LLM Collaboration with Multi-Agent Reinforcement Learning

Shuo Liu, Zeyu Liang, Xueguang Lyu, Christopher Amato
Khoury College of Computer Sciences,
Northeastern University
Boston, MA, 02115, USA
{liu.shuo2, liang.zey, lu.xue, c.amato}@northeastern.edu

Abstract

A large amount of work has been done in Multi-Agent Systems (MAS) for modeling and solving problems with multiple interacting agents. However, most LLMs are pretrained independently and not specifically optimized for coordination. For example, existing LLM fine-tuning frameworks rely on individual rewards, which require complex reward designs for each agent to encourage collaboration. To address this challenge, we model LLM collaboration as a cooperative Multi-Agent Reinforcement Learning (MARL) problem. We develop a multi-agent, multi-turn algorithm, Multi-Agent Group Relative Policy Optimization (MAGRPO), to solve it, building on current RL approaches for LLMs as well as MARL techniques. Our experiments on LLM writing and coding collaboration demonstrate that fine-tuning multiple LLMs with MAGRPO enables agents to generate high-quality responses efficiently through effective cooperation. Our approach opens the door to using MARL methods for LLM collaboration and highlights the associated challenges.

Introduction

State-of-the-art LLMs have demonstrated remarkable capabilities across diverse domains (Grattafiori et al. 2024; Achiam et al. 2023; Anil et al. 2025). To adapt to specific applications or align with human preferences, fine-tuning has emerged as a critical training stage. Compared to supervised fine-tuning, Reinforcement Learning (RL) enables more generalizable learning for complex, multi-turn tasks through human-aligned reward design, making it an important technique for fine-tuning (Ouyang et al. 2022; Guo et al. 2025; Ziegler et al. 2019).

Likewise, Multi-Agent Systems (MAS) have been extensively studied over the past decades, with substantial progress in modeling and solving problems involving multiple agents (Weiss 1999; Van der Hoek and Wooldridge 2008; Shoham and Leyton-Brown 2009; Stone and Veloso 2000). In particular, advances in cooperative MAS have demonstrated strong potential for enabling effective collaboration in distributed settings, such as games, robotics, and traffic control (Samvelyan et al. 2019; Vinyals et al. 2017; Berner et al. 2019; Wiering 2000; Liu et al. 2022; Xiao et al. 2025). These developments motivate the application of MAS principles and techniques to LLM collaboration, where multiple LLMs working together can solve more complex tasks more robustly and efficiently.

There has been some recent work on coordinating multiple LLMs. Some approaches implement coordination at the inference stage, enabling agents to interact through debate, discussion, or verification (Du et al. 2023; Wu et al. 2023; Lifshitz, McIlraith, and Du 2025). These methods operate at the prompt level, with fixed models that are not tuned toward coordination-centric objectives. The agents may have conflicting answers or spread incorrect information to other participants, limiting performance (Cemri et al. 2025; Estornell and Liu 2024). Moreover, the design of effective prompts remains difficult and unclear. Other approaches fine-tune agents independently with individual or role-conditioned rewards. However, they require carefully curated rewards for each individual or role (Slumbers et al. 2024; Liu et al. 2025a; Subramaniam et al. 2025), and, as independent learning methods, lack convergence guarantees (Tan 1993).

We model LLM collaboration as a cooperative MARL problem (Albrecht, Christianos, and Schäfer 2024; Amato 2025) and formalize it as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek and Amato 2016). In LLM collaboration, multiple trainable LLMs generate responses synchronously based on their individual prompts. The external environment evolves according to the joint responses until the dialog ends. This general model allows a wide range of problems to be modeled and solved using versions of MARL algorithms. Following the efficient practice of Group Relative Policy Optimization (GRPO) (Shao et al. 2024), we propose Multi-Agent GRPO (MAGRPO) that trains LLMs in a multi-turn setting. MAGRPO leverages centralized group-relative advantages for joint optimization, while preserving decentralized execution for each agent. The resulting method builds off of state-of-the-art LLM approaches in GRPO and MARL approaches for centralized training and decentralized execution, such as MAPPO (Yu et al. 2022). Our experiments demonstrate that MAGRPO is able to learn different cooperation schemes while producing efficient and high-quality responses.

Our contributions can be summarized as follows: (i) We model the LLM collaboration as a cooperative MARL problem, where multiple LLMs cooperate to generate joint responses; (ii) We develop the MAGRPO algorithm, which optimizes agent cooperation through aligned rewards while maintaining decentralized execution to maintain efficiency; (iii) Our experiments demonstrate that fine-tuning with MAGRPO improves both response efficiency and quality in writing and coding collaboration; (iv) We provide an analysis of the limitations of existing approaches and outline open challenges in applying MARL to LLM collaboration.

Related Work

Test-Time Multi-Agent Interaction Recent work employs multiple agents with specialized roles interacting through diverse pipelines at test-time to enhance response quality. In multi-agent debate, agents iteratively formulate positions by reviewing other agents' outputs, where the final decision or answer is determined by majority voting or a summarizer (Du et al. 2023; Chan et al. 2023; Liang et al. 2024). Role-based approaches allocate tasks across specialized agents (Wu et al. 2023; Qian et al. 2024; Hong et al. 2023). An agent may function as a verifier to assess the correctness of outputs (Skreta et al. 2023; Lifshitz, McIlraith, and Du 2025), or as a macro-planner to orchestrate workers' responses. However, these multi-agent frameworks rely on prompt-level interactions among agents, often leading to ineffective communication and computational inefficiency. Moreover, the design of effective prompts and role assignment remains unclear, as prompts usually fail to reliably guide agent behavior, enforce role adherence, or support coherent coordination across tasks. These limitations motivate us to fine-tune LLMs in MAS to improve their cooperation.

Multi-Agent Fine-Tuning Recent work has explored fine-tuning LLMs to improve their performance across diverse domains, e.g., arithmetic reasoning, navigation, and hidden-role games (Ma et al. 2024; Slumbers et al. 2024; Sarkar et al. 2025). These approaches typically employ individual rewards or rewards conditioned on specific roles (Liu et al. 2025a; Subramaniam et al. 2025; Zhang et al. 2025; Chen et al. 2025). Such reward structures often require careful manual specification, and their underlying rationale is rarely well justified. The misaligned or conflicting incentives can hinder effective coordination. Moreover, these methods lack convergence guarantees, as each agent learns independently in a non-stationary environment where other agents are simultaneously updating their policies. In this paper, we focus on cooperative scenarios, where LLMs are trained with verified, human-aligned rewards.

Cooperative MARL for LLM Collaboration

Since LLMs can act as a special class of agents, we leverage advances in MAS to improve their collaboration. We model LLM collaboration as a cooperative MARL problem and outline its unique challenges. We formalize this problem as a Dec-POMDP, as shown in Figure 1.

LLM Collaboration

LLM collaboration is the problem where LLMs cooperatively solve a class of tasks in MAS. Tasks are specified in language and provided to the LLMs as prompts. Each LLM generates a response synchronously in response to its own instructions. All responses jointly form a solution to the task.

Most tasks cannot be resolved in one turn. Users, external models, or systems validate the solutions and provide additional requirements or suggestions for LLMs. These components also serve as part of the environment for LLM collaboration, whose states may change based on the agents' outputs. The updates are embedded into prompts for subsequent turns. This iterative process continues until the task is successfully completed or a predefined turn limit is reached.

As discussed by a number of companies (NVIDIA 2024; Anthropic 2024), a team of agents could generate a complex codebase. The code would be difficult, costly, and time-consuming to generate with a single agent, but a group of LLMs could do so quickly and cheaply. None of these agents is self-interested, but they can be trained using a scheme such as the one in this paper. Using a joint reward allows agents to specialize as needed to complete the task without complex prompt or reward engineering.

Problem Formalization

We formalize LLM collaboration as a subclass of the cooperative MARL problem, considering LLMs are agents and the types of problems they are solving. This problem is a form of Dec-POMDP (Oliehoek and Amato 2016), which allows cooperation through a joint reward while preserving scalable decentralized control. We show 2 instantiations of our framework in writing and coding tasks in the experiments.

Mathematically, our LLM Dec-POMDP is defined by a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}, \{\mathcal{A}_i\}, R, T, H \rangle$.

- $\mathcal{I} = \{1, \cdots, n\}$ denotes the set of n LLM agents, each instantiated with a pre-trained language model.
- $\mathcal{S}$ denotes the full global state space. At turn t, a full state $s_t = (s_t^{\text{acc}}, s_t^{\text{usr}})$ consists of parts that are accessible in the model and provided to the reward model $s_t^{\text{acc}} \in \mathcal{S}^{\text{acc}}$ (e.g., external models or systems), and the inaccessible user state $s_t^{\text{usr}} \in \mathcal{S}^{\text{usr}}$ that updates over time but is not maintainable. In a Dec-POMDP, the state can not be directly observed by the agents.
- $\mathcal{O}_i$ is the observation space for agent i with $\mathcal{O} = \times_i \mathcal{O}_i$ the joint observation space. A local observation $o_{i,t}$ consists of natural language instructions (i.e., prompts), providing a partial and noisy view of s_t.
- $\mathcal{A}_i$ is the action space for agent i with $\mathcal{A} = \times_i \mathcal{A}_i$ the joint action space. A local action $a_{i,t}$ is a response in natural language to the given prompt.
- $R : \mathcal{S}^{\text{acc}} \times \mathcal{A} \to \mathbb{R}$ is the joint reward function implemented via predefined rules or a pretrained reward model. At turn t, the joint rewards r_t are determined by the accessible part of current state s_t^{acc} and the agents' joint action $\mathbf{a}_t = \{a_{1,t}, \cdots, a_{n,t}\}$.
- $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the underlying stochastic state transition function. At turn t, the agents' joint actions $\mathbf{a}_t$ induce a shift to a new state $s_{t+1} \sim T(\cdot|s_t, \mathbf{a}_t)$, which reflects the updates in the user state and the states of external models and systems.
- H is the episode horizon, i.e., the turn limit of the dialog.

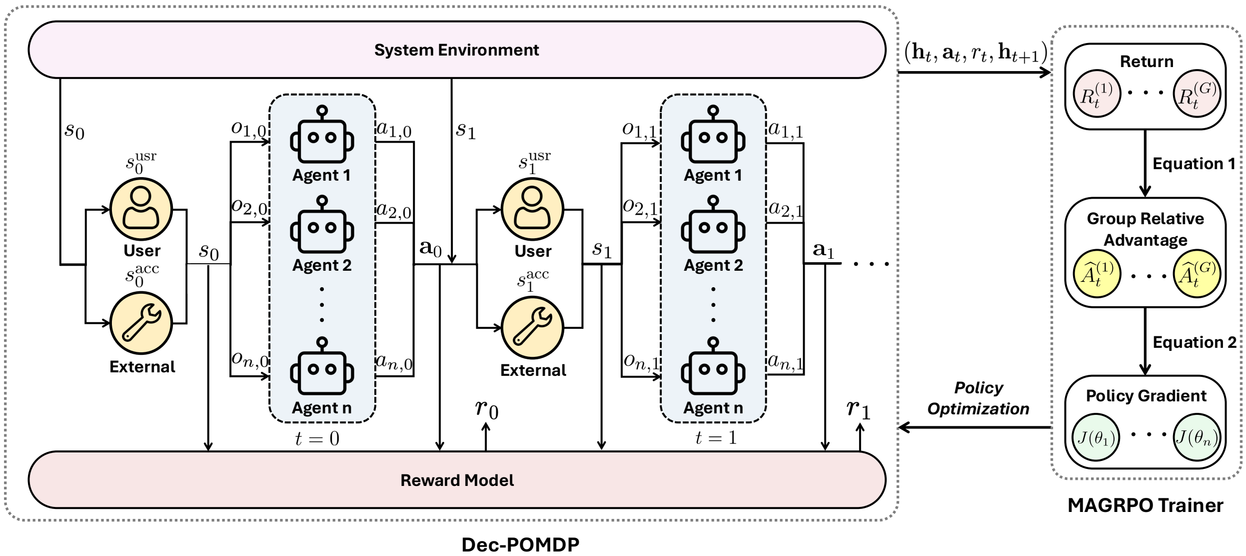


Figure 1: Illustration of Dec-POMDP and our MAGRPO algorithm.

In Dec-POMDP, since the states are not directly observed, each agent maintains its local observation-action history $\mathbf{h} = \{h_1, \cdots, h_n\}$ to infer information about the state. A solution to a Dec-POMDP is a joint policy that maximizes the expected cumulative reward, $\boldsymbol{\pi}^* = \{\pi_1^*, \cdots, \pi_n^*\} = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{H-1} R(s_t^{\mathrm{acc}}, \mathbf{a}_t)\right]$. A joint policy is a set of local policies π_i, which condition on the local observation-action history $h_{i,t} = \{o_{i,0}, a_{i,0}, \ldots, o_{i,t}\}$.

RL methods for Dec-POMDPs have become a popular topic (e.g., (Foerster et al. 2018b; Lowe et al. 2017; Foerster et al. 2018a; Rashid et al. 2020; Wang et al. 2021; Yu et al. 2022; Albrecht, Christianos, and Schäfer 2024; Lyu et al. 2023; Marchesini et al. 2025)) with methods successful at scaling to large state, action, and observation spaces. Many methods use Centralized Training for Decentralized Execution (CTDE), where they use some centralized information during training (e.g., a centralized value function estimate) but are still able to execute in a decentralized manner when training is completed (Amato 2024).

Challenges in LLM Collaboration

LLM collaboration presents unique challenges compared to traditional MARL problems, where LLM agents receive and process tasks through natural language.

Representations in Natural Language Unlike traditional cooperative MARL agents, LLM agents operate over natural language, receiving instructions and generating responses as sequences of tokens. MARL approaches could model this problem at the token or sequence level. At the token level, the number of actions and observations is smaller, but the problem horizon can be very long. At the sequence level, the action and observation space is much larger, but the horizon is much shorter. Moreover, token-level rewards are often uninformative, as both queries and responses must form coherent and semantically meaningful structures. As adopted in prior RL methods (Ouyang et al. 2022; Rafailov et al. 2023), we model each agent's decision-making process as a direct mapping from input instructions to complete responses to enable efficient and stable training. Nevertheless, the best modeling and solution approaches remain an open question.

Training Paradigm As mentioned above, many MARL methods use centralized training for decentralized execution (CTDE). Unfortunately, standard CTDE methods use centralized value models in the form of centralized critics (Foerster et al. 2018b; Lowe et al. 2017; Yu et al. 2022) or mixers in value decomposition methods (Rashid et al. 2020; Wang et al. 2021), which allow additional information during training but do not scale well to very large action and observation spaces (such as those in LLM collaboration). Conversely, Decentralized Training and Execution (DTE) methods (Amato 2025) train a set of models, one for each agent in a decentralized manner, which are typically more scalable but do not use additional information during training (even when it is available). It is an open question which paradigm to use to maximize performance while maintaining scalability in the LLM collaboration problem. In this paper, we balance decentralized execution with centralized training using group-based Monte Carlo estimates. Experiments show the effectiveness of our approach on short-horizon tasks.

MAGRPO

We propose the Multi-Agent GRPO (MAGRPO) algorithm to jointly train LLM agents in MAS while maintaining decentralized execution.

Algorithm 1 shows the procedure of MAGRPO. Given a dataset $\mathcal{D}$ containing task information (e.g., the descriptions of coding problems), n LLMs that are optimized, each with a policy parameterized by θ_i and guided by a (shared) reward model R. In each episode, a task is sampled from the given dataset $\mathcal{D}$, which is used to construct initial observations

Algorithm 1: MAGRPO

Require: Dataset $\mathcal{D}$, n pretrained LLMs with policies $\{\pi_{\theta_1}, \cdots, \pi_{\theta_n}\}$, reward model R, generation group size G, learning rate α

for each episode **do**
  Sample a task $\sim \mathcal{D}$
  Initialize observations $o_{i,0}, \forall i \in \mathcal{I}$, according to the task, and $\mathbf{o}_0 = \{o_{1,0}, \cdots, o_{n,0}\}$
  $h^{\mathcal{G}}_{i,0} \leftarrow o_{i,0}, \forall i \in \mathcal{I}$, and $\mathbf{h}^{\mathcal{G}}_0 = \{h^{\mathcal{G}}_{1,0}, \cdots, h^{\mathcal{G}}_{n,0}\}$
  for turn $t = 0$ to $H-1$ **do**
    Generate a group of responses $a^{\mathcal{G}}_{i,t} \leftarrow \pi_{\theta_i}(\cdot|h^{\mathcal{G}}_{i,t})$, $\forall i \in \mathcal{I}$, where $h^{\mathcal{G}}_{i,t} = \{h^{(1)}_{i,t}, \cdots, h^{(G)}_{i,t}\}$, $a^{\mathcal{G}}_{i,t} = \{a^{(1)}_{i,t}, \cdots, a^{(G)}_{i,t}\}$, and $\mathbf{a}^{\mathcal{G}}_0 = \{a^{\mathcal{G}}_{1,t}, \cdots, a^{\mathcal{G}}_{n,t}\}$
    Obtain joint rewards $r^{\mathcal{G}}_t$ from system
    Receive new observations $o^{\mathcal{G}}_{i,t+1}$, and update history $h^{\mathcal{G}}_{i,t+1} \leftarrow \{h^{\mathcal{G}}_{i,t}, a^{\mathcal{G}}_{i,t}, o^{\mathcal{G}}_{i,t+1}\}, \forall i \in \mathcal{I}$
  end for
  for turn $t = H-1$ to 0 **do**
    Calculate return $R^{(g)}_t \leftarrow \sum_{\tau=t}^{H-1} r^{(g)}_\tau, \forall g \in \mathcal{G}$
    Estimate $\widehat{A}^{(g)}_t, \forall g \in \mathcal{G}$ according to Equation 1
    Calculate $J(\theta_i), \forall i \in \mathcal{I}$ according to Equation 2
    $\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J(\theta_i), \forall i \in \mathcal{I}$
  end for
end for
return $\boldsymbol{\pi_\theta} = \{\pi_{\theta_1}, \cdots, \pi_{\theta_n}\}$

$\mathbf{o}_0 = \{o_{1,0}, \cdots, o_{n,0}\}$ and histories $\mathbf{h}_0 = \{h_{1,0}, \cdots, h_{n,0}\}$. Taking inspiration from the single-agent GRPO algorithm (Shao et al. 2024), at each turn t, each agent takes action by generating a group of responses $a^{\mathcal{G}}_{i,t} = \{a^{(1)}_{i,t}, \cdots, a^{(G)}_{i,t}\}$ following its policy $\pi_i(\cdot|h^{\mathcal{G}}_{i,t})$ based on its observation-action history $h^{\mathcal{G}}_{i,t} = \{h^{(1)}_{i,t}, \cdots, h^{(G)}_{i,t}\}$. The actions of individual agents are aggregated to form a group of joint actions $\mathbf{a}^{\mathcal{G}}_t = \{a^{\mathcal{G}}_{0,t}, \cdots, a^{\mathcal{G}}_{n,t}\}$. The agents receive a group of joint rewards $r^{\mathcal{G}}_t$ for their responses $\mathbf{a}^{\mathcal{G}}_t$, which conditions on the accessible part of the state $R(\cdot|s^{\text{acc},\mathcal{G}}_t, \mathbf{a}^{\mathcal{G}}_t)$. The joint actions triggers the transition $T(\cdot|s^{\mathcal{G}}_t, \mathbf{a}^{\mathcal{G}}_t)$, where agents receive new observations $o^{\mathcal{G}}_{i,t+1} = \{o^{(1)}_{i,t+1}, \cdots, o^{(G)}_{i,t+1}\}$ and use them to construct histories $h^{\mathcal{G}}_{i,t+1} = \{h^{\mathcal{G}}_{i,t}, a^{\mathcal{G}}_{i,t}, o^{\mathcal{G}}_{i,t+1}\}$. This process continues until terminated at turn H or the task is completed.

We employ stochastic gradient descent to train agents at the end of each episode. Without explicit value models, estimating history-action values incurs high variance. To stabilize training, we estimate the expected return of the current history by averaging over a group of Monte Carlo samples of the joint return $R^{\mathcal{G}}_t = \{R^{(1)}_t, \cdots, R^{(G)}_t\}$. As a result, we are able to generate a centralized estimate (common in MARL) without a large value model. For each turn t, the advantage of each joint action in the group is calculated as,

$$\widehat{A}^{(g)}_t = R^{(g)}_t - \frac{1}{G}\sum_{g=1}^{G} R^{(g)}_t, \tag{1}$$

where $R^{(g)}_t = \sum_{\tau=t}^{H-1} r^{(g)}_\tau$. Inspired by GRPO (Shao et al. 2024; Liu et al. 2025b), and MAPPO (Yu et al. 2022), the centralized advantage values can be used to update policy π_i (parameterized by θ_i) for each agent i. MAGRPO does not have importance sampling and thereby epsilon clipping for simplicity, and the KL divergence coefficient is set to be 0 to encourage greater policy deviation from the base model,

$$J(\theta_i) = \mathbb{E}_{\mathbf{o}_0 \sim \mathcal{D}, \mathbf{h}^{\mathcal{G}} \sim \boldsymbol{\pi_\theta}}\left[\frac{1}{G}\sum_{g=1}^{G} \widehat{A}^{(g)}_t \log \pi_{\theta_i}(a^{(g)}_{i,t}|h^{\mathcal{G}}_{i,t})\right]. \tag{2}$$

Experiments

We evaluate MAGRPO on LLM writing and coding collaboration. Datasets, reward specifications, and additional results are provided in the Appendix.

Writing Collaboration

We explore LLM collaboration for article writing using MAGRPO across 2 classic tasks: summarization and expansion.

TLDR Summarization When reading a long article, readers often seek to quickly grasp its core ideas. If the topic is of interest, they may wish to delve deeper into specific details while still avoiding a complete reading through the full document. This calls for a summarization system to generate summaries at varying levels of detail. We frame this task using TLDR summarization as an illustrative example.

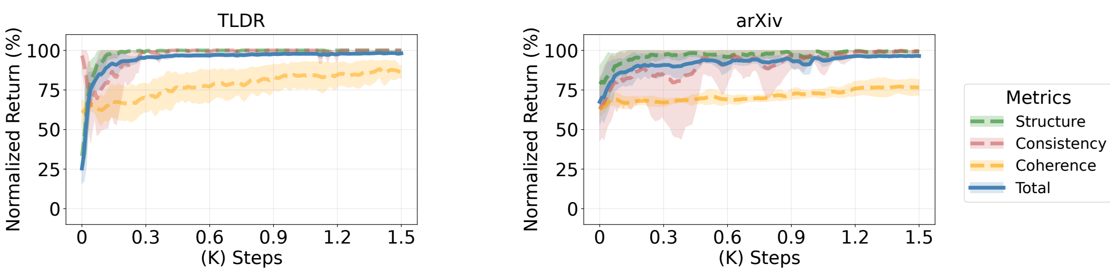
The TLDR dataset comprises unabridged Reddit posts in the `prompt` and concise summaries appended by the author in the `completion`. In our experiment, 2 *Qwen3-1.7B* agents independently summarize the `prompt` without using `completion`. The first agent functions as a core-idea (TLDR) generator, producing a concise paragraph, while the second agent serves as a detailed summarizer, providing more comprehensive information.

To quantify the summarization quality, we employ a relatively simple combination of 3 metrics. Structure measures the lengths and the length ratio of the 2 summaries, to ensure the TLDR is concise and the detailed summary is sufficiently long. Style consistency is assessed using the normalized Jaccard similarity coefficient, calculated as the ratio of the intersection size to the union size of unique words (or n-grams) between responses. A high style consistency reward typically indicates that the summarizers adopt similar stylistic patterns while avoiding identical wording. Logical coherence is quantified by counting the occurrences of transition words. Positive reward is given for using transition words, but the reward decreases logarithmically as more are used. These metrics are simple approximations of what more complex reward models may evaluate. Other (simpler or more complex) metrics or reward models could also be used. The total reward combines these metrics through a weighted summation. More details regarding our reward model and hyperparameters are provided in the Appendix.

arXiv Expansion Writing a long article typically requires contributions from multiple writers, each responsible for different sections. As a simple scenario, 2 agents can collabo-

Method	Dataset	Efficiency		Article Quality (%)			Return (%)
		Speed	Response Time	Structure	Consistency	Coherence	
Single Model	TLDR	64.1	6.6	43.8	97.6	52.8	36.7
	arXiv	65.4	6.5	51.2	87.2	**71.1**	44.9
Parallel Generation	TLDR	185.6	**2.1**	25.9	98.3	56.5	23.2
	arXiv	190.6	**2.1**	71.5	64.2	61.5	59.6
Sequential Generation	TLDR	98.7	4.3	33.5	98.5	64.5	21.7
	arXiv	85.8	4.3	92.4	**97.8**	64.3	87.7
One-Round Discussion	TLDR	100.4	4.3	35.9	**98.8**	60.8	22.3
	arXiv	95.4	4.3	84.6	71.8	66.0	76.6
MAGRPO (Ours)	TLDR	**202.3**	**2.1**	**98.7**	97.1	**78.5**	**94.5**
	arXiv	**193.8**	**2.1**	**97.9**	96.2	69.7	**93.1**

Table 1: Performance of MAGRPO against baselines on TLDR and arXiv. Speed (tokens/s) and response time (s) are measured on GeForce RTX 5090s. Results are normalized to the return scale. **Bolds** indicate the best performance on each dataset.



(a) TLDR summarization

(b) arXiv abstract expansion

Figure 2: Normalized returns on writing collaboration: (a) structural wellness (dashed green); (b) style consistency (dashed red); (c) coherence (dashed orange); (d) total rewards (solid blue). All returns are normalized to the return scale.

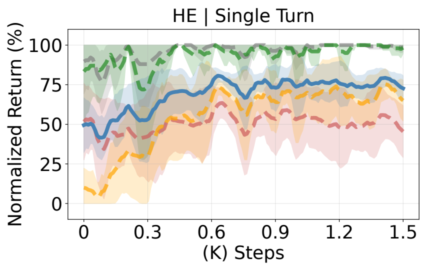
rate to generate introductions from the `abstract` of arXiv papers. The first agent outlines the research background and motivation, while the other presents the proposed methods and their experiments. The combined paragraphs should be coherent and consistent in style. Similar to the reward model in TLDR summarization, we employ the same evaluation metrics as proxies, with threshold hyperparameters specifically adjusted for this task.

Baselines We adopt a single model and 3 multi-agent methods as our baselines. To minimize the influence of prompts on our comparison, we keep the task description fixed and only add minimal instructions. In the single-agent baseline, we prompt with the article, the agent's role (summarizer or expanding writer), and specific format instructions. Naive concatenation builds on it by dividing the task into subtasks, assigning each agent a specific portion to complete in parallel without explicit communication. The sequential pipeline introduces one-way communication, allowing one agent to respond based on both the task description and the other agent's output. The one-round discussion baseline enables bidirectional communication: agents first receive the same prompts as in naive concatenation, then the prompts are augmented with the other's first-turn response in the second turn. All baseline methods operate without fine-tuning and depend solely on prompt-level interactions. Detailed prompts for each baseline are in the Appendix.

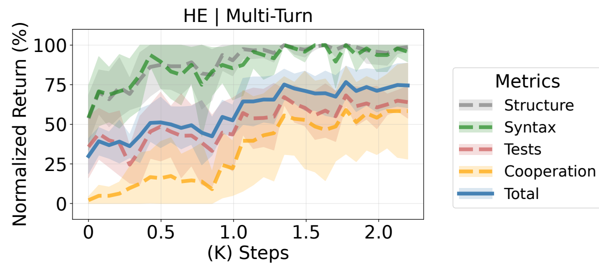
Results In this experiment, we apply MAGRPO to optimize the dual *Qwen3-1.7B* system in one turn. Figure 2a and Figure 2b show the evaluation results on TLDR and arXiv over 10 runs. The upward trend on all metric curves indicates that 2 agents gradually cooperate to generate coherent and consistent content with a well-organized structure. In the TLDR summarization, while the structure and logical coherence monotonically increase throughout training, the style consistency curves exhibit a decrease in the first 100 steps. This occurs as agents temporally diverge in styles to optimize other cooperative objectives, but their styles are gradually realigned and stabilized with sufficient training.

As shown in Table 1, MAGRPO is 3 times faster compared to the single *Qwen3-4B* model, which has a comparable number of parameters to our dual *Qwen3-1.7B* system. Despite receiving detailed instructions, *Qwen3-4B* fails to produce well-structured responses. A similar issue appears in TLDR summarization but not in arXiv expansion under multi-agent settings. This is because the outputs of homogeneous agents are naturally similar in length, which fortuitously aligns with the preference of the reward model.

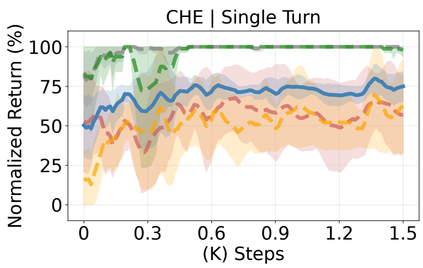
Among the multi-agent baselines, parallel generation is the only one that achieves a comparable speed to ours, but it fails to generate well-structured and coherent texts due to the lack of cooperation. Sequential generation and discussion-based approaches occasionally enhance coordination through specific prompts. However, they still under-



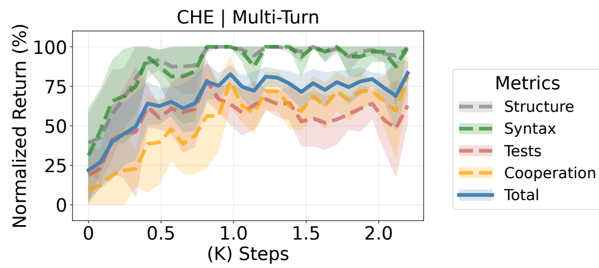
(a) Single-Turn MAGRPO on HE



(b) Multi-Turn MAGRPO on HE



(c) Single-Turn MAGRPO on CHE



(d) Multi-Turn MAGRPO on CHE

Figure 3: Normalized returns on coding collaboration: (a) structural wellness (dashed grey); (b) syntax correctness (dashed green); (c) Test score (dashed red); (d) cooperation rewards (dashed yellow); (e) total return (solid blue).

perform ours in efficiency and coherence, resulting in lower total return. The limited effectiveness of prompt-instructed coordination constrains their scalability to more complex scenarios involving large numbers of agents or extended multi-turn interactions (Estornell and Liu 2024).

Coding Collaboration

In large-scale software development, numerous developers collaborate to implement complex systems. Employing LLMs as developers is a promising direction, but coordinating them is challenging due to diverse cooperation schemes and complex failure modes. We simplify this task by using 2 *Qwen2.5-Coder-3B* agents to generate Python functions collaboratively. A helper agent produces auxiliary functions to support a main function generator, without any direct communication. The outputs from both agents, along with required libraries, are aggregated into complete code snippets.

HumanEval We evaluate MAGRPO on the HumanEval (HE) dataset, which contains 164 handwritten programming problems, each containing a natural language description (`prompt`), a function signature (`entry_point`), and a set of unit tests (`test`). To guide learning, we design a level-based reward model that prioritizes fundamental aspects of code generation. Structural integrity verifies the presence and correctness of both main and auxiliary function definitions; syntactic correctness ensures compliance with Python syntax; test pass rate assesses functional correctness based on the proportion of successfully passed unit tests; and a cooperation quality bonus is granted when the main function properly invokes and utilizes the auxiliary function. Rewards are accumulated only when all requirements at each preceding level are satisfied.

CoopHumanEval Some entries in HumanEval (HE) are not designed for coding collaboration; certain atomic operations (e.g., `strlen(string)`) can hardly be decomposed in a way that has meaningful cooperation. These noisy instances bias it towards invalid cooperation, such as merely wrapping the auxiliary function, thereby making training unstable. Thus, we construct a cooperative code generation dataset, CoopHumanEval (CHE), which comprises both original HE problems with cooperative potential (e.g., `prime_fib(n)`) and additional problems (e.g., `unique_digits(x)`). CHE problems are decomposable, where agents can explore effective cooperation schemes.

Baselines We adopt the fixed and fine-tuned single model, and 3 multi-agent methods on fixed base models, as our baselines. In the single-agent setting, the *Qwen2.5-Coder-7B* model generates a Python function based on the problem description in `prompt`. We also fine-tune this model on the training set to adapt it to this task. In the multi-agent setting, 2 *Qwen2.5-Coder-3B* models serve as agents: one generates a helper function, and the other produces the main. To minimize the influence of prompts on our comparison, we keep the problem description fixed and only add minimal coordination instructions. In the naive concatenation, agents are informed of their roles and generate outputs in parallel without communication. The sequential pipeline allows the main agent to respond based on the other's output. In the one-round discussion, agents first receive the same prompts as naive concatenation, then the prompts are augmented with the other's last-turn response in the subsequent turns.

Results We optimize the interaction between 2 agents using single-turn and multi-turn MAGRPO. In the multi-turn setting, problem descriptions and the agents' initial re-

Method	Dataset	Efficiency		Code Quality (%)				Return (%)
		Speed	Response Time	Structure	Syntax	Tests	Cooperation	
Single Model	HE	73.1	1.6	**100.0**	**100.0**	64.8	–	–
	CHE	65.5	1.4	**100.0**	**100.0**	63.4	–	–
Fine-Tuned Single Model	HE	72.2	1.6	**100.0**	99.2	65.6	–	–
	CHE	60.6	1.4	**100.0**	**100.0**	66.7	–	–
Naive Concatenation	HE	**194.9**	**1.1**	96.1	90.6	42.5	22.7	53.9
	CHE	189.4	**1.1**	97.5	95.0	40.1	24.0	54.3
Sequential Pipeline	HE	99.6	2.2	98.4	96.5	56.4	35.1	63.1
	CHE	97.4	2.0	97.5	96.3	55.2	35.2	62.5
One-Round Discussion	HE	82.5	2.8	98.1	94.8	41.2	30.2	57.5
	CHE	78.3	2.8	97.5	96.3	41.9	34.8	59.5
Single-Turn MAGRPO (Ours)	HE	190.0	1.5	**100.0**	97.8	61.6	83.4	83.7
	CHE	**192.4**	1.5	98.8	97.5	71.2	83.7	86.0
Multi-Turn MAGRPO (Ours)	HE	96.5	2.7	99.7	96.9	**68.4**	**84.9**	**86.7**
	CHE	97.1	2.5	98.6	98.2	**75.0**	**86.3**	**88.5**

Table 2: Performance comparison of MAGRPO against baselines on HE and CHE. Speed (tokens/s) and response time (s) are recorded on GeForce RTX 5090s. Results are normalized to the return scale and averaged over 10 runs; rewards are level-based. **Bolds** indicate the best performance of each metric on each dataset.

sponses are provided to static checking (*AST*) and dynamic execution models, which report errors for each agent. Ablations of external tools used, including self-evolving and expert guidance, are provided in the Appendix.

Figure 3a and Figure 3b show the normalized return of MAGRPO on HE over 10 runs. Single-turn MAGRPO training improves the syntactical correctness and develops valid cooperation, while the test pass rate does not show much progress. As for the multi-turning training, agents are initially overwhelmed by the external model's feedback, resulting in even lower initial returns. They gradually adopt the error signals and improve their returns. However, the test pass rate shows no significant improvement over the single-agent model, due to noisy entries in the dataset and hence unreliable feedback. This reflects the complexity and delicacy of decentralized coder coordination, where the main agent must accurately infer the functionality of auxiliary modules and trust their correctness without communication.

The performance of single-turn and multi-turn MAGRPO on the CHE dataset is shown in Figures 3c and 3d. Results show that MAGRPO achieves higher overall returns and lower variance when trained on CHE over HE. In the multi-turn setting, although agents initially struggle to interpret the feedback, like training on HE, the returns gradually improve and eventually surpass those in the single-turn training. This demonstrates that, when trained on a dataset with well-defined cooperative structures, agents can learn to utilize error messages to improve their response quality.

Table 2 presents a performance comparison between MAGRPO and baselines on both HE and CHE. By GRPO fine-tuning, the performance of *Qwen2.5-Coder-7B* model only improves slightly as the logic of test problems differs substantially. Compared to a single model, the naive concatenation method has lower test pass rates, as the main agent may generate code based on incorrect assumptions about auxiliary functionality. In the sequential pipeline method, the main agent can provide a backup for the auxiliary function when it identifies potential vulnerabilities, thereby improving the test pass rate. However, this comes at the cost of slower inference speed. Although the one-round discussion method involves more communication between agents, its effectiveness remains limited because the agents' mutual adaptations to each other's last responses can become misaligned. MAGRPO outperforms all baselines on both CHE and HE by facilitating effective cooperation and leveraging feedback from the external model. Additional results, including pass@k, are presented in the Appendix.

Cooperation Schemes MAGRPO identifies diverse cooperation schemes. In some cases, the auxiliary function handles the core logic, while the main agent adds backup logic or decorations to improve the overall solution. Alternatively, the main agent may act as a coordinator, decomposing the problem and assigning subtasks to the auxiliary agent. The auxiliary function may serve as a strategy filter, guiding the main agent to generate code for specific cases. While coordinator and strategy-filter schemes can improve inference efficiency, they are more prone to syntax and logical errors. With limited cooperation-oriented training data, the main agent typically resorts to more conservative roles, i.e., fallback or decoration. These cooperation schemes emerge during training under a relatively simple joint reward. More refined design patterns will likely be found when training agents to develop large-scale coding projects.

Conclusion

In this paper, we model LLM collaboration as a cooperative MARL problem and formalize it as a Dec-POMDP. We propose the MAGRPO algorithm to optimize agent cooperation through shared rewards. Our experiments in coding and writing collaboration show that MAGRPO enables agents to efficiently generate high-quality responses via effective collaboration. Our work encourages future exploration of MARL-based methods for scalable and robust LLM collaboration.

Acknowledgments

This work was partially funded by NSF grants #2044993 and #2409351. It used Delta and DeltaAI computing resources at the National Center for Supercomputing Applications through allocation CIS250443 and CIS250554 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support program, which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296.

We thank Tianle Chen for improving our MARL training framework for LLM collaboration, CoMLRL, and members of the Lab for Learning and Planning in Robotics (LLPR) for the valuable discussion. We thank Gregory Bauer and Brett Bode for helping us resolve job failure issues.

References

Achiam, J.; Adler, S.; Agarwal, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Amato, C. 2024. An Introduction to Centralized Training for Decentralized Execution in Cooperative Multi-Agent Reinforcement Learning. arXiv 2409.03052. arXiv:2409.03052.

Amato, C. 2025. An Initial Introduction to Cooperative Multi-Agent Reinforcement Learning. arXiv:2405.06161.

Anil, R.; Borgeaud, S.; Alayrac, J.-B.; et al. 2025. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805.

Anthropic. 2024. How We Built a Multi-Agent Research System.

Berner, C.; Brockman, G.; Chan, B.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Cemri, M.; Pan, M. Z.; Yang, S.; Agrawal, L. A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; Zaharia, M.; Gonzalez, J. E.; and Stoica, I. 2025. Why Do Multi-Agent LLM Systems Fail?

Chan, C.-M.; Chen, W.; Su, Y.; Yu, J.; Xue, W.; Zhang, S.; Fu, J.; and Liu, Z. 2023. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate.

Chen, Y.; Yan, L.; Sun, W.; Ma, X.; Zhang, Y.; Wang, S.; Yin, D.; Yang, Y.; and Mao, J. 2025. Improving Retrieval-Augmented Generation through Multi-Agent Reinforcement Learning. In *Proceedings of the 39th Conference on Neural Information Processing Systems*.

Du, Y.; Li, S.; Torralba, A.; Tenenbaum, J. B.; and Mordatch, I. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. arXiv:2305.14325.

Estornell, A.; and Liu, Y. 2024. Multi-LLM Debate: Framework, Principals, and Interventions. In *NeurIPS*.

Foerster, J.; Chen, R.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018a. Learning with Opponent-Learning Awareness. In *AAMAS 2018: Proceedings of the Seventeenth International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018b. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; et al. 2024. The Llama 3 Herd of Models.

Guo, D.; Yang, D.; Zhang, H.; et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081): 633–638.

Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Liang, T.; He, Z.; Jiao, W.; Wang, X.; Wang, Y.; Wang, R.; Yang, Y.; Shi, S.; and Tu, Z. 2024. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. arXiv:2305.19118.

Lifshitz, S.; McIlraith, S. A.; and Du, Y. 2025. Multi-Agent Verification: Scaling Test-Time Compute with Multiple Verifiers. arXiv:2502.20379.

Liu, B.; Guertler, L.; Yu, S.; Liu, Z.; Qi, P.; Balcells, D.; Liu, M.; Tan, C.; Shi, W.; Lin, M.; et al. 2025a. SPIRAL: Self-Play on Zero-Sum Games Incentivizes Reasoning via Multi-Agent Multi-Turn Reinforcement Learning. *arXiv preprint arXiv:2506.24119*.

Liu, S.; Wang, Y.; Chen, X.; Fu, Y.; and Di, X. 2022. SMART-eFlo: An integrated SUMO-gym framework for multi-agent reinforcement learning in electric fleet management problem. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 3026–3031. IEEE.

Liu, Z.; Chen, C.; Li, W.; Qi, P.; Pang, T.; Du, C.; Lee, W. S.; and Lin, M. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*.

Lyu, X.; Baisero, A.; Xiao, Y.; Daley, B.; and Amato, C. 2023. On Centralized Critics in Multi-Agent Reinforcement Learning. *Journal of Artificial Intelligence Research*, 77: 295–354.

Ma, H.; Hu, T.; Pu, Z.; Boyin, L.; Ai, X.; Liang, Y.; and Chen, M. 2024. Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 15497–15525.

Marchesini, E.; Baisero, A.; Bathi, R.; and Amato, C. 2025. On Stateful Value Factorization in Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

NVIDIA. 2024. Introduction to LLM Agents.

Oliehoek, F. A.; and Amato, C. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15174–15186.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36: 53728–53741.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research*, 21(178): 1–51.

Samvelyan, M.; Rashid, T.; De Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*.

Sarkar, B.; Xia, W.; Liu, C. K.; and Sadigh, D. 2025. Training language models for social deduction with multi-agent reinforcement learning. *arXiv preprint arXiv:2502.06060*.

Shao, Z.; Wang, P.; Zhu, Q.; et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Shoham, Y.; and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press.

Skreta, M.; Yoshikawa, N.; Arellano-Rubach, S.; Ji, Z.; Kristensen, L. B.; Darvish, K.; Aspuru-Guzik, A.; Shkurti, F.; and Garg, A. 2023. Errors are Useful Prompts: Instruction Guided Task Programming with Verifier-Assisted Iterative Prompting. arXiv:2303.14100.

Slumbers, O.; Mguni, D. H.; Shao, K.; and Wang, J. 2024. Leveraging Large Language Models for Optimised Coordination in Textual Multi-Agent Reinforcement Learning.

Stone, P.; and Veloso, M. 2000. Multiagent Systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3): 345–383.

Subramaniam, V.; Du, Y.; Tenenbaum, J. B.; Torralba, A.; Li, S.; and Mordatch, I. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*.

Tan, M. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Proceedings of the Tenth International Conference on Machine Learning*, 330–337. San Francisco, CA, USA: Morgan Kaufmann.

Van der Hoek, W.; and Wooldridge, M. 2008. Multi-agent systems. *Foundations of Artificial Intelligence*, 3: 887–928.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *Proceedings of the International Conference on Learning Representations*.

Weiss, G. 1999. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.

Wiering, M. A. 2000. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 1151–1158. Stanford, CA, USA: Morgan Kaufmann.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation.

Xiao, Y.; Tan, W.; Hoffman, J.; Xia, T.; and Amato, C. 2025. Asynchronous Multi-Agent Deep Reinforcement Learning under Partial Observability. *International Journal of Robotics Research*, 35(14): 1760–1778.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Thirty-Sixth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Zhang, K.; Liu, R.; Zhu, X.; Tian, K.; Zeng, S.; Jia, G.; Fan, Y.; Lv, X.; Zuo, Y.; Jiang, C.; Liu, Z.; Wang, J.; Wang, Y.; Zhao, R.; Hua, E.; Wang, Y.; Wang, S.; Gao, J.; Long, X.; Sun, Y.; Ma, Z.; Cui, G.; Bai, L.; Ding, N.; Qi, B.; and Zhou, B. 2025. MARTI: A Framework for Multi-Agent LLM Systems Reinforced Training and Inference.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.