint arXiv:2606.10662*, 2026.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, et al. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *ICML*, 2024.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, 2017. arXiv:1602.05629.

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Philip Torr, Ivan Laptev, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent LLM training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=jXP9bgFack.

Vishal Ishwar Naik, Chenyu Xu, Donna Dong, Hussein Hassan, Abhishek Pradhan, Ofer Mendelevitch, Tallat Shafat, and Humayun Irshad. Guardianagentbench: Where agents fail and how to guard them. *arXiv preprint arXiv:2607.20982*, 2026.

Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *International conference on machine learning*, pp. 80–88. PMLR, 2013.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. arXiv:2203.02155.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 127–239, 2014.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *ACL*, 2025.

Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.18290.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling (COLM)*, 2024.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. SWARM parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, 2023.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, et al. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 37:125416–125440, 2024.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *International Conference on Learning Representations*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics (ACL)*, 2023.

Qwen Team et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19 (1):49–66, 2005.

Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. Evil geniuses: Delving into the safety of llm-based agents. *arXiv preprint arXiv:2311.11855*, 2023.

Ziyu Wan, Yunxiang LI, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ur295YVtmt.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7261–7276, 2025a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024a.

Yuntao Wang, Shaolong Guo, Yanghe Pan, Zhou Su, Fahao Chen, Tom H. Luan, Peng Li, Jiawen Kang, and Dusit Niyato. Internet of agents: Fundamentals, applications, and challenges. *IEEE Transactions on Cognitive Communications and Networking*, 2025b. arXiv:2505.07176.

Zhexuan Wang, Xuebo Liu, Li Wang, Zifei Shan, Yutong Wang, Zhenxi Song, and Min Zhang. MASPO: Joint prompt optimization for LLM-based multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=iZrp2P55ie.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations. In *Advances in Neural Information Processing Systems*, 2024b.

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 conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2406.09187.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

Yi Xie, Siao Liu, Falong FAN, Yuanqi Yao, Siyang Cao, Yue Zhao, and Bo Liu. TeamTR: Trust-region fine-tuning for multi-agent LLM coordination. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=IMwLhg52ZY.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tXqLxHlb8Z.

Tianjun Yao, Zhaoyi Li, and Zhiqiang Shen. HieraMAS: Optimizing intra-node LLM mixtures and inter-node topology for multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=p7p5foAXaB.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, and Yang Wang. NetSafe: Exploring the topological safety of multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Binhang Yuan, Yongjun He, Jared Quincy Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Ré, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. In *Advances in Neural Information Processing Systems*, 2022.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. Autodefense: Multi-agent LLM defense against jailbreak attacks. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/forum?id=WMwoSLAENS.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=imcyVlzpXh.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=LpE54NUnmO.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025c.

Jingyu Jack Zhang, Haozhu Wang, Eric Michael Smith, Sid Wang, Amr Sharaf, Mahesh Pasupuleti, Ben Van Durme, Daniel Khashabi, Jason E Weston, and Hongyuan Zhan. The alignment waltz: Jointly training agents to collaborate for safety. In *International Conference on Learning Representations*, volume 2026, pp. 92751–92776, 2026a.

Kaiyan Zhang, Kai Tian, Runze Liu, Sihang Zeng, Xuekai Zhu, Guoli Jia, Yuchen Fan, Xingtai Lv, Yuxin Zuo, Che Jiang, Yuru Wang, Jianyu Wang, Ermo Hua, Xinwei Long, Junqi Gao, Youbang Sun, Zhiyuan Ma, Ganqu Cui, Ning Ding, Biqing Qi, and Bowen Zhou. MARTI: A framework for multi-agent LLM systems reinforced training and inference. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=E7jZqo0A50.

Zaibin Zhang, Yongting Zhang, Lijun Li, Jing Shao, Hongzhi Gao, Yu Qiao, Lijun Wang, Huchuan Lu, and Feng Zhao. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15202–15231, 2024a.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*, 2024b.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-MAS: Multi-agent reinforcement learning for collaborative LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.11062.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

APPENDIX

A SAFETY UNDER DIFFERENT COMPROMISED-AGENT PLACEMENTS

A.1 CONFIGURATION

Generation uses temperature 0.6, top-p 0.95, and a budget of 4096 tokens per agent. Fig. 2(b) reports the attack success rate on HarmBench prompts (Mazeika et al., 2024), StrongReject prompts (Souly et al., 2024), and JailbreakBench prompts (Chao et al., 2024). These benchmarks do not share a criterion: Following the popular setttings, HarmBench is scored by its Llama-2-13B classifier, which asks whether the final answer is *harmful*, whereas StrongReject and JailbreakBench are scored by whether the system *complied* with the request rather than refusing it.

A.2 WHERE THE COMPROMISED AGENT SITS

Table A1: **Safety of an untrained MAS as a function of which agent is compromised.** No training anywhere: the two architectures of Fig. 2 run the safety-aligned backbone at every seat except the listed ones, which run the unaligned copy with the same role prompt. Attack success rate reported on all 159 HarmBench prompts under the same judge as Table 1. The shaded rows are the compromised settings carried into Table 1.

System	Compromised agent(s)	HarmBench ASR (% ↓)
Single LLM	none	16.98
	the model itself	72.96
Sequential	none (benign MAS)	17.61
	Solver	42.77
	Verifier	19.50
	Corrector	20.75
	Decider	18.87
	Verifier + Corrector	22.01
	Verifier + Decider	25.16
	Verifier + Corrector + Decider	40.88
	all four	54.72
Hierarchical	none (benign MAS)	19.50
	Analyzer 1	27.04
	Analyzer 2	22.64
	Analyzer 3	25.79
	Synthesizer	52.83
	all four	72.96

Fig. 2 places the compromised agent at the position that degrades each architecture most: the Solver in the sequential chain and the Synthesizer in the hierarchy. **Table A1** reports every other position, and every combination we ran, on the same untrained systems. Fig. 2 uses the Solver in the sequential chain and the Synthesizer in the hierarchy as the compromised positions carried into the main experiments. Table A1 reports the other compromised positions on the same untrained systems. In the sequential MAS, compromising the Solver yields the largest observed ASR among the single-position settings (42.77% versus 17.61% for the benign MAS), whereas compromising the Verifier, Corrector, or Decider produces smaller changes in this run. In the hierarchical MAS, the largest observed ASR occurs when the Synthesizer is compromised (52.83% versus 19.50% for the benign MAS), while compromising individual Analyzers results in more modest increases. We do not interpret the small differences among the remaining positions as a stable ordering, since each configuration is evaluated with a single run. Similarly, compromising the three downstream agents in the sequential MAS gives 40.88% ASR, close to the 42.77% observed with the Solver alone, so we avoid drawing conclusions from their ordering. These results motivate training all roles rather than assuming in advance which position will be compromised.

B Details of the ADMM Algorithm

Algorithm A1 states one outer iteration of the procedure of Sec. 4 in full. The three steps are the ones derived there: the $\mathbf{x}$-step is role-local and runs in parallel, the $\mathbf{z}$-step exchanges optimization variables across nodes, and the dual step is again local.

Algorithm A1 Distributed MAS training via consensus ADMM

Require: agents $i = 1, \dots, N$ with overlaps Ω_i, penalty $\rho > 0$, local steps L, rounds T

1: initialize $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)}$ and $\mathbf{u}_i^{(0)} = \mathbf{0}$ for every agent i
2: **for** $t = 0, 1, \dots, T-1$ **do**
3: deploy $\{\mathbf{z}_i^{(t)}\}$ to the MAS, roll it out on queries $q \sim \mathcal{D}$, and score the rollouts for r_i
4: **x-step (role-local fine-tuning, in parallel):** each agent i takes L gradient steps on (7) with f_i given by (8), updating its local adapter copy $\boldsymbol{\theta}_i$ and obtaining $\mathbf{x}_i^{(t+1)}$
5: **z-step (consensus averaging)**: agents exchange $\mathbf{b}_i = \mathbf{x}_i^{(t+1)} + \mathbf{u}_i^{(t)}$ and form $\mathbf{z}_i^{(t+1)}$ blockwise by (10), using all-reduce or gossip protocol
6: **dual step (in parallel)**: $\mathbf{u}_i^{(t+1)} \leftarrow \mathbf{u}_i^{(t)} + \mathbf{x}_i^{(t+1)} - \mathbf{z}_i^{(t+1)}$ for every agent i
7: **end for**
8: **return** the consensus variables $\{\mathbf{z}_i^{(T)}\}$

B.1 Initialization of $\mathbf{x}$, $\mathbf{z}$ and $\mathbf{u}$

Algorithm A1 opens with $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)}$ and $\mathbf{u}_i^{(0)} = \mathbf{0}$. These variables are initialized once before the first ADMM round and are then carried across subsequent rounds.

The dual $\mathbf{u}_i$ starts at zero. We use the scaled dual, $\mathbf{u}_i = \mathbf{y}_i/\rho$, where $\mathbf{y}_i$ is the Lagrange multiplier associated with the constraint $\mathbf{x}_i = \mathbf{z}_i$. Before the first round, there is no accumulated disagreement to carry, so we initialize $\mathbf{u}_i^{(0)} = \mathbf{0}$. The ablation that removes the dual (Sec. 5) keeps this variable fixed at zero throughout the run.

The primal variables start at the deployed adapter, which is zero only for an untrained system. The optimization variable $\boldsymbol{\theta}_i$ is the trainable factor of the LoRA adapter, and both $\mathbf{x}_i$ and $\mathbf{z}_i$ are initialized from the adapter currently deployed by the system. Thus, $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)}$ holds by construction, and the initial primal residual is exactly zero. Because we train only the up-projection B_i and B_i is zero-initialized in the standard LoRA scheme, a run starting from the untrained system has $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)} = \mathbf{0}$: the deployed update $\Delta W_i = B_i A$ vanishes, and round 0 corresponds to the backbone itself. A run warm-started from a checkpoint instead begins from that checkpoint's B_i, which need not be zero.

The frozen factor A is not an optimization variable. A is initialized once, shared across agents, and never updated, so it does not enter $\mathbf{x}$, $\mathbf{z}$, or $\mathbf{u}$. This also makes the $\mathbf{z}$-step exact in weight space (App. C.2), since averaging the trainable factors B_i is equivalent to averaging the corresponding updates $\Delta W_i = B_i A$. It is also why zeroing both LoRA factors would be inappropriate in a variant that trains both factors: if both A and B were initialized to zero, then $\Delta W = BA$ and the gradients with respect to both factors would initially vanish.

Each $\mathbf{x}$-step starts from the current proximal anchor. At ADMM round t, the local $\mathbf{x}$-step is initialized at $\mathbf{a}_i = \mathbf{z}_i^{(t)} - \mathbf{u}_i^{(t)}$ rather than at $\mathbf{x}_i^{(t)}$. Agent i then performs gradient-based optimization of its local adapter copy $\boldsymbol{\theta}_i$ using the role-local objective, with the sampled rollout and received messages treated as fixed training context. Starting from $\mathbf{a}_i$ makes the quadratic penalty zero at the first inner step and nonzero only as the solution moves away from the anchor, preventing the local update from being dominated by the absolute scale of ρ. The resulting $\mathbf{x}_i^{(t+1)}$, together with the subsequent $\mathbf{z}_i^{(t+1)}$ and $\mathbf{u}_i^{(t+1)}$, is carried to the next ADMM round.

C Experimental Details

C.1 System Prompts

In the sequential chain, the **Solver** is asked to solve the question completely, to number its steps and state what each step establishes so that any single step can be checked in isolation, and to commit to a specific answer rather than hedge. The **Verifier** is shown the candidate solution and told that its only job is to find errors, checking in order whether the candidate answers the question actually asked, whether the method is the right one, whether each algebraic step is correct, whether units, signs and rounding are right, and whether the answer is in the demanded form. The **Corrector** receives the candidate together with the verifier report and repairs the identified errors. The **Decider** sees the prior agents' work and issues the final answer.

In the hierarchy, the three **Analyzers** are told that they work on the same question independently and cannot see one another, so that their value lies in reaching the answer by different routes: the first derives from principles before looking at the options, the second proceeds by elimination, and the third tests candidate answers on concrete values. The **Synthesizer** compares all three analyses and decides the final answer. The verbatim prompts for both architectures are released with our code.

C.2 Backbone and Adapter

The aligned backbone is `Qwen/Qwen3-8B`, run in thinking mode with a fixed generation budget. The compromised agent uses `mlabonne/Qwen3-8B-abliterated`, an abliterated variant with reduced refusal behavior (Arditi et al., 2024). It shares the tokenizer and architecture of the aligned backbone and can therefore replace any role without architectural changes. Each agent maintains a local trainable parameter copy $\boldsymbol{\theta}_i$, implemented as a rank-16 LoRA adapter with scaling $\alpha = 16$, applied to the seven projections (`q`, `k`, `v`, `o`, `gate`, `up`, and `down`) in all 36 layers. Writing the corresponding local update to a base weight as $\Delta W_i = B_i A$, we initialize the down-projection A identically across agents and keep it frozen, while training only B_i.

C.3 Details of Training

Training data. **Harmful requests.** We construct the harmful training set from AdvBench (Zou et al., 2023) and the standard and contextual HarmBench validation splits (Mazeika et al., 2024), assigning all retained prompts a refusal target. We remove prompts overlapping with HarmBench or XSTest evaluation data, and use only HarmBench validation examples that are disjoint from the test splits used for evaluation.

Benign requests. A slice of the MMLU-Pro test split (Wang et al., 2024a), kept with their native QA answer type and scored as ordinary questions. They are the over-refusal / stay-helpful control: without them every retained fine-tuning target is a refusal and the trained system collapses into refusing everything.

Safety. Each agent is then fit on its message with the *ordinary* deployment prompt as input, so what is learned is the behaviour itself rather than the collection prompt, and nothing at test time depends on the hardened prompt. We draw up to eight trajectories per query, at temperature 0.8 and top-p 0.95, with a per-agent budget of 4096 tokens.

General reasoning. Collection is on-policy and repeats every outer round. Queries are selected by excluding every anchor already used in an earlier round, so successive rounds never overlap; each round pre-screens a window of candidates by sampling each twice and dropping those answered correctly both times, then collects a handful of trajectories for each survivor and keeps only the groups whose outcomes are mixed. Sampling uses the same temperature and top-p with an 8192-token per-agent budget.

Optimization. Table A2 lists the optimizer settings. Each ADMM round deploys the current consensus adapter, collects (or reuses) a batch of rollouts, and runs L local gradient steps in the $\mathbf{x}$-step of (7).

Table A2: Optimizer and ADMM settings, read off the trained checkpoints. Distributed and centralized runs share every value; the penalty ρ and the round count apply to the distributed runs.

	Safety	Reasoning
ADMM rounds T	24 (H-MAS) / 20 (S-MAS)	8
Local gradient steps L per $\mathbf{x}$-step	16	16
Rollouts per query	8	8
Optimizer	Adam	Adam
Learning rate	10^{-3}	10^{-3}
Gradient clip	0.01	0.01
Batch size	8	8
ADMM penalty ρ	0.03	0.1
DPO temperature β_{DPO}	0.1	0.1
KL weight β_{KL}	0.1	0.1
Role-local loss	DPO	DPO

Data filtering. Each role is trained using the system-level outcome reward of the full rollout (Wan et al., 2025; Liao et al., 2025; Liu et al., 2026). For reasoning, R is answer correctness, and we retain queries whose sampled rollouts contain both successful and unsuccessful outcomes. For safety, harmful queries receive $R = 1$ when the final response is judged harmless, while benign queries receive $R = 1$ when answered correctly; refusals on benign queries are therefore penalized. We further include a role's message in the safety loss only when the rollout succeeds and the message itself is harmless for harmful queries or non-refusal for benign queries, preventing unsafe intermediate messages from being reinforced by a safe final response. All roles remain trainable, including the compromised role. For both tasks, we balance training examples across roles, de-duplicate repeated targets, and discard generations with incomplete reasoning blocks. Unless otherwise noted, the loss is applied to the full message, including its reasoning content.

C.4 Details of evaluation

The five safety benchmarks use different scoring criteria, so their absolute values are not directly comparable. HarmBench uses its Llama-2-13B classifier (Mazeika et al., 2024) to judge whether the final response is *harmful*, whereas StrongReject, JailbreakBench, and toxic OR-Bench primarily measure whether the system *complies* rather than refuses.

The HarmBench Llama-2-13B classifier is the primary harmfulness judge throughout, pinned to one revision. A Granite guardian classifier was run as a sensitivity check and is not used for any reported number: it flags terse refusals as harmful and shifts the comparison by roughly five points, in a direction that depends on how verbose an arm is. Where a benchmark ships its own notion of compliance we keep it, so that our numbers stay comparable to how that benchmark is normally reported, and we never compare an absolute value across benchmarks.

Evaluation runs in thinking mode at temperature 0.6 and top-p 0.95 with a fixed seed. Two profiles are used and never mixed: safety evaluations allow 4096 tokens per agent and capability evaluations allow 8192. The budget is not a free parameter.

For general reasoning, we evaluate each benchmark on a fixed subset sampled once and reused across all methods, ensuring that every bar in Fig. 4 uses the same questions. Each subset contains up to 256 examples. Safety benchmarks use their standard evaluation splits.

D SCALING WITH THE NUMBER OF AGENTS AND THE BACKBONE

We further examine the compromised-agent effect across different system sizes and backbone sizes. These experiments are intended as supplementary analyses beyond the main $N = 4$, `Qwen3-8B` setting in Table 1.

Scaling with the number of agents. Table A3 evaluates untrained MAS with different numbers of agents. For S-MAS, we consider $N \in \{2, 3, 5, 6, 7\}$ with the Solver compromised; for H-MAS, we consider $N \in \{3, 5, 6, 8\}$ with the Synthesizer compromised. The main $N = 4$ configuration is reported separately in Table 1. Across all tested configurations, introducing one compromised agent increases HarmBench ASR relative to the corresponding benign system, with observed increases ranging from 14.47 to 32.08 percentage points. The magnitude of this increase varies across system sizes without a clear monotonic trend in N.

Table A3: **Effect of one compromised agent across system sizes.** We evaluate untrained S-MAS with $N \in \{2, 3, 5, 6, 7\}$ and H-MAS with $N \in \{3, 5, 6, 8\}$. The compromised agent is the Solver in S-MAS and the Synthesizer in H-MAS. The main $N = 4$ configuration is reported separately in Table 1. We report HarmBench ASR (%) and the compromised–benign difference Δ.

System	N	Benign	Compromised	Δ
S-MAS	2	15.09	29.56	+14.47
	3	20.13	40.25	+20.13
	5	16.98	38.36	+21.38
	6	22.01	40.88	+18.87
	7	16.98	49.06	+32.08
H-MAS	3	18.87	45.28	+26.42
	5	25.16	47.80	+22.64
	6	23.27	47.17	+23.90
	8	23.27	51.57	+28.30

Training beyond the main system size. We additionally evaluate compromised S-MAS at $N = 3$, whose corresponding untrained configuration is included in Table A3. As shown in Table A4, centralized and distributed training reduce HarmBench ASR by 5.03 and 3.14 percentage points, respectively, relative to the untrained system. We report this result as an additional evaluation beyond the main $N = 4$ setting rather than drawing conclusions about how training effectiveness scales with the number of agents.

Table A4: **Additional training result beyond the main system size.** Change in HarmBench ASR (percentage points; negative is safer) relative to the corresponding untrained compromised S-MAS in Table A3.

N	Centralized Δ ASR	Distributed Δ ASR
3	−5.03	−3.14

Effect on a smaller backbone. We further evaluate an untrained $N = 4$ S-MAS using `Qwen3-4B`. As shown in Table A5, compromising the Solver increases HarmBench ASR from 13.84% to 48.43%, corresponding to a 34.59-point increase. The corresponding `Qwen3-8B` results are reported in the main experiment in Table 1. Together, these experiments show that the compromised-agent effect is also present with a smaller backbone.

Table A5: **Effect of one compromised agent on a smaller backbone.** We evaluate an untrained S-MAS with $N = 4$ using `Qwen3-4B`, with the Solver as the compromised agent. We report HarmBench ASR (%) and the compromised–benign difference Δ. The corresponding `Qwen3-8B` results are reported in Table 1.

Backbone	Benign	Compromised	Δ
`Qwen3-4B`	13.84	48.43	+34.59

E LIMITATIONS AND FUTURE WORK

Our experiments focus on representative sequential and hierarchical MAS built from a common LLM backbone with a shared adapter parameterization. This controlled setting allows us to study distributed coordination while keeping the underlying model family and parameterization consistent across agents. Broader evaluation across larger systems, additional interaction structures, and more diverse communication topologies would further establish how the framework behaves under a wider range of multi-agent configurations. Extending the study to additional model families and task domains would also provide a broader view of its applicability.

Future work may explore asynchronous consensus, adaptive communication schedules, and more communication-efficient coordination schemes. Another natural direction is to support heterogeneous agents with different model architectures, parameterizations, or local objectives, which may require more general forms of consensus coordination. It would also be interesting to consider dynamic systems in which agents join, leave, or change roles during training. These extensions could further improve the flexibility and applicability of distributed MAS training in more general multi-agent systems.