Improving Multi-Agent Collaboration in Large Language Models

Sudhanva Devanathan, Aaron Tamte, Sebastian Thomas, Lile Zhang

Khoury College of Computer Sciences, Northeastern University
devanathan.s@northeastern.edu, tamte.aa@northeastern.edu
thomas.seba@northeastern.edu, zhang.lile@northeastern.edu

Abstract

Large Language Models (LLMs) have revolutionized decision-making, reasoning, and problem-solving, offering significant advancements in computational agents across various domains. While LLMs excel in individual tasks, their potential in multi-agent systems (MAS)—where collaboration among agents is critical—remains under-explored.

This project aims to address this gap by achieving two key objectives: (1) developing multi-agent-specific benchmarks to assess the collaborative capabilities of LLMs, and (2) investigating whether collaboration can be enhanced as a trainable dimension through fine-tuning open-source LLMs. Drawing inspiration from prior studies, we propose customizing a grid environment with diverse tasks requiring agent collaboration, establishing communication protocols for inter-agent interactions, and designing a robust benchmark to evaluate performance. By fine-tuning LLMs, we seek to enhance their ability to perform collaborative tasks effectively.

Code: `https://github.com/tamteaa/CS5100B-Project.git`

Fine-tuned model: `https://huggingface.co/aarontamte/LLama3.1-8B-cooperative-agent-finetune`.

1 Introduction

Adopting Large Language Models (LLMs) as the core component in computational agents has opened new horizons for decision-making, reasoning, and problem-solving across various domains(Xi et al., 2023). While LLMs like GPT-4 have demonstrated various emergent abilities and excellence in individual tasks, their performance in multi-agent systems (MAS)—where collaboration among agents is essential—remains a new study area.

Multi-agent systems involve multiple agents interacting within a shared environment to achieve individual or collective goals. Leveraging LLMs within multi-agent frameworks can enhance agents' abilities to communicate, reason, and adapt to dynamic environments. However, integrating LLMs into multi-agent systems raises questions about the assessment of collaborative capabilities and the potential to train models specifically for improved collaboration.

Recent advancements in LLMs have revealed a gap in how to evaluate multi-agent coordination tasks (Chen et al., 2023). Existing research primarily emphasizes enhancing reasoning or task-solving within single-agent contexts rather than addressing scenarios where multiple agents must coordinate actions to achieve a task that cannot be completed by any single agent alone. For example, solving problems in environments that require coordinated actions from multiple agents – where each agent must perform specific and interdependent tasks to succeed – remains an under-explored area in the field of LLM-based agents.

This project aims to fill this gap by pursuing two key objectives:

- Developing multi-agent-specific benchmarks to assess an LLM's collaborative capabilities rigorously.
- Investigating whether collaboration can be considered a trainable dimension by fine-tuning an open-source large language model.

Through this project, we aim to deepen the understanding of LLMs' capacity to engage in both intra-agent and inter-agent communication and coordination.

2 Related Work

Due to their versatile capabilities, LLMs have become the foundation for constructing LLM-based agents. Some researchers (Andreas, 2022) suggest that language models can, in a narrow sense, serve as models of agents, while others (Xi et al., 2023) explore their suitability and applications in settings such as single-agent, multi-agent, and human-agent interactions.

Recent research has evaluated the performance of LLM-based agents across diverse environments. In (Agashe et al., 2024), the authors introduce an LLM Coordination Benchmark to analyze multi-agent coordination and propose a Cognitive Architecture for Coordination (CAC), enabling effective participation in coordination games like Overcooked-AI. They emphasize the importance of environment comprehension and Theory of Mind reasoning for success. Meanwhile, (Park et al., 2023) presents generative agents simulating human behavior, equipped with memory, reflection, and planning capabilities, demonstrating emergent social behaviors in a simulated town. Finally, (Wang et al., 2023) highlights LLM applications in open-ended learning environments like Minecraft, where embodied agents exhibit real-time learning and goal-driven exploration.

3 Methodology

Inspired by (Agashe et al., 2024; Park et al., 2023; Wang et al., 2023), we aim to customize a grid environment with diverse tasks requiring agent collaboration, establish a communication protocol for agents to communicate with one another, develop a multi-agent benchmark to assess the collaborative abilities of LLM-based agents and fine-tune the LLM to enhance their collaborative performance with generated training data.

3.1 Environment

Before exploring the potential of multi-agent collaboration in Large Language Models (LLMs), the first step is to design a structured environment to effectively evaluate their collaboration capabilities. For this purpose, we have chosen a grid-world-like environment, which we classify into two types: single-agent and multi-agent.

Single-agent environments serve as a base case where a single agent attempts specific tasks, while multi-agent environments involve multiple agents working collaboratively to achieve a common goal. The details of these environments and their respective action spaces are discussed below.

In a single-agent environment, a single agent operates to accomplish a defined task. We have devised two primary tasks for this setup

- Navigation: The agent navigates from a start cell to a goal cell.
- Pick up items: The agent picks up an item from a specified cell and navigates to the goal cell.

In a multi-agent environment, multiple agents collaborate to achieve a shared objective. These agents take turns executing an action. The tasks designed for this environment are as follows:

- Navigation - Multiple agents navigate from their respective start cells to a goal cell, taking turns for each action.
- Alphabetical order - multiple agents must complete a specific task in the alphabetical order of their names.
- Pick up items - multiple agents work together to pick up items at specified locations and move to a goal cell.
- Pick up items with permissions - In this task, each item can only be picked up by a specific agent with the required permissions. The agents must coordinate effectively to accomplish the goal.

3.1.1 Action Space

Table 1 shows the action space we have defined and it is common to all the variations of the grid-world environment mentioned above.

Action	Description
north	Agent moves one cell up.
south	Agent moves one cell down.
west	Agent moves one cell to the left.
east	Agent moves one cell to the right.
pick	Agent picks up an item.
drop	Agent drops an item.

Table 1: Actions and their descriptions

3.1.2 Environment Scoring

The scoring system is designed to evaluate agent performance across different tasks and environments. Each task has specific success criteria, and scores are computed based on task completion and agent cooperation. Table 2 and 3 provide a brief explanation of the scoring approach for each task.

3.2 Agents

Our agents support various API backends, including OpenAI, Groq, and Together AI. They can interact with the environment and communicate with other agents through the design outlined in the following section.

3.2.1 Architecture

The architecture of our agents is designed to build on existing best practices. Agents utilize reflection and memory mechanisms to analyze past actions, retain information, and adapt throughout episodes. Output is standardized to a JSON schema, where the fields include reflection, rationale, actionName, actionParameters, message, and addMemory, in that order. This modular design allows the agents in the simulation to engage in communication with other agents while simultaneously allowing the agent to reflect on past mistakes and take actions immediately.

3.2.2 Communication

Most LLMs have three roles: system, user/human, and assistant. They have different purposes and

Task	Task Description	Score	Score Formula
Navigation	The agent must navigate from a start cell to a specified target position.	A perfect score (100 points) is awarded if the agent reaches the target position.	100 if the agent reaches the target, otherwise 0.
Pick-Up Items	The agent must pick up an item from a specific location and deliver it to a target position.	The score is calculated based on the number of correctly delivered items.	(Number of Correct Deliveries / Total Items) × 100.

Table 2: Single-Agent Environment Tasks with Descriptions, Scores, and Formulas

Task	Task Description	Score	Score Formula
Multi-Agent Navigation	The agents must navigate from a start cell to their designated corner cells on the grid.	A corner is said to be occupied if the designated agent is present there.	(Number of Occupied Corners / 4) × 100.
Pick-Up Items	Agents must pick up items and deliver them to the correct target positions.	The score is based on how many target cells contain items.	(Correct Deliveries / Total Targets) × 100.
Random Points Navigation	Agents must occupy specified random target cells.	The score is based on the number of targets occupied by any agent.	(Occupied Targets / Total Targets) × 100.
Pick-Up Items with Permissions	Items are restricted to specific agents (permissions). Only authorized agents can pick up and deliver them.	The score is calculated based on correct deliveries to target cells.	(Correct Deliveries / Total Targets) × 100.
Alphabetical-Order Navigation	Agents are sorted alphabetically by their names. Each agent must occupy a specific position based on this order.	The score is based on how many agents are correctly positioned.	Score = (Correctly Positioned Agents / Total Agents) × 100.

Table 3: Multi-Agent Environment Tasks with Descriptions, Scores, and Formulas

usually, the interaction starts with the system and follows by alternating user-assistant pairs. The system role can set high-level instructions to guide LLMs' behavior. In our design, using the system role, we offer agents some required information and guidelines such as the environment setting, the goal of the task, and the rules to follow. The user role represents the user's input including the commands or questions. We use the user role to inform the agent of its current observation space and remind the agent of memory, goal, team score (how many agents complete the task), important notes to follow, etc. The assistant role is LLM's response to the user's input. From LLM's response, we can extract the agent's intended action and input it into the step function to interact with the environment.

Note that during the whole process, every agent only knows their current position from us but has no knowledge of other agents' positions or plans. We set an inbox-message mechanism for agents to communicate and discuss with others. Specifically, we add an inbox section in the user role's input and update the system instructions to allow the agent to send messages in the required format. Therefore, we can extract the message from LLM's response and update the inbox section with the message. In our design, the inbox section for each agent only shows the latest message sent from all other agents. This mechanism enhances the communication and collaboration between agents. From observation, agents send messages to discuss task assignments, share their goals, resolve conflict, confirm task completeness, etc.

4 Experiment

4.1 Data Generation

For each task, we simulate several episodes and record all user inputs and agent responses. The agent responses include reflection, rationale, action, message, and memory. Since user inputs and

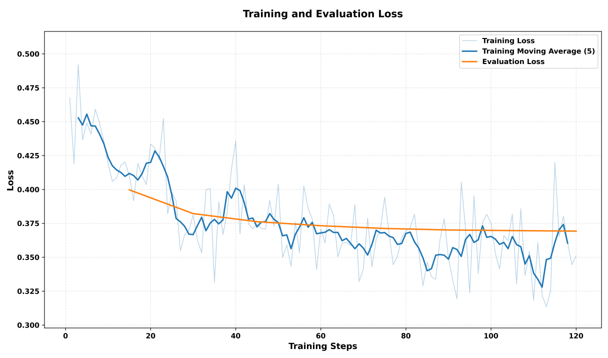

Figure 1: Training loss over epochs during fine-tuning. The loss decreased from approximately 0.46 to 0.35, demonstrating stable convergence.

agent responses are paired, we score the agent responses using the team score information extracted from the user inputs. We then prune the data to retain only successful simulations with a score of 50 or higher. The data is formatted to match the format required by the fine-tuning API and split into training and validation datasets. As a result, we have 5.3M tokens of training data and 1.6M tokens of validation data.

4.2 Fine-tuning

To enhance the collaborative performance of our LLM-based agents, we fine-tuned the Meta-Llama-3.1-8B-Instruct model using Low-Rank Adaptation (LoRA). The fine-tuning process was conducted over six epochs with a batch size of 32. Key parameters included a learning rate of 5×10^{-5} with a warmup ratio of 0.15, a weight decay of 0.03, and a maximum gradient norm of 0.3. LoRA-specific settings—such as a rank of 8, an alpha of 16, and a dropout rate of 0.2—were applied to all linear layers. The model was optimized using a cross-entropy loss function, reducing the loss from approximately 0.46 to 0.35 over the course of training. This approach ensured stable convergence while enhancing the model's ability to collaborate effectively in multi-agent environments.

5 Results

The benchmarking results were generated by evaluating three models: Llama 3.1 8B, Llama 3.1 70B and the fine-tuned version of Llama 3.1 8B. The fine-tuning was done across various grid-based environments, each designed to test specific capabilities such as navigation, multi-agent coordination, item manipulation, and task sequencing. Each model was deployed in simulated environments and performance was measured using metrics such as steps taken and messages sent with the emphasis being on the average score. The models were tested multiple times in each environment to ensure statistical reliability and the fine-tuned version was evaluated to assess the impact of environment-specific training.

5.1 Analysis

Figure 2 compares how each model performed across different environments. The performance of each model is analyzed in detail for every environment below:

1. Single Agent Navigation: In this relatively straightforward environment, Llama 3.1 8B achieved a decent score of around 60 indicating a moderate understanding in basic navigation tasks. Llama 3.1 70B significantly outperformed it with a score nearing 90, leveraging its larger model size and greater capacity for understanding the task dynamics. However, the fine-tuned Llama 3.1 8B demonstrated the strongest performance, achieving a perfect score of 100. This underscores the value of fine-tuning for specialized tasks, even when starting with a smaller model.

2. Corner Multi-Agent Navigation: This environment involves coordinating multiple agents to navigate to specific corners, increasing the task's complexity. Llama 3.1 8B struggled with a low score of around 30, likely due to insufficient knowledge in managing multi-agent strategies. The 70B model, with its larger size, performed better, scoring around 65. Interestingly, the fine-tuned 8B model achieved a comparable score of about 70, highlighting that task-specific fine-tuning can rival the performance of larger models in multi-agent coordination.

3. Alphabetical Line Task: In this sequential task, agents must align alphabetically, which requires strong sequential reasoning and order maintenance. Llama 3.1 8B performed poorly, with an average score of 10, suggesting a lack of capability in handling such ordered tasks. The 70B model showed significant improvement, scoring around 60, benefiting from its enhanced reasoning abilities. The fine-tuned 8B model further improved, scoring around 70, demonstrating that fine-tuning enhanced

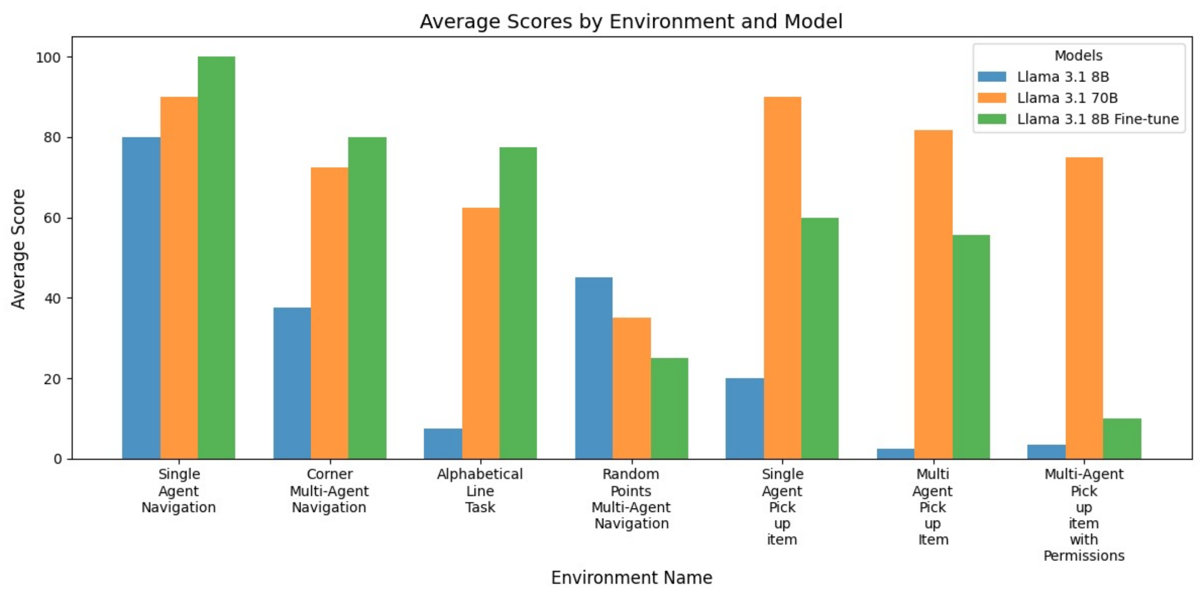

Figure 2: Average Scores by environment demonstrating how every model has performed in every environment.

its ability to grasp and execute complex sequential logic.

4. Random Points Multi-Agent Navigation: This environment introduces randomness, requiring agents to coordinate their navigation to dynamic, randomly assigned points. Llama 3.1 8B scored moderately at 50, showing some capability but struggling with dynamic agent coordination. The 70B model slightly improved, scoring around 60, while the fine-tuned 8B model matched this performance with a similar score. The comparable performance of the fine-tuned 8B suggests that model adaptation can significantly enhance performance in dynamic multi-agent settings.

5. Single Agent Pick Up Item: In this environment, a single agent must pick up an item and complete the task. Llama 3.1 8B struggled, scoring around 30, indicating difficulties in object manipulation tasks. The 70B model performed exceptionally well, achieving a near-perfect score of 100, showcasing its strength in single-agent item handling. The fine-tuned 8B model also improved significantly, scoring around 85, bridging much of the gap with the 70B model, although not quite reaching its peak performance. This suggests that fine-tuning improves item-handling efficiency but may still lag behind larger models in precision.

6. Multi-Agent Pick Up Item: Multiple agents must coordinate to pick up items, increasing the task complexity. Llama 3.1 8B struggled significantly, scoring only around 20, highlighting its limited capability in complex coordination. The 70B model excelled, scoring close to 90, showing a strong ability to handle multi-agent item manipulation. The fine-tuned 8B model also performed well, scoring around 85, further emphasizing the effectiveness of task-specific fine-tuning in improving multi-agent coordination performance.

7. Multi-Agent Pick Up Item with Permissions: This environment adds another layer of complexity by introducing permissions, requiring agents to not only coordinate but also manage permissions for item handling. Llama 3.1 8B performed poorly, scoring around 10, indicating significant difficulty in managing permission-based coordination. The 70B model excelled again, with a score of 90, showing its superior capacity to handle constrained, rule-driven environments. The fine-tuned 8B model followed closely, scoring 85, demonstrating that fine-tuning can effectively prepare smaller models to handle complex environments with additional constraints.

6 Conclusion

The models showed mixed performance across environments, highlighting areas for improvement.

Addressing these gaps can boost task completion and enhance multi-agent collaboration. Key areas for improvement include:

1. Improve Communication: Simplifying and standardizing message protocols can improve agent communication. Ensuring that agents maintain a consistent view of the environment in all states will improve coordination and decision-making.

2. Better Role Assignment: Tasks should be dynamically assigned based on the capabilities of the agent. Breaking down complex tasks into smaller, clearer sub-tasks will make them more manageable and increase task efficiency.

3. Introduce Rewards: Developing a more robust rewards framework can promote teamwork and reinforce desired behaviors. Probably providing immediate feedback will help agents learn faster and adapt more effectively to changing environments.

4. Focused Fine-Tuning: Fine-tuning models with data from collaborative task scenarios specific to these environments can significantly improve performance. This approach will enable models to better handle the nuances of multi-agent cooperation.

Through developing task-specific benchmarks and refining models through fine-tuning, this research has made several key contributions to understanding and improving multi-agent collaboration in Large Language Models. Our result demonstrate that while larger models like Llama 3.1 70B generally perform better in complex collaborative tasks, targeted fine-tuning of smaller models can achieve comparable performance in many scenarios. Additionally, our findings show that collaborative ability should be considered a fundamental dimension in both benchmarking and training of language models, rather than treating it as an emergent property. By systematically evaluating and enhancing collaborative abilities, we can develop more effective multi-agent systems that better serve real-world use cases requiring coordinated action and decision-making.

7 Team Contributions

Aaron developed the agent class, fine-tuned the base model, and developed the GUI. Sudhanva developed benchmarking modules and the Action class to capture, validate, and manage agent actions effectively. Lile worked on the database and the pick-up items task. Sebastian developed the simulator and environment wrapper class and worked on the environment configuration yaml file parser.

*ChatGPT was used for various grammatical checks and summarization, but no original content was generated via ChatGPT.

References

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. 2024. Llm-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models.

Jacob Andreas. 2022. Language models as agent models.

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. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey.