AgentDropout: Dynamic Agent Elimination for Token-Efficient and High-Performance LLM-Based Multi-Agent Collaboration

Zhexuan Wang[1] **Yutong Wang**[1] **Xuebo Liu**[1*] **Liang Ding**[2]
Miao Zhang[3] **Jie Liu**[4] **Min Zhang**[1]

[1]Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen, China
[2]The University of Sydney, Sydney, Australia
[3]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[4]State Key Lab of Smart Farm Technologies and Systems, Harbin Institute of Technology, Harbin, China
{wangzhexuan,wangyutong}@stu.hit.edu.cn, liangding.liam@gmail.com
{liuxuebo,zhangmiao,jieliu,zhangmin2021}@hit.edu.cn

Abstract

Multi-agent systems (MAS) based on large language models (LLMs) have demonstrated significant potential in collaborative problem-solving. However, they still face substantial challenges of low communication efficiency and suboptimal task performance, making the careful design of the agents' communication topologies particularly important. Inspired by the management theory that roles in an efficient team are often dynamically adjusted, we propose **AgentDropout**, which identifies redundant agents and communication across different communication rounds by optimizing the adjacency matrices of the communication graphs and eliminates them to enhance both token efficiency and task performance. Compared to state-of-the-art methods, AgentDropout achieves an average reduction of 21.6% in prompt token consumption and 18.4% in completion token consumption, along with a performance improvement of 1.14 on the tasks. Furthermore, the extended experiments demonstrate that AgentDropout achieves notable domain transferability and structure robustness, revealing its reliability and effectiveness. We release our code at https://github.com/wangzx1219/AgentDropout.

1 Introduction

In recent years, LLM-based (Achiam et al., 2023; Team et al., 2024) agent technologies have achieved robust performance across a wide range of tasks. Prior works have discovered that LLM-based agents (Liang et al., 2024; Wang et al., 2025a; Du et al., 2024; Wang et al., 2024b; Zhuge et al., 2024; Wang et al., 2024a) can emulate human-like collaboration through mutual communication, leading to enhanced performance in tasks and forming MAS that resemble cooperative human teams.

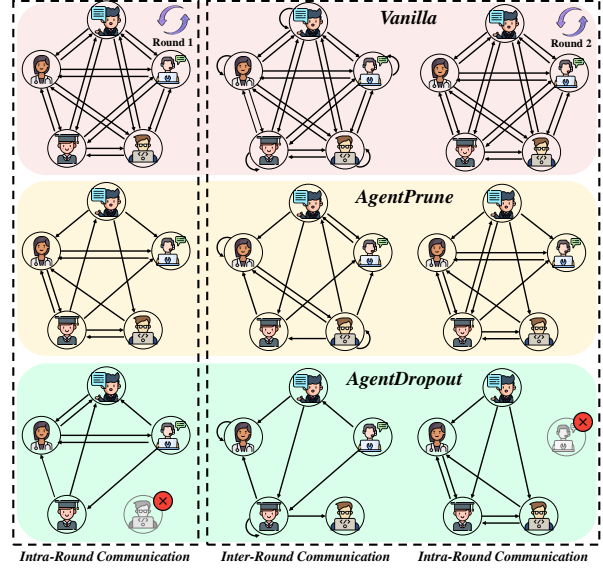However, MAS often suffers from high token consumption and inefficiency (Li et al., 2024). This

Figure 1: An overview of AgentDropout, in comparison with vanilla MAS and AgentPrune (Zhang et al., 2025), highlighting its dynamic adjustments of participant roles in each discussion to enhance cooperation effectiveness and efficiency.

is primarily due to the frequent generation and communication among multiple agents, which incurs substantial token overhead. Existing methods mainly apply simple agent profile evolution (Chen et al., 2024a) or memory evolution (Wang et al., 2025b; Qian et al., 2024a,b; Gao et al., 2024), which fails to address the core problem of multi-agent interaction optimization. To address this issue, Zhuge et al. (2024) proposed the pruning of inter-graph communication edges, while Zhang et al. (2025) defined communication redundancy and introduced a pruning method AgentPrune within MAS. Specifically, existing methods employ trainable communication graph masks to iteratively identify critical message exchanges and prune redundant edges, thereby generating a sparser and more efficient communication graph. However, constrained by the unchanged roles participating in the discussion in each round in MAS, the existing method applies a uniform pruning strategy across

---

* Corresponding Author

multiple communication rounds, which may limit its effectiveness in improving system efficiency and performance.

Drawing an analogy between MAS and human teams, it is evident that dynamic role allocation and team collaboration are pivotal factors in enhancing decision-making efficiency and overall performance in high-performing teams. Locke (2011) have pointed out that an effective team must adaptively adjust the roles and responsibilities of its members based on task requirements, Kozlowski and Ilgen (2006) further emphasizing the critical role of such dynamic adjustments in improving team decision-making efficacy. Inspired by these insights, we propose AgentDropout, as shown in figure 1. Specifically, AgentDropout iteratively updates the communication topology using a small amount of data, simulating the behavior of dynamic role assignment during multi-round discussions. This process consists of two steps: (1) **Node Dropout**: Similar to how team members assume different responsibilities at various stages, we train the edge weights and compute the degree of nodes in this step. By selectively retaining or removing certain nodes across different rounds, the flexibility and efficiency of the entire system are enhanced. (2) **Edge Dropout**: We optimize the communication and collaboration mechanisms within teams by training edge weights and pruning redundant or low-contribution edges to refine the connections between nodes. This approach not only reduces the computational complexity but also enhances the effectiveness of information propagation.

Experiments conducted on a variety of tasks, including reasoning, mathematics, and code generation, demonstrate that AgentDropout outperforms different kinds of MAS and AgentPrune in terms of performance and token efficiency. Our main contributions are as follows:

- We observe that in MAS with both intra- and inter-round characteristics, involving different roles in distinct rounds of discussion is more efficient and leads to better performance compared to engaging the same roles in every communication round.

- We propose AgentDropout, a simple and flexible optimization method for MAS. It adjusts the overall communication topology through two steps: **Node Dropout** and **Edge Dropout**.

- We observe that for larger-scale and more

powerful LLMs, the performance of MAS designed on top of them can still be improved through interaction. AgentDropout can further enhance efficiency, domain transferability, and structural robustness.

2 Background

2.1 LLM-based MAS as Graph

Recently, Chan et al. (2024); Jiang et al. (2023); Wu et al. (2023) have proposed representing MAS using directed communication graphs, denoted as $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{F})$. $\boldsymbol{V} = \{v_i\}_{i=1}^{N}$ is a set of nodes, and each node represents an agent. $\boldsymbol{E} \subset \boldsymbol{V} \times \boldsymbol{V}$ is a set of edges, where each edge $(v_i, v_j) \in \boldsymbol{E}$ ($i \neq j$) denotes the communication path from agent v_i to agent v_j. $\boldsymbol{F} = \{f_i\}_{i=1}^{N}$ is a set of mapping functions that abstract the agent inference process. Each node v_i receives a query q and the outputs of its in-neighbors $\boldsymbol{Z}_i = \{o_j \mid (v_j, v_i) \in \boldsymbol{E}\}$ to produce its own output $o_i = f_i(q, \boldsymbol{Z}_i)$.

For some complex tasks, the entire communication graph needs to be run over multiple rounds to obtain the final answer, with each round operating on the communication graph defined as: $\boldsymbol{G}^{(t)} = \left(\boldsymbol{V}^{(t)}, \boldsymbol{E}_{\text{intra}}^{(t)}, \boldsymbol{F}^{(t)}\right)$, where $\boldsymbol{E}_{\text{intra}}^{(t)}$ denotes the **intra-round edges** which only connects nodes in the same round. Besides, the output of any node in step $t-1$ can be passed to any node in step t, forming **inter-round edges** $\boldsymbol{E}_{\text{inter}}^{(t)} \subset \boldsymbol{V}^{(t-1)} \times \boldsymbol{V}^{(t)}$ between graph $\boldsymbol{G}^{(t-1)}$ and $\boldsymbol{G}^{(t)}$. The subgraphs for all steps $\left\{\boldsymbol{G}^{(1)}, \boldsymbol{G}^{(2)}, \ldots, \boldsymbol{G}^{(T)}\right\}$ are combined using $\boldsymbol{E}_{\text{inter}}^{(t)}$ to form a multi-round communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V} = \bigcup_t \boldsymbol{V}^{(t)}$, $\mathcal{E} = \left(\bigcup_t \boldsymbol{E}_{\text{intra}}^{(t)}\right) \cup \left(\bigcup_t \boldsymbol{E}_{\text{inter}}^{(t)}\right)$, and $\mathcal{F} = \bigcup_t \boldsymbol{F}^{(t)}$.

2.2 Communication Redundancy in MAS

MAS often incurs high token consumption associated with the frequent communications between agents. Zhang et al. (2025) defined communication redundancy in MAS – excessive information flow that does not help improve performance. Considering a subgraph $\mathcal{G}_{\text{sub}} = (\mathcal{V}, \mathcal{E}', \mathcal{F})$, where $\mathcal{E}' \subseteq \mathcal{E}$, communication redundancy is considered to exist if $\mu(\mathcal{G}_{\text{sub}}) \geq \mu(\mathcal{G})$, where $\mu(\cdot)$ is the task-specified utility function evaluating the performance of the MAS. Redundant edges are those in $\mathcal{E} \setminus \mathcal{E}'$.

AgentPrune is an approach to reduce communication redundancy. By iteratively optimizing two trainable matrix masks (intra- and inter-round), AgentPrune parameterizes and prunes the edges in
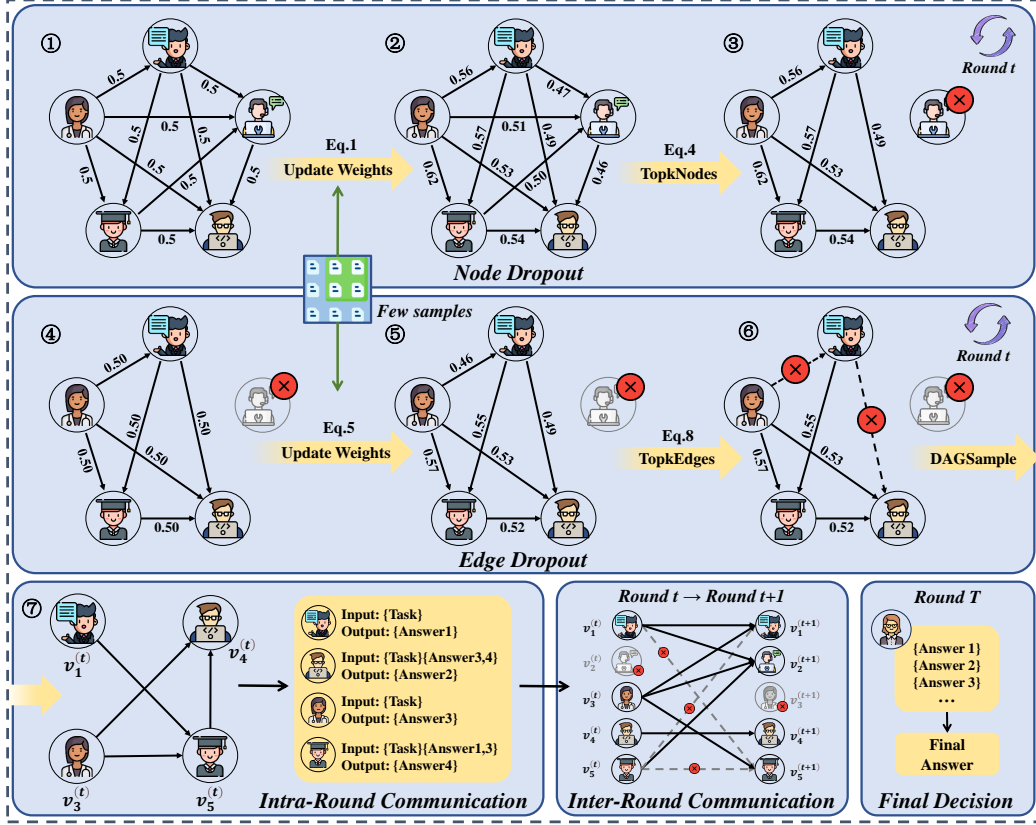
Figure 2: The overall process of AgentDropout. The first and second rows present Node Dropout and Edge Dropout procedures, respectively. The third row illustrates the cooperative reasoning process for both intra- and inter-round communication, as well as the generation of the final answer. For simplicity, the edges of inter-round communication are omitted in the demonstration of step 1~7, but they also participate in the update, dropout, and reasoning processes as shown in the middle block of the third row.

the communication graph to get rid of redundant edges (i.e. $\mathcal{E} \setminus \mathcal{E}'$).

3 AgentDropout

3.1 Overview

Motivation We argue that communication redundancy in MAS primarily arises from two sources: redundant agent roles and redundant information exchange. AgentPrune prunes redundant information exchange but neglects the existence of redundant agent roles and applies the same pruning strategy across all communication rounds, which still leaves redundancy in MAS and may result in performance degradation since different combinations of agent roles might be better suited for different stages of task solving. To this end, we propose a strategy called **AgentDropout** to enhance MAS efficiency and performance. We aim to remove the node that contributes least to the current task from each round's communication graph $G^{(t)}$ and eliminate both intra- and inter-round edges from the global graph $\mathcal{G}$ that have a minimal positive impact on the task solution.

Preparation We first convert the communication graph into a weighted graph $\tilde{\mathcal{G}}$, where each edge in the graph is a trainable continuous weight in the range of 0 to 1. Let the adjacency matrix set of $\tilde{\mathcal{G}}$ be $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_{\text{intra}} \cup \tilde{\mathcal{A}}_{\text{inter}}$, where $\tilde{\mathcal{A}}_{\text{intra}} = \bigcup_t \tilde{A}_{\text{intra}}^{(t)}$ is the subset containing intra-round adjacency matrices, $\tilde{A}_{\text{intra}}^{(t)} \in [0,1]^{N \times N}$ is the intra-round adjacency matrix of the round t communication graph and $\tilde{\mathcal{A}}_{\text{inter}} = \bigcup_t \tilde{A}_{\text{inter}}^{(t)}$ is the subset containing inter-round adjacency matrices, $\tilde{A}_{\text{inter}}^{(t)} \in [0,1]^{N \times N}$ is the inter-round adjacency matrix between round $t-1$ and t. The actual communication graph used during inference $\mathcal{G}$ is obtained by algorithm `DAGSample`, where each edge is sampled with its own weight as probability, and the output graph is guaranteed to be a directed acyclic graph (DAG). AgentDropout consists of two steps to optimize the graph: **Node Dropout** and **Edge Dropout**.

3.2 Node Dropout

In human teams tackling complex tasks, varying the roles involved in discussions across different rounds can enhance the team's performance by fo-

cusing on the most relevant contributors at each stage. Therefore, we first remove nodes from the communication graph that have less contribution to each round of communication. As shown in the first row in Figure 2, we initialize the intra- and inter-round adjacency matrices $\tilde{\mathcal{A}}_{\text{intra}} = \left\{ \tilde{\boldsymbol{A}}_{\text{intra}}^{(t)} \right\}_t$ and $\tilde{\mathcal{A}}_{\text{inter}} = \left\{ \tilde{\boldsymbol{A}}_{\text{inter}}^{(t)} \right\}_t$ by setting all the edge weights to 0.5.[1] We then train $\tilde{\mathcal{A}}_{\text{intra}}$ to get an optimized weighted communication graph, which is prepared for Node Dropout. The optimization objective is simply high task performance, formulated as:

$$\underset{\tilde{\mathcal{A}}_{\text{intra}}}{\arg\max} \, \mathbb{E}_{\mathcal{G} \sim \mathbb{G}} \left[\mu \left(\mathcal{G} \right) \right] \qquad (1)$$

where $\mathbb{G}$ represents the feasible domain of graph sampling. Since the utility function $\mu(\cdot)$ often depends on external APIs, it is commonly non-differentiable. Therefore, we apply a reinforcement algorithm that employs an unbiased policy gradient estimation, as proposed by Zhuge et al. (2024). This approach uses the probability-weighted average performance of a limited number of samples to approximate the overall performance expectation. We sample M communication graphs upon $\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}$ using DAGSample and run test instances on each graph $\mathcal{G}_m$ to obtain the performance score $\mu\left(\mathcal{G}_m\right)$:

$$\nabla_{\tilde{\mathcal{A}}_{\text{intra}}} \mathbb{E}_{\mathcal{G} \sim \mathbb{G}} \left[\mu \left(\mathcal{G} \right) \right]$$
$$\approx \frac{1}{M} \sum_{m=1}^{M} \mu \left(\mathcal{G}_m \right) \nabla_{\tilde{\mathcal{A}}_{\text{intra}}} \log \left(p_{\tilde{\mathcal{A}}_{\text{intra}}} \left(\mathcal{G}_m \right) \right) \qquad (2)$$

where $p_{\tilde{\mathcal{A}}_{\text{intra}}}\left(\mathcal{G}_m\right)$ represents the probability of sampling the communication graph $\mathcal{G}_m = \left(\mathcal{F}_m, \mathcal{E}_m, \mathcal{V}_m\right)$ given the current weighted intra-round adjacency matrices $\tilde{\mathcal{A}}_{\text{intra}}$:

$$p_{\tilde{\mathcal{A}}_{\text{intra}}} \left(\mathcal{G}_m \right) = \prod_t \prod_{\left(v_i, v_j \right) \in \boldsymbol{E}_{m,\text{intra}}^{(t)}} \tilde{\boldsymbol{A}}_{\text{intra}}^{(t)} [i,j] \qquad (3)$$

With the gradient calculated, we update $\tilde{\mathcal{A}}_{\text{intra}}$ using the gradient ascent method.

After the graph optimization, we select the nodes with the smallest weighted in- and out-degree in each intra-round graph and remove these nodes with all the edges connected to them:

$$\mathcal{V}_\Delta = \bigcup_t \left(\boldsymbol{V}^{(t)} \setminus \text{TopkNodes} \left(\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}, 1 - \alpha \right) \right) \quad (4)$$

---

[1]In detail, in the intra-round adjacency matrices, the diagonal elements are 0, and the other elements are 0.5. The inter-round adjacency matrices are upper triangular matrices with diagonal elements as 0 and others as 0.5.

where $\text{TopkNodes}\left(\cdot\right)$ is a function returning a set of k nodes with the largest sum of in-degree and out-degree according to the adjacency matrix, and α is the node dropout rate. $\tilde{\mathcal{A}}_{\text{intra}}$ and $\tilde{\mathcal{A}}_{\text{inter}}$ is also modified simultaneously by removing all rows and columns associated with the removed nodes.

3.3 Edge Dropout

Promoting more focused and efficient interactions between roles also improves overall teamwork effectiveness of task completion. Thus, we delete the intra- and inter-round edges in the communication graph that are less important to enhance communication efficiency. As shown in the second row in Figure 2, we reinitialize the intra- and inter-round adjacency matrices $\tilde{\boldsymbol{A}}_{\text{intra}}^{(t)}, \tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}$ after Node Dropout and train **both** of them from scratch. Differently, we aim at both high performance and token efficiency this time, formulated as:

$$\underset{\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}}{\arg\max} \, \mathbb{E}_{\mathcal{G} \sim \mathbb{G}_{\text{Node}}} \left[\mu \left(\mathcal{G} \right) \right]$$
$$- \left[\sum_{t=1}^{T} \text{rank} \left(\tilde{\boldsymbol{A}}_{\text{intra}}^{(t)} \right) + \sum_{t=2}^{T} \text{rank} \left(\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)} \right) \right] \quad (5)$$

where $\mathbb{G}_{\text{Node}}$ represents the feasible domain of graph sampling after Node Dropout. We also use the weighted average performance of the samples to approximate the performance expectation as in Equation (2). However, different from Equation (2), we calculate the probability and gradient for both $\tilde{\mathcal{A}}_{\text{intra}}$ and $\tilde{\mathcal{A}}_{\text{inter}}$ here:

$$p_{\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}} \left(\mathcal{G}_m \right) = \prod_{t=1}^{T} \prod_{\left(v_i, v_j \right) \in \boldsymbol{E}_{m,\text{intra}}^{(t)}} \tilde{\boldsymbol{A}}_{\text{intra}}^{(t)} [i,j]$$
$$\cdot \prod_{t=2}^{T} \prod_{\left(v_i, v_j \right) \in \boldsymbol{E}_{m,\text{inter}}^{(t)}} \tilde{\boldsymbol{A}}_{\text{inter}}^{(t)} [i,j] \qquad (6)$$

The second term here represents low-rank sparsity, which serves as a proxy for communication efficiency. Since low-rank optimization is an NP-hard problem, we replace the rank function with the nuclear norm following Zhang et al. (2025):

$$\underset{\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}}{\arg\min} \, \sum_{t=1}^{T} \|\tilde{\boldsymbol{A}}_{\text{intra}}^{(t)}\|_* + \sum_{t=2}^{T} \|\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}\|_* \qquad (7)$$

Then in each intra- and inter-round adjacency matrix, we select the edges corresponding to the largest elements with a ratio of $1 - \alpha$ to keep and delete the rest of them:

$$\mathcal{E}_\Delta = \bigcup_t \left(\boldsymbol{E}_{\text{intra}}^{(t)} \setminus \text{TopkEdges} \left(\tilde{\boldsymbol{A}}_{\text{intra}}^{(t)}, 1 - \beta \right) \right)$$
$$\cup \bigcup_t \left(\boldsymbol{E}_{\text{inter}}^{(t)} \setminus \text{TopkEdges} \left(\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}, 1 - \beta \right) \right) \quad (8)$$

Method	Edge DR.	Node DR.	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Avg.
				Base model: Llama3-8B-Instruct					
Vanilla	✗	✗	53.59	70.23	41.67	91.11	75.00	53.33	64.16
CoT	✗	✗	$56.86_{\uparrow3.27}$	$70.47_{\uparrow0.24}$	$43.75_{\uparrow2.08}$	$92.25_{\uparrow1.14}$	$76.17_{\uparrow1.67}$	$54.17_{\uparrow0.84}$	$65.61_{\uparrow1.45}$
$MAS_{round=1}$	✗	✗	$56.21_{\uparrow2.62}$	$69.30_{\downarrow0.97}$	$45.29_{\uparrow3.62}$	$91.67_{\uparrow0.55}$	$76.67_{\uparrow1.67}$	$48.33_{\downarrow5.00}$	$64.58_{\uparrow0.42}$
$MAS_{round=T}$	✗	✗	$60.13_{\uparrow6.54}$	$71.48_{\uparrow1.25}$	$45.41_{\uparrow3.75}$	$90.56_{\downarrow0.55}$	$77.56_{\uparrow2.56}$	$49.17_{\downarrow4.16}$	$65.72_{\uparrow1.56}$
AgentPrune	✓	✗	$60.78_{\uparrow7.19}$	$71.02_{\uparrow0.79}$	$47.22_{\uparrow5.55}$	$90.00_{\downarrow1.11}$	$78.34_{\uparrow3.34}$	$51.67_{\downarrow1.66}$	$66.51_{\uparrow2.35}$
AgentDropout	✓	✓	$\mathbf{62.75}_{\uparrow9.16}$	$\mathbf{73.13}_{\uparrow2.90}$	$\mathbf{47.78}_{\uparrow6.11}$	$\mathbf{92.59}_{\uparrow1.48}$	$\mathbf{80.11}_{\uparrow5.11}$	$\mathbf{55.84}_{\uparrow2.51}$	$\mathbf{68.70}_{\uparrow4.54}$
				Base model: Qwen2.5-72B-Instruct					
Vanilla	✗	✗	82.35	91.02	83.75	98.89	92.67	85.28	88.99
CoT	✗	✗	$83.66_{\uparrow1.31}$	$92.19_{\uparrow1.17}$	$84.58_{\uparrow0.83}$	$\mathbf{100}_{\uparrow1.11}$	$93.35_{\uparrow0.68}$	$86.67_{\uparrow1.39}$	$90.08_{\uparrow1.09}$
$MAS_{round=1}$	✗	✗	$82.35_{\uparrow0.00}$	$93.52_{\uparrow2.50}$	$84.58_{\uparrow0.83}$	$\mathbf{100}_{\uparrow1.11}$	$92.36_{\downarrow0.31}$	$84.17_{\downarrow1.11}$	$89.50_{\uparrow0.51}$
$MAS_{round=T}$	✗	✗	$84.31_{\uparrow1.96}$	$93.28_{\uparrow2.26}$	$85.83_{\uparrow2.08}$	$\mathbf{100}_{\uparrow1.11}$	$94.07_{\uparrow1.40}$	$87.08_{\uparrow1.80}$	$90.76_{\uparrow1.77}$
AgentPrune	✓	✗	$83.66_{\uparrow1.31}$	$93.67_{\uparrow2.65}$	$87.08_{\uparrow3.33}$	$99.44_{\uparrow0.56}$	$94.33_{\uparrow1.66}$	$86.67_{\uparrow1.39}$	$90.81_{\uparrow1.82}$
AgentDropout	✓	✓	$\mathbf{84.97}_{\uparrow2.62}$	$\mathbf{93.75}_{\uparrow2.73}$	$\mathbf{87.50}_{\uparrow3.75}$	$\mathbf{100}_{\uparrow1.11}$	$\mathbf{95.34}_{\uparrow2.67}$	$\mathbf{87.92}_{\uparrow2.64}$	$\mathbf{91.58}_{\uparrow2.59}$
				Base model: Deepseek-V3-671B-Instruct					
Vanilla	✗	✗	84.97	94.68	84.58	$\mathbf{100}$	93.67	88.43	91.06
CoT	✗	✗	$84.31_{\downarrow0.66}$	$95.15_{\uparrow0.47}$	$85.42_{\uparrow0.84}$	$\mathbf{100}_{\uparrow0.00}$	$93.94_{\uparrow0.27}$	$89.26_{\uparrow0.83}$	$91.34_{\uparrow0.28}$
$MAS_{round=1}$	✗	✗	$89.98_{\uparrow5.01}$	$95.54_{\uparrow0.86}$	$86.67_{\uparrow2.19}$	$\mathbf{100}_{\uparrow0.00}$	$93.50_{\downarrow0.17}$	$89.17_{\uparrow0.74}$	$92.63_{\uparrow1.57}$
$MAS_{round=T}$	✗	✗	$89.54_{\uparrow4.57}$	$95.49_{\uparrow0.81}$	$87.50_{\uparrow2.92}$	$\mathbf{100}_{\uparrow0.00}$	$94.33_{\uparrow0.66}$	$89.26_{\uparrow0.83}$	$92.69_{\uparrow1.63}$
AgentPrune	✓	✗	$90.20_{\uparrow5.23}$	$95.49_{\uparrow0.81}$	$87.92_{\uparrow3.34}$	$\mathbf{100}_{\uparrow0.00}$	$95.00_{\uparrow1.33}$	$90.91_{\uparrow2.47}$	$93.25_{\uparrow2.19}$
AgentDropout	✓	✓	$\mathbf{90.85}_{\uparrow5.88}$	$\mathbf{95.63}_{\uparrow0.95}$	$\mathbf{88.33}_{\uparrow3.75}$	$\mathbf{100}_{\uparrow0.00}$	$\mathbf{95.79}_{\uparrow2.12}$	$\mathbf{91.74}_{\uparrow3.31}$	$\mathbf{93.72}_{\uparrow2.66}$

Table 1: Performance comparison between AgentDropout and other baseline reasoning techniques. **Edge DR.** and **Node DR.** represent the Edge Dropout and Node Dropout methods, respectively.

where TopkEdges($\cdot$) is a function returning a set of edges corresponding to top-k elements in the adjacency matrix, and β is the edge dropout rate.

As shown in the third row in Figure 2, the final communication graph $\hat{\mathcal{G}}$ is sampled from the optimized $\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}$ using DAGSample. The agents then engage in multi-round communication based on the topology defined in $\hat{\mathcal{G}}$, and their individual responses are then aggregated to generate the final answer. More detailed information on the optimization procedure is presented Appendix A.1.

3.4 Discussion

Previous studies (You et al., 2020; Zhang et al., 2024a) have demonstrated that high-quality topological graphs can be trained using a limited number of samples. Therefore, in our approach, we employ only a small number of samples to update the parameters of the adjacency matrices for the communication graph, subsequently applying these matrices during two dropout processes. The number of these samples is merely in the order of tens, extracted from either the train or the validation set of the dataset, thereby maintaining the training cost and token consumption at relatively low levels.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks We conduct experiments on MAS based on models of varying sizes, namely Meta-Llama3-8B-Instruct (AI@Meta, 2024), Qwen2.5-72B-Instruct (Yang et al., 2024), and Deepseek-V3-671B-Instruct (Liu et al., 2024a). We evaluate the general reasoning capabilities of our method using the MMLU (Hendrycks et al., 2021) dataset; the mathematical reasoning abilities using the GSM8k (Cobbe et al., 2021), MultiArith (Roy and Roth, 2015), AQuA (Patel et al., 2021), and SVAMP (Ling et al., 2017) datasets; and the code generation capabilities using the HumanEval (Chen et al., 2021) dataset.

Baselines For tasks performed by a single agent, we compare with the direct reasoning method, known as Vanilla, and the Chain-of-Thought (CoT, Wei et al., 2022) approach. In the context of multi-agent collaboration, for methods that only communicate within intra-round, we compare the single-round dialogue MAS, denoted as $MAS_{\text{round}=1}$. For methods that communicate both within intra-round and across inter-round, we compare $MAS_{\text{round}=T}$ and AgentPrune (Zhang et al., 2025), where round represents the number of inter-round communication rounds among agents. In particular, both AgentPrune and AgentDropout are optimized based on $MAS_{\text{round}=T}$, allowing for a more intuitive comparison of these methods.

Implementation Details For experiments with Llama3 and Qwen2.5, we perform inference using vllm on Nvidia A800 GPUs. For Deepseek-V3, we utilize the Deepseek API to evaluate each method. We set the temperature to 1, the sample times M

Method	MMLU		GSM8K		AQuA		MultiArith		SVAMP		HumanEval		Avg.	
	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.
Vanilla	99K	44K	142K	209K	32K	89K	16K	15k	27K	19K	91K	25K	68K	67K
CoT	129K	70K	154K	337K	34K	105K	18K	23k	30K	50K	93K	29K	76K	102K
$MAS_{round=1}$	1.4M	355K	8.5M	1.9M	1.1M	390K	1.1M	218K	1.9M	402K	581K	134K	2.4M	567K
$MAS_{round=T}$	1.6M	387K	16M	3.4M	2.4M	745K	2.1M	388K	3.7M	721K	2.6M	492K	4.7M	1.0M
AgentPrune	1.3M	367K	15M	3.6M	2.0M	759K	1.9M	393K	3.4M	714K	1.5M	465K	4.2M	1.0M
AgentDropout	1.1M	333K	12M	2.8M	1.3M	634K	1.4M	312K	2.6M	594K	1.1M	359K	3.3M	839K

Table 2: Token consumption comparison in the Llama model. **Ptok.** denotes the number of prompt tokens for the agents, while **Ctok.** represents the number of completion tokens generated by the agents.

Graph	Method	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Avg.	Ptok.	Ctok.
Layered	MAS	57.52	71.72	47.92	87.26	78.84	52.89	66.13	4.3M	1.1M
	AgentPrune	59.48	71.09	46.67	88.93	78.67	56.20	66.84	3.7M	1.0M
	AgentDropout	**61.44**	**72.27**	**49.17**	**90.25**	**79.75**	**57.02**	**68.32**	2.8M	797K
Random	MAS	62.75	70.63	46.67	87.22	77.17	50.41	65.81	4.2M	1.0M
	AgentPrune	63.40	69.61	44.17	91.11	78.34	51.24	66.31	3.4M	1.0M
	AgentDropout	**64.71**	**72.03**	**47.50**	**91.67**	**79.50**	**53.72**	**68.19**	2.7M	834K

Table 3: Performance and average token consumption achieved with different initial communication graph topological structures.

in $\{10, 20\}$, the learning rate η to 0.1, and the note and edge dropout rate α and β in $\{0.1, 0.2\}$, respectively. For all benchmarks, the number of training samples we extracted is 40. For general reasoning tasks and mathematical tasks, we set the number of communication rounds $T = 2$, while for code generation tasks, we configure $T = 4$. To ensure a fair comparison, we directly adopt the agent configuration files from AgentPrune.

4.2 Main Result

AgentDropout outperforms other baselines on multiple benchmarks. As shown in Table 1, our method consistently outperforms both the single-agent method CoT and the state-of-the-art multi-agent communication method AgentPrune across all benchmarks, regardless of the model scale. Specifically, using the Llama3 model, Agent-Dropout improves accuracy by 2.19 compared to the SOTA method. In experiments with Qwen2.5 and Deepseek-V3, the accuracy improves by 0.77 and 0.47, respectively. These results demonstrate the effectiveness of AgentDropout. A case study of the reasoning results is provided in Appendix A.2.

AgentDropout improves the stability of MAS. In Table 1, it can be observed that when the model capability is relatively weak (e.g., Llama3), optimization methods for single agents, such as CoT, exhibit more significant performance improvements, whereas the performance gains from multi-agent interaction and collaboration are less stable, as seen in MAS and AgentPrune. However, when the model capability is sufficiently strong, the performance improvements brought by single-agent optimization methods diminish, while multi-agent communication methods achieve more stable enhancements. Notably, for models of different scales, AgentDropout consistently enables stable performance improvements in MAS, demonstrating its model robustness.

AgentDropout reduces the token cost of multi-agent communication and cooperation. As shown in Table 2, compared to the multi-round communication method $MAS_{round=T}$, Agent-Dropout significantly reduces token consumption. In contrast to AgentPrune, which primarily reduces prompt token consumption, AgentDropout not only further decreases prompt tokens but also significantly reduces completion tokens, revealing its token efficiency. For more details, please refer to Appendix A.3.

4.3 Analysis

In this part, we conducted experiments using the Llama3 model, while maintaining the same experimental settings as described in Section 4.1.

Structure Robustness To verify the stability of AgentDropout when initialized with different topological structures, we also conducted supplementary experiments on communication graphs initialized with layered and random graphs, which are introduced in Appendix A.4. As shown in Table 3, AgentDropout consistently enhances the performance and reduces the average consumption of prompt tokens and completion tokens of

Dropout Rate	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Avg.	Ptok.	Ctok.
0.8	57.52	70.86	46.67	89.44	78.67	52.89	66.01	**856K**	**230K**
0.6	60.78	71.09	45.42	89.44	**80.37**	50.41	66.25	1.3M	434K
0.4	60.13	**73.36**	44.99	90.00	78.33	52.07	66.48	1.9M	648K
0.2	**62.75**	73.13	**47.78**	**92.59**	80.11	**55.84**	**68.70**	3.3M	839K

Table 4: Impact of varying dropout rates on the performance and average token consumption of AgentDropout.

Method	Edge DR.	Node DR.	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Avg.
MAS	✗	✗	60.13	71.48	45.41	90.56	77.56	49.17	65.72
Random Dropout	✗	✓	61.72	70.53	43.75	92.23	79.00	52.89	66.69
	✓	✗	60.13	71.41	46.05	90.27	77.67	51.23	66.13
Learned Dropout	✗	✓	**63.76**	70.70	46.25	90.28	79.17	50.83	66.83
	✓	✗	61.87	70.63	47.08	90.00	78.03	48.76	66.06
Single Learning	✓	✓	60.78	72.58	46.25	91.67	78.83	48.73	66.47
AgentDropout	✓	✓	62.75	**73.13**	**47.78**	**92.59**	**80.11**	**55.84**	**68.70**

Table 5: Performance achieved with different dropout strategies. For Random Dropout, ✓ denotes the random dropout strategy is applied to the marked dropout step, while the dropout strategy in the other step remains unchanged, as per AgentDropout.

Method	GSM8K	AQuA	MultiArith	SVAMP	Avg.
GSM8K	73.13	46.67	92.22	**80.33**	73.09
AQuA	**73.84**	**47.78**	**93.13**	78.21	73.24
MultiArith	71.41	47.50	92.59	78.33	72.46
SVAMP	70.86	47.50	91.95	80.11	72.61
AgentPrune	71.02	47.22	90.00	78.34	71.65
AgentDropout	73.13	**47.78**	92.59	80.11	**73.40**

Table 6: Performance comparison using different combinations of training and test sets. The row headers indicate the training set used for graph topology learning, and the column headers represent the test set.

MAS across various graph structures, which further demonstrates its robustness.

Effect of Dropout Rate To investigate the impact of different dropout rates on the performance of MAS, we conducted experiments with varying values of α and β, as shown in Table 4. The results demonstrate that higher dropout rates lead to a corresponding decline in the average performance of MAS. However, compared with the SOTA methods, AgentDropout still achieves superior average performance across different dropout rates, which substantiates the effectiveness of dropout. In extreme cases, higher dropout rates can be set to ensure efficiency. Nevertheless, for tasks such as AQuA and HumanEval, MAS exhibits lower sensitivity to dropout rates, suggesting that future work could focus on designing more appropriate MAS frameworks for such scenarios.

Different Dropout Strategies To verify the effectiveness of our dropout strategy, we conducted a series of comparative experiments. First, we performed an ablation study on the overall Agent-Dropout methodology, comparing it against scenarios that use only Node Dropout and only Edge Dropout. Additionally, to rule out the possibility that the performance improvement is simply due to the shortening of the reasoning chain, we conduct an experiment where the proposed dropout mechanism is replaced with a random dropout approach in either the Node Dropout or Edge Dropout stage while keeping all other processes unchanged.

As shown in the "Learned Dropout" block of Table 5, applying either Node Dropout or Edge Dropout alone results in improvements compared to the vanilla MAS, indicating the reasonableness of the dropout approaches. However, the simultaneous application of both Node Dropout and Edge Dropout, referred to as "AgentDropout", yields the optimal performance. In the random dropout experiments, randomly dropping either edges or nodes within the AgentDropout framework leads to performance degradation, further substantiating the effectiveness of our proposed dropout strategy.

Training Strategy To validate the effectiveness of our two-round adjacency matrix learning strategy (one for Node Dropout and one for Edge Dropout), we conducted comparative experiments with single-round learning. Specifically, we trained both intra- and inter-round adjacency matrices simultaneously, and then directly applied them to perform Node and Edge Dropout on MAS. The experimental results are presented in Table 5. The performance of AgentDropout significantly outperforms that of single-round learning. This improvement can be attributed to the fact that after Node Dropout, the communication graph generates new inference chains, necessitating an additional train-

ing round to effectively implement further dropout.

Domain Transferability Our approach requires some training instances of the current task to conduct communication graph optimization and Node/Edge dropout. However, in the evaluation phase, there may be no training data available that directly corresponds to the current test data. To this point, we conduct a domain robustness test, assessing how well our system works when only roughly similar training data is available. Specifically, we utilize dataset A to optimize the communication topology graph and evaluate the trained MAS on dataset B. We conduct experiments on mathematics-related datasets.

As shown in Table 6. When using other datasets as training samples, AgentDropout still maintains robust performance. When trained on the more challenging AQuA dataset, it achieves an average performance improvement of 1.59 over the SOTA method, while for the simpler MultiArith dataset, the average improvement is 0.81. This demonstrates that the communication subgraphs learned by AgentDropout for tasks of the same category exhibit strong generalizability. Moreover, the effectiveness of AgentDropout is positively correlated with the difficulty of the training samples. Specifically, for tasks of the same type, the more challenging the task, the stronger the generalizability learned by MAS. Therefore, when applying AgentDropout to few-shot tasks, selecting similar datasets for training is recommended, which further validates its domain robustness.

5 Related Work

5.1 LLM-based MAS

MAS based on LLMs has emerged as a crucial tool for tackling complex tasks across various domains. Early research has demonstrated the significant potential of LLM-driven agents in collaboratively solving problems through multi-agent debates (Liang et al., 2024; Du et al., 2024), laying the foundation for several research directions, including role-playing in complex reasoning (Wang et al., 2024c; Chen et al., 2024a), collaborative software development (Ishibashi and Nishimura, 2024; Qian et al., 2024c; Hong et al., 2024), and embodied agent interactions (Mandi et al., 2024; Zhang et al., 2024b; Guo et al., 2024). Recent studies have shown that increasing the number and diversity of agents can enhance the overall performance of MAS (Wang et al., 2025a; junyou li et al., 2024;

Chen et al., 2025). As LLM-based MAS continues to scale in size and complexity, the issues of computational cost and communication efficiency have become more pronounced (Chen et al., 2024b; Li et al., 2024). Recent work has proposed methods to optimize MAS efficiency, such as Optima (Chen et al., 2024c) and AgentPrune (Zhang et al., 2025). In this paper, we address the problem of communication redundancy by applying dropout to nodes and edges in the graph structure of MAS, significantly improving computational cost and communication efficiency in practical applications.

5.2 MAS as Graphs

In recent years, researchers have proposed learning graph connections to enhance the collaborative performance of MAS (Hu et al., 2024; Pesce and Montana, 2023). Prior to the widespread adoption of LLMs, existing methods explored optimal communication graph structures for MAS using techniques such as graph diffusion, weighted GNNs, and transformers (Liu et al., 2022; Pesce and Montana, 2023; Hu et al., 2024). For LLM-based MAS, previous work has also attempted to use graphs to represent interactions between agents. For instance, methods like ChatEval (Chan et al., 2024) and AutoGen (Wu et al., 2023) implicitly incorporate graph structures to model simultaneous conversations, while other approaches such as STOP (Zelikman et al., 2023) and DSPy (Khattab et al., 2024) optimize prompt and reasoning structures. Additionally, methods like MacNet (Qian et al., 2025), GPTSwarm (Zhuge et al., 2024), EvoMAC (Hu et al., 2025) and DyLAN (Liu et al., 2024b) optimize the topology of agents communication.

6 Conclusion

In this paper, inspired by management theories that roles in efficient teams are often dynamically adjusted, we propose AgentDropout, a novel topology optimization method for MAS with domain transferability and structure robustness. AgentDropout dynamically adjusts the participating agents and communication links among agents in each round, allowing for more flexible and adaptive team configurations. Experiments on a series of tasks demonstrate that AgentDropout not only reduces token consumption but also consistently enhances the performance of MAS across various tasks. This provides valuable insights for future exploration of larger-scale multi-agent interactions.

Limitations

The tasks validated in this paper are relatively limited and focus mainly on conventional reasoning tasks, mathematical tasks, and code generation tasks. To better evaluate the generalizability of AgentDropout, additional experiments across a broader and more diverse range of tasks are essential. Furthermore, although AgentDropout demonstrates promising efficiency and performance, the current framework remains constrained by predefined agent roles and prompts. When addressing tasks in unfamiliar domains, the careful design of new initial roles and prompts is still crucial. Future work could also explore integrating methods for automatically designing roles and prompts to enhance flexibility. Additionally, although AgentDropout reduces the token consumption of MAS, it remains relatively high, indicating significant opportunities for further optimization.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

AI@Meta. 2024. Llama 3 model card. *None*.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374.

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. 2024a. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2025. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *The Thirteenth International Conference on Learning Representations*.

Weize Chen, Chenfei Yuan, Jiarui Yuan, Yusheng Su, Chen Qian, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2024b. Beyond natural language: LLMs leveraging alternative formats for enhanced reasoning and communication. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10626–10641, Miami, Florida, USA. Association for Computational Linguistics.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2024c. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. *ArXiv preprint*, abs/2410.08115.

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. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Shen Gao, Hao Li, Zhengliang Shi, Chengrui Huang, Quan Tu, Zhiliang Tian, Minlie Huang, and Shuo Shang. 2024. 360 {\deg} rea: Towards a reusable experience accumulation with 360 {\deg} assessment for multi-agent system. *ArXiv preprint*, abs/2404.05569.

Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L Griffiths, and Mengdi Wang. 2024. Embodied llm agents learn to cooperate in organized teams. *ArXiv preprint*, abs/2403.12482.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang

Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2024. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*.

Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. 2025. Self-evolving multi-agent networks for software development. In *The Thirteenth International Conference on Learning Representations*.

Yoichi Ishibashi and Yoshimasa Nishimura. 2024. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization. *ArXiv preprint*, abs/2404.02183.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

junyou li, Qin Zhang, Yangbin Yu, QIANG FU, and Deheng Ye. 2024. More agents is all you need. *Transactions on Machine Learning Research*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Steve WJ Kozlowski and Daniel R Ilgen. 2006. Enhancing the effectiveness of work groups and teams. *Psychological science in the public interest*, 7(3):77–124.

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

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *ArXiv preprint*, abs/2412.19437.

Yuntao Liu, Yong Dou, Yuan Li, Xinhai Xu, and Donghong Liu. 2022. Temporal dynamic weighted graph convolution for multi-agent reinforcement learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 44.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024b. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

E. Locke. 2011. *Handbook of Principles of Organizational Behavior: Indispensable Knowledge for Evidence-Based Management*. Wiley.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Emanuele Pesce and Giovanni Montana. 2023. Learning multi-agent coordination through connectivity-driven communication. *Machine Learning*, 112(2):483–514.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, YiFei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024a. Experiential co-learning of software-developing agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5628–5640, Bangkok, Thailand. Association for Computational Linguistics.

Chen Qian, Jiahao Li, Yufan Dang, Wei Liu, YiFei Wang, Zihao Xie, Weize Chen, Cheng Yang, Yingli Zhang, Zhiyuan Liu, et al. 2024b. Iterative experience refinement of software-developing agents. *ArXiv preprint*, abs/2405.04219.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng

Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024c. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv preprint*, abs/2403.05530.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025a. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Yutong Wang, Jiali Zeng, Xuebo Liu, Fandong Meng, Jie Zhou, and Min Zhang. 2024b. TasTe: Teaching large language models to translate through self-reflection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6144–6158, Bangkok, Thailand. Association for Computational Linguistics.

Yutong Wang, Jiali Zeng, Xuebo Liu, Derek F. Wong, Fandong Meng, Jie Zhou, and Min Zhang. 2025b. DelTA: An online document-level translation agent based on multi-level memory. In *The Thirteenth International Conference on Learning Representations*.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2024c. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 257–279, Mexico City, Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *ArXiv preprint*, abs/2308.08155.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *ArXiv preprint*, abs/2412.15115.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. 2020. Drawing early-bird tickets: Toward more efficient training of deep networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-taught optimizer (stop): Recursively self-improving code generation. *ArXiv preprint*, abs/2310.02304.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2025. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*.

Guibin Zhang, Yanwei Yue, Kun Wang, Junfeng Fang, Yongduo Sui, Kai Wang, Yuxuan Liang, Dawei Cheng, Shirui Pan, and Tianlong Chen. 2024a. Two heads are better than one: Boosting graph sparse training via semantic and topological awareness. In *Forty-first International Conference on Machine Learning*.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. 2024b. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen

Schmidhuber. 2024. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

A Appendix

A.1 Optimization Algorithm of AgentDropout

Algorithm 1 presents the detailed pseudo code of AgentDropout. The input parameters for this algorithm mainly include communication graph $\mathcal{G}$ to be optimized, along with its intra- and inter-round adjacency matrices $\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}$. The output is the optimized communication graph $\hat{\mathcal{G}}$.

During NodeDropout, we first call the Initialize($\cdot$) function to initialize the adjacency matrices by setting the relevant elements to 0.5 (Line 1). We then train the intra-round matrices to optimize the task performance of the communication graphs sampled from these matrices (Lines 2 ~5). After the training process, we identify the nodes in each communication round with the smallest in- and out-degree (Line 6) and remove them, along with all the edges connected to them, by invoking the NodeDropout($\cdot$) function (Line 7).

During EdgeDropout, we initialize the intra- and inter-round adjacency matrices again (Line 8) and train both of them to optimize task performance and graph sparsity simultaneously (Lines 9 ~12). Next, we identify the intra- and inter-round edges with the smallest weights in their respective matrices (Line 13) and remove them from the graph by calling the EdgeDropout($\cdot$) function (Line 14). Finally, we sample the communication graph used in actual evaluation by calling the DAGSample($\cdot$) function, which samples the graph by treating the weights in the matrices as sampling possibility and guarantees the output to be a DAG graph (Line 15).

A.2 Case Study

Given a multiple-choice physics question from the MMLU dataset, in the first round of interaction, the Knowledgeable Expert provides an initial analysis, emphasizing the need to search for information related to the equation and relevant atomic properties. The Critic critiques the response of the Knowledgeable Expert, pointing out the lack of practical analysis and application, and suggests further refinement of the reasoning. Upon receiving feedback, the Mathematician conducts a detailed analysis of the problem, noting that the equation cannot be applied to H^-, and discusses variations across different atoms, providing more specific reasoning. The Psychologist offers feedback from a psychological perspective, affirming the process of critique and revision, and encourages more comprehensive critical thinking. In the second round of interaction, the Historian briefly summarizes the reasons. The Critic revisits the perspectives of other agents, suggesting that the approach of Mathematicians to different atoms could be further improved, and emphasizes the impact of atomic properties on the equation. After receiving feedback, the Mathematician revises the analysis, clarifies which atoms can be addressed by modifying the application of the Rydberg equation, and concludes that the Hydride ion (H^-) is the only unsuitable option. The Historian in the first round and the Knowledgeable Expert in the second round are dropped out.

A.3 Token Efficiency

As shown in Table 7, AgentDropout significantly reduces token consumption compared to the multi-round method $\text{MAS}_{\text{round}=T}$. Unlike AgentPrune, which reduces only prompt tokens, AgentDropout reduces both prompt and completion tokens. In the Llama3 model, it achieves a 21.4% reduction in prompt tokens and 16.1% in completion tokens. For Qwen2.5 and Deepseek-V3, the reductions are 24.4% and 18.9% in prompt tokens, and 21.4% and 17.6% in completion tokens, respectively, demonstrating its token efficiency.

A.4 Different Communication Topologies

In the main experiments, we initialize the intra-round communication graph using a fully connected graph. In the analysis experiments, we supplement this by conducting additional experiments with intra-round graphs initialized using layered graphs and random graphs, as shown in Figure 3. For both fully connected graphs and random graphs, we first process the communication topology into a DAG using DAGSample, and then perform inference and communication based on the topological order (Zhuge et al., 2024; Qian et al., 2025). For layered graphs, similar to multi-layer perceptrons, the agents in the first layer perform inference first, followed by message passing to the agents in the second layer, and so on.

Algorithm 1: Communication Graph Dropout

Input: Initial communication graph $\mathcal{G} = \left(\bigcup_t \boldsymbol{V}^{(t)}, \left(\bigcup_t \boldsymbol{E}_{\text{intra}}^{(t)} \right) \cup \left(\bigcup_t \boldsymbol{E}_{\text{inter}}^{(t)} \right), \bigcup_t \boldsymbol{F}^{(t)} \right)$, Initial adjacency matrices $\tilde{\mathcal{A}}_{\text{intra}} = \left\{ \tilde{\mathcal{A}}_{\text{intra}}^{(t)} \right\}_t, \tilde{\mathcal{A}}_{\text{inter}} = \left\{ \tilde{\mathcal{A}}_{\text{inter}}^{(t)} \right\}_t$, Training steps K, Sampling times M, Learning rate η, Node dropout rate α, Edge dropout rate β

Output: Communication graph after dropout $\hat{\mathcal{G}}$

`// Node Dropout`

1   `Initialize` $\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \right)$ `// Set the elements to 0.5`

2   **for** iter $k \leftarrow 1$ **to** K **do**

3     $\{\mathcal{G}_m\}_{m=1}^{M} \leftarrow \left\{ \texttt{DAGSample}\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \right) \right\}_{m=1}^{M}$ `// Sample` M `communication graphs`

4     $\tilde{\mathcal{A}}_{\text{intra}} \leftarrow \tilde{\mathcal{A}}_{\text{intra}} + \eta \cdot \frac{1}{M} \sum_{m=1}^{M} \mu\left(\mathcal{G}_m \right) \nabla_{\tilde{\mathcal{A}}_{\text{intra}}} \log\left(p_{\tilde{\mathcal{A}}_{\text{intra}}}\left(\mathcal{G}_m \right) \right)$ `// Optimize` $\tilde{\mathcal{A}}_{\text{intra}}$ `according to Equation (1)`

5   **end**

6   $\mathcal{V}_\Delta = \bigcup_t \left(\boldsymbol{V}^{(t)} \setminus \texttt{TopkNodes}\left(\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}, 1 - \alpha \right) \right)$ `// Select nodes with small degrees`

7   $\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \leftarrow \texttt{NodeDropout}\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}, \mathcal{V}_\Delta \right)$ `// Remove nodes from the graph`

  `// Edge Dropout`

8   `Initialize` $\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \right)$

9   **for** iter $k \leftarrow 1$ **to** K **do**

10     $\{\mathcal{G}_m\}_{m=1}^{M} \leftarrow \left\{ \texttt{DAGSample}\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \right) \right\}_{m=1}^{M}$ `// Sample` M `communication graphs`

11     $\left\{ \tilde{\boldsymbol{A}}_{\text{intra}}, \tilde{\boldsymbol{A}}_{\text{inter}} \right\} \leftarrow \left\{ \tilde{\boldsymbol{A}}_{\text{intra}}, \tilde{\boldsymbol{A}}_{\text{inter}} \right\} + \eta \cdot \frac{1}{M} \sum_{m=1}^{M} \mu\left(\mathcal{G}_m \right) \nabla_{\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}} \log\left(p_{\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}}\left(\mathcal{G}_m \right) \right)$ `// Optimize` $\tilde{\mathcal{A}}$ `according to Equation (5)`

12   **end**

13   $\mathcal{E}_\Delta = \bigcup_t \left(\boldsymbol{E}_{\text{intra}}^{(t)} \setminus \texttt{TopkEdges}\left(\tilde{\boldsymbol{A}}_{\text{intra}}^{(t)}, 1 - \beta \right) \right) \cup \bigcup_t \left(\boldsymbol{E}_{\text{inter}}^{(t)} \setminus \texttt{TopkEdges}\left(\tilde{\boldsymbol{A}}_{\text{inter}}^{(t)}, 1 - \beta \right) \right)$ `// Select edges with small weights`

14   $\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \leftarrow \texttt{EdgeDropout}\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}}, \mathcal{E}_\Delta \right)$ `// Remove edges from the graph`

15   $\hat{\mathcal{G}} \leftarrow \texttt{DAGSample}\left(\tilde{\mathcal{A}}_{\text{intra}}, \tilde{\mathcal{A}}_{\text{inter}} \right)$ `// Sample the final communication DAG`
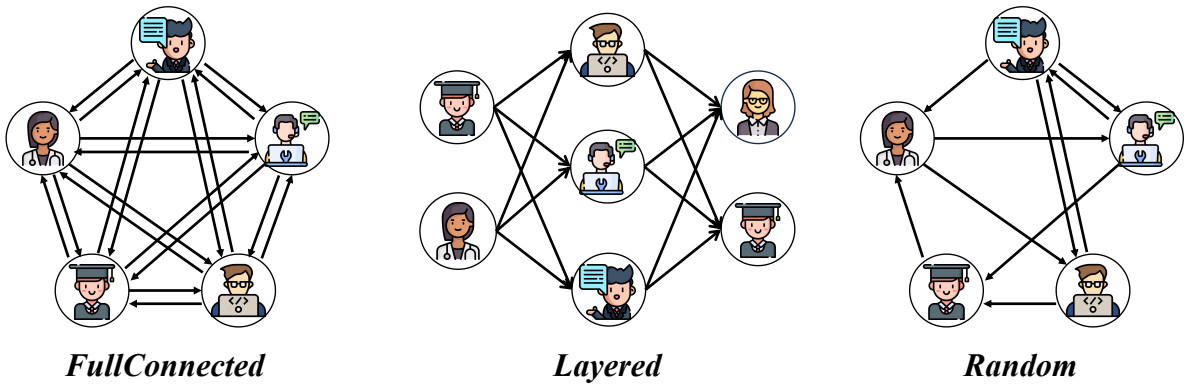
16   **return** $\hat{\mathcal{G}}$

FullConnected   **Layered**   **Random**

Figure 3: Full connected, layer and random graph as demonstrations of intra-round communication topologies.

Method	MMLU		GSM8K		AQuA		MultiArith		SVAMP		HumanEval		Avg.	
	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.	Ptok.	Ctok.
Base model: Llama3-8B-Instruct														
Vanilla	99K	44K	142K	209K	32K	89K	16K	15k	27K	19K	91K	25K	68K	67K
CoT	129K	70K	154K	337K	34K	105K	18K	23k	30K	50K	93K	29K	76K	102K
$MAS_{round=1}$	1.4M	355K	8.5M	1.9M	1.1M	390K	1.1M	218K	1.9M	402K	581K	134K	2.4M	567K
$MAS_{round=T}$	1.6M	387K	16M	3.4M	2.4M	745K	2.1M	388K	3.7M	721K	2.6M	492K	4.7M	1.0M
AgentPrune	1.3M	367K	15M	3.6M	2.0M	759K	1.9M	393K	3.4M	714K	1.5M	465K	4.2M	1.0M
AgentDropout	1.1M	333K	12M	2.8M	1.3M	634K	1.4M	312K	2.6M	594K	1.1M	359K	3.3M	839K
Base model: Qwen2.5-72B-Instruct														
Vanilla	99K	42K	142K	232K	32K	86K	16K	14k	27K	18K	91K	24K	68K	69K
CoT	129K	71K	154K	340K	34K	113K	18K	28k	30K	56K	93K	26K	76K	106K
$MAS_{round=1}$	966K	315K	9.0M	1.8M	1.4M	503K	1.1M	168K	1.8M	293K	271K	42K	2.4M	520K
$MAS_{round=T}$	2.2M	663K	18M	3.5M	2.9M	988K	2.0M	314K	3.4M	546K	814K	121K	4.9M	1.0M
AgentPrune	2.0M	665K	15M	3.3M	2.4M	995K	1.8M	313K	3.1M	517K	512K	83K	4.1M	979K
AgentDropout	1.3M	456K	11M	2.6M	2.0M	779K	1.6M	234K	2.4M	466K	421K	77K	3.1M	769K
Base model: Deepseek-V3-671B-Instruct														
Vanilla	99K	39K	137K	233K	30K	79K	16K	14K	26K	28K	92K	27K	67K	70K
CoT	129K	65K	148K	289K	34K	98K	17K	23K	29K	50K	93K	27K	75K	92K
$MAS_{round=1}$	845K	247K	7.9M	1.6M	1.2M	444K	951K	157K	1.6M	258K	339K	71K	2.1M	463K
$MAS_{round=T}$	2.1M	575K	16M	3.2M	2.7M	915K	1.9M	285K	3.0M	456K	1.3M	286K	4.5M	953K
AgentPrune	1.6M	582K	13M	3.2M	2.2M	902K	1.7M	296K	2.9M	498K	649K	206K	3.7M	947K
AgentDropout	1.0M	328K	11M	2.7M	1.6M	847K	1.4M	237K	2.3M	399K	469K	169K	3.0M	780K

Table 7: Token consumption comparison between AgentDropout and other baseline reasoning techniques. **Ptok.** denotes the number of prompt tokens for the agents, while **Ctok.** represents the number of completion tokens generated by the agents.

Task (1/12) & 🎓 Knowledgeable Expert 3FeF: Round 1 (2/12)

The Rydberg equation v = $R_H(1/n_1^2 - 1/n_2^2)$ accurately predicts the UV-visible emission spectrum of the hydrogen atom. A form of the Rydberg equation may also be used to predict the UV-visible emission for all of the following EXCEPT
Option A: hydride ion, H-
Option B: deuterium atom, D
Option C: tritium atom, T
Option D: helium cation, He+

A fascinating topic!

To solve this problem, we need to search for entities related to the Rydberg equation and atomic properties. Here are the key entities to search in Wikipedia:

@Rydberg equation@
@Hydrogen atom@
@Hydride ion (H-)@
@Deuterium atom (D)@
@Tritium atom (T)@
@Helium cation (He+)@
These entities will provide us with the necessary information to determine which option(s) is/are incorrect.

As a critic, I'll analyze the output of Agent 3FeF and identify potential issues:

1. Missing analysis: The output doesn't provide any actual analysis or reasoning for why the Rydberg equation might not apply to the mentioned options. It merely states that searching for entities related to the Rydberg equation and atomic properties is necessary, without actually applying those concepts to the problem.
2. Lack of specific mention: The output doesn't mention a specific aspect of the Rydberg equation that might not be applicable to the mentioned options. It's unclear what part of the equation might be affected by the changes in atomic properties.
3. Overemphasis on searching: While searching for relevant Wikipedia articles is a good idea, it's not enough to solve the problem. The output could benefit from a more in-depth analysis of the Rydberg equation and its applications to different atoms.
4. Unclear role: Agent 3FeF claims to be a "Knowledgeable Expert" but doesn't demonstrate any expert-level knowledge or critical thinking in their output.

To improve the analysis, Agent 3FeF could provide a more comprehensive approach, such as:

1. Identifying specific aspects of the Rydberg equation that might not be applicable to the mentioned options (e.g., the assumption of a single electron or a specific type of atomic orbital)
2. Providing examples or analogies to illustrate why the Rydberg equation might not work for certain options
3. Offering alternative solutions or approaches to understanding the emission spectra of the mentioned atoms
4. Explicitly stating the limitations and assumptions involved in applying the Rydberg equation to different atoms

By addressing these issues, Agent 3FeF can enhance the quality of their analysis and provide a more robust solution to the problem.

Thank you for the feedback from Agent 6PWE. As a mathematician, I must agree that my previous output was lacking in depth and analysis. Here's my revised answer:

The Rydberg equation, $v = R_H(1/n_1^2 - 1/n_2^2)$, accurately predicts the UV-visible emission spectrum of the hydrogen atom because it assumes a single electron in a hydrogen-like atom. However, when applying this equation to other atoms, such as hydride ion (H-), deuterium atom (D), tritium atom (T), and helium cation (He+), we must consider the following limitations:

1. The Rydberg equation assumes a single electron, which is not the case for the hydride ion (H-), which has two electrons. This means that the equation cannot accurately predict the emission spectrum of the hydride ion.
2. Deuterium (D) and tritium (T) are isotopes of hydrogen, which have slightly different atomic properties. While the Rydberg equation can still be applied to these atoms, the predicted emission spectra will be slightly different due to the changes in atomic properties.

On the other hand, the Rydberg equation can be applied to the helium cation (He+) because it assumes a single electron, and helium is a noble gas that has a single electron in its outermost orbital.

In conclusion, the Rydberg equation is not applicable to the hydride ion (H-) because of its assumption of a single electron. The equation can be applied to deuterium (D) and tritium (T) with minor modifications to account for their atomic properties, but the predicted emission spectra will be slightly different.

Thank you again for the feedback from Agent 6PWE. I hope this revised answer addresses the issues raised and provides a more comprehensive analysis of the Rydberg equation and its applications to different atoms.

Psychologist vtXr: Round 1 (5/12)

What a delightful exercise in critical thinking and problem-solving! As a psychologist, I'd like to offer some insights from a psychological perspective.

Firstly, I'd like to commend Agent 3FeF for acknowledging the limitations of their initial output and recognizing the need for a more comprehensive approach. This is a crucial aspect of critical thinking, and it's excellent that they're willing to learn from their mistakes.

Secondly, Agent 32fy has provided an outstanding revised analysis that addresses the issues raised by Agent 6PWE. Their output is now more detailed, specific, and evidence-based, demonstrating a clear understanding of the Rydberg equation and its limitations. The examples they provided to illustrate the applicability of the equation to different atoms are excellent, and the conclusion is well-supported by the logic of the analysis.

From a psychological perspective, I'd like to highlight the importance of self-awareness and humility in critical thinking. By acknowledging their own limitations and being open to feedback, both Agents 3FeF and 32fy have demonstrated a high level of self-awareness. Furthermore, Agent 32fy's willingness to revise their output and provide a more comprehensive analysis shows an excellent grasp of the concept of constructive feedback.

In conclusion, as a psychologist, I'm particularly pleased to see such a high level of critical thinking and problem-solving skills on display. I hope that these agents will continue to refine their skills and apply them to even more complex problems in the future.

Historian 53AY: Round 1 (6/12) & Knowlegable Expert 3FeF: Round 2 (7/12)

None.

Let's review the outputs of the other agents and identify potential issues:

Agent 53AY:

1. The initial output is correct that the Rydberg equation predicts the UV-visible emission spectrum of the hydrogen atom.
2. However, the reason provided for why the Rydberg equation is not applicable to the hydride ion (H-) is limited. The equation is not only affected by the number of electrons but also by the atomic properties of the atom, such as the mass of the nucleus and the electron configuration.
3. The output does not consider the possibility of applying the Rydberg equation to deuterium, tritium, and helium cation with some modifications.

Agent 3FeF:

1. The output is incomplete and lacks specific analysis. It merely states that the entities should be searched in Wikipedia without providing any relevant information or insights.

Agent 32fy:

1. The initial output is lacking in depth and analysis, as acknowledged by the agent themselves.
2. The revised output provides a more detailed analysis, but it still has some limitations:
+ It implies that the Rydberg equation can't be applied to deuterium and tritium at all, which is not entirely accurate. With some modifications, the equation can still be applied to these isotopes.
+ The conclusion that the Rydberg equation is not applicable to the hydride ion (H-) due to its assumption of a single electron is correct, but the explanation could be more detailed.
+ The analysis could be more comprehensive by considering the atomic properties of helium cation (He+) and why the Rydberg equation is applicable to it.

In summary, while the agents have made some efforts to analyze the topic, they still lack a thorough understanding of the Rydberg equation and its limitations. A more in-depth analysis would consider the atomic properties of the atoms, the modifications required for deuterium, tritium, and helium cation, and the resulting differences in the predicted emission spectra.

I understand that the task is to identify which option is incorrect with regards to the applicability of the Rydberg equation to predict UV-visible emission spectrum.

After reviewing the outputs of other agents, I agree with Agent 6PWE that Agent 3FeF's output lacks analysis and specific information, merely suggesting searching for Wikipedia articles without applying the concepts to the problem.

Based on the Rydberg equation, $v = R_H(1/n_1^2 - 1/n_2^2)$, and the atomic properties of the atoms, I believe that the Rydberg equation is not applicable to the hydride ion (H-). This is because the equation assumes a single electron, which is not the case for negatively charged ions like H-.

nHowever, the Rydberg equation can be applied to deuterium, tritium, and helium cation with some modifications, considering the changes in atomic properties. This is because these atoms have different electron configurations and masses of the nucleus.

Therefore, the correct answer is Option A: hydride ion, H-.

As for Agent 32fy's revised output, I agree that it is more comprehensive and detailed than Agent 3FeF's output. However, I would like to point out that the analysis could still be improved by considering the atomic properties of helium cation (He+) and why the Rydberg equation is applicable to it.

Overall, I believe that my analysis takes into account the limitations of the Rydberg equation and provides a more comprehensive understanding of its applicability to different atoms.

Thank you for sharing the outputs of the other agents. As a psychologist, I'd like to offer a more nuanced and scientific approach to this problem.

The Rydberg equation, $v = R_H(1/n_1^2 - 1/n_2^2)$, is a fundamental concept in atomic physics. It's not just a simple formula, but rather a mathematical expression that describes the energy levels of an atom. The equation assumes that the atom has a single electron, which is a simplifying assumption.

Considering the atomic properties of the entities mentioned, we can analyze each option:

Option A: Hydride ion (H-). As mentioned, the Rydberg equation assumes a single electron, which is not the case for the hydride ion, as it has two electrons. Therefore, the equation is not directly applicable to the hydride ion.

Option B: Deuterium atom (D). Deuterium is an isotope of hydrogen, and its atomic mass and electron configuration are very similar to those of the hydrogen atom. With some modifications, the Rydberg equation can still be applied to deuterium, but the predicted emission spectrum would be slightly different due to the increased mass of the nucleus.

Option C: Tritium atom (T). Tritium is another isotope of hydrogen, and its properties are also similar to those of the hydrogen atom. Again, with some modifications, the Rydberg equation can be applied to tritium, but the predicted emission spectrum would be slightly different due to the increased mass of the nucleus.

Option D: Helium cation (He+). The Rydberg equation can be applied to the helium cation, as it assumes a single electron, which is the case for the helium cation.

In conclusion, while the Rydberg equation is applicable to all the options, the hydride ion (H-) is the only one where the equation is not directly applicable due to the assumption of a single electron. The other options - deuterium, tritium, and helium cation - can be analyzed using modified versions of the Rydberg equation that take into account the atomic properties of the atoms.

I hope this analysis provides a more comprehensive understanding of the Rydberg equation and its limitations.

The Rydberg equation, $v = R_H(1/n_1^2 - 1/n_2^2)$, accurately predicts the UV-visible emission spectrum of the hydrogen atom. A form of the Rydberg equation may also be used to predict the UV-visible emission for all of the following EXCEPT option A: hydride ion, H-.

The reason for this exception is that the hydride ion (H-) has two electrons, whereas the Rydberg equation assumes a single electron. This means that the equation cannot accurately predict the emission spectrum of the hydride ion.

A ✓