Latent Collaboration in Multi-Agent Systems

Jiaru Zou Xiyuan Yang Ruizhong Qiu Gaotang Li
Katherine Tieu Pan Lu Ke Shen Hanghang Tong
Yejin Choi Jingrui He James Zou Mengdi Wang Ling Yang

Presenter: Yicheng Tao

December 15, 2025

Outline

Overview

TLDR

A comprehensive model collaboration framework unifying both latent reasoning and latent communication.

Summary

- **Latent Reasoning**: Leverage hidden representations within transformers to enable single model's internal latent chain-of-thought reasoning.
- **Latent Communication**: Employ KV caches for information exchange across two models.

Key Feature

- Training-free
- Lower computational complexity
- Higher performance (Need check)

Preliminary

Notation

Let W_{in} denote the input embedding layer, W_{out} denote the language model head, $E = [e_1, e_2, \ldots, e_t] \in \mathbb{R}^{t \times d_h}$ denote the input token embeddings and $H = [h_1, h_2, \ldots, h_t] \in \mathbb{R}^{t \times d_h}$ denote the final-layer hidden representations.

KV Cache

In decoder-only Transformers, the Key-Value (KV) cache functions as a **dynamic working memory** during auto-regressive generation, storing intermediate representations from previous decoding steps to avoid redundant computation. Specifically, when the next token at step $t + 1$ is generated, the KV cache at each layer $(K_{\text{cache}}, V_{\text{cache}})$ will be updated as:

$$K_{\text{cache}} \leftarrow [K_{\leq t}; K_{t+1}], \quad V_{\text{cache}} \leftarrow [V_{\leq t}; V_{t+1}]$$

System Overview

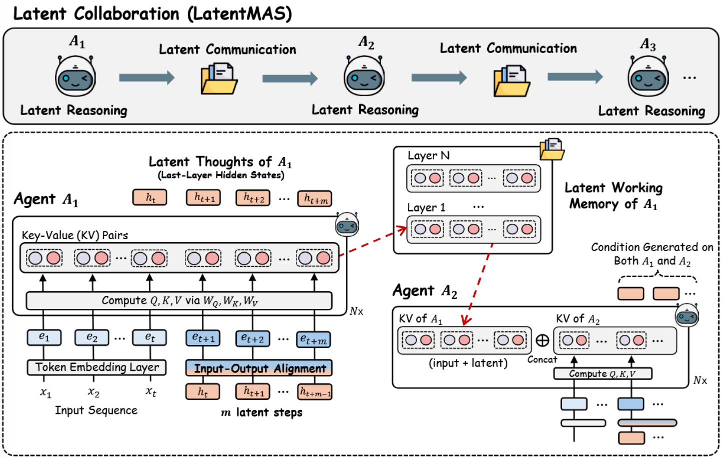

Figure 3 | **Overview of LatentMAS.** Each LLM agent in the system first generates latent thoughts through last-layer hidden states, then transfers information layer-wise via shared latent working memory stored in KV-caches, enabling completely system-wide latent collaboration.

Latent Reasoning

Generation Procedure

Given the input embeddings $E = [e_1, e_2, \ldots, e_t]$ and the obtained last-layer hidden representation h_t at step t, insert h_t into the input embedding instead of decoding and next-token embedding. m latent steps yield a sequence of newly generated last-layer hidden states $H = [h_{t+1}, h_{t+2}, \ldots, h_{t+m}]$, which is defined as **latent thoughts**.

Input-Output Distribution Alignment

To mitigate the distribution shift between input token embedding and last-layer hidden state, a linear alignment operator is employed. A projection matrix $W_a \in \mathbb{R}^{d_h \times d_h}$ maps output vector $h \in H$ to a new input vector e.

$$e = hW_a, \quad \text{where}\, W_a \approx W_{\text{out}}^{-1} W_{\text{in}}$$

Latent Communication

Latent Working Memory

Let A_1, A_2 be two agents. A_1 first performs m latent steps and obtain KV Cache or latent memory $\mathcal{M}_{A_1}$.

$$\mathcal{M}_{A_1} = \left\{ \left(K_{A_1,\text{cache}}^{(l)}, V_{A_1,\text{cache}}^{(l)} \right) \mid l = 1, 2, \ldots, L \right\}$$

Information Transfer

Layer-wise concatenate KV caches from $\mathcal{M}_{A_1}$ to A_2 by prepending each $K_{A_1,\text{cache}}^{(l)}$ and $V_{A_1,\text{cache}}^{(l)}$ to $K_{A_2,\text{cache}}^{(l)}$ and $V_{A_2,\text{cache}}^{(l)}$.

Multi-agent Setting

Sequential

Chain-of-agents design: planer, critic, refiner, solver (judger).

Hierarchical

Domain-specialized design: code, math, science agents to a summarizer.

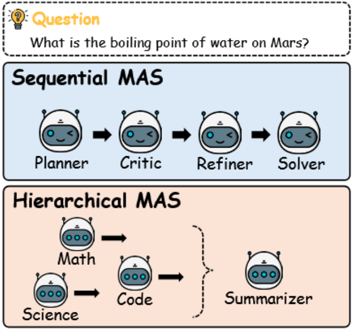

Figure 2 | Illustration of sequential and hierarchical MAS.

Results

Table 1 | **Main results of LatentMAS on 6 general tasks under the Sequential MAS setting.** We report 3 metrics in total, including task accuracy (%, **"Acc."**), total output token usage (**"Token"**), and end-to-end inference speed (time(s) / run, **"Speed"**). We compare LatentMAS with both TextMAS and single-model ("Single") baselines. For each metric, we **bold** the better performance and visualize LatentMAS gains over TextMAS in the **Improve** columns.

Tasks	Metrics	Qwen3-4B			Improve	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Sequential MAS Setting													
ARC-E	Acc.	95.4	96.4	**98.6**	↑ 2.2	95.6	**99.1**	98.8	↓ 0.3	97.2	99.0	**99.4**	↑ 0.4
	Token	724	2420	**581**	↓ 76.0%	656	2085	**490**	↓ 76.5%	608	1670	**224**	↓ 86.6%
	Speed	369	2874	**512**	×5.6	404	3702	**1759**	×2.1	551	9171	**2124**	×4.3
ARC-C	Acc.	89.2	90.0	**92.3**	↑ 2.3	91.0	**94.6**	94.4	↓ 0.2	92.6	**95.9**	95.6	↓ 0.3
	Token	913	2678	**718**	↓ 73.2%	846	2252	**529**	↓ 76.5%	773	2985	**426**	↓ 85.7%
	Speed	97	1579	**260**	×6.1	266	2059	**703**	×2.9	338	5125	**1136**	×4.5
GSM8K	Acc.	82.4	**89.8**	88.2	↓ 1.6	81.1	92.3	**93.8**	↑ 1.5	83.7	93.8	**95.2**	↑ 1.4
	Token	1136	3172	**607**	↓ 80.9%	1280	2324	**860**	↓ 63.0%	1118	3324	**644**	↓ 80.6%
	Speed	469	1970	**375**	×5.3	449	1739	**543**	×3.2	536	3729	**1952**	×1.9
MedQA	Acc.	47.7	65.3	**66.3**	↑ 1.0	53.0	75.0	**75.3**	↑ 0.3	64.7	80.3	**80.7**	↑ 0.4
	Token	2134	3962	**1685**	↓ 57.5%	2098	4260	**1555**	↓ 63.5%	1746	3444	**1841**	↓ 46.5%
	Speed	236	1267	**438**	×2.9	476	1923	**928**	×2.1	1360	4142	**1420**	×2.9
MBPP+	Acc.	63.5	69.8	**73.5**	↑ 3.7	64.8	69.5	**74.6**	↑ 5.1	68.5	72.8	**75.7**	↑ 2.9
	Token	1634	4420	**1339**	↓ 69.7%	2053	3695	**1164**	↓ 68.5%	1858	4971	**1621**	↓ 67.4%
	Speed	523	2148	**577**	×3.7	1064	3628	**1275**	×2.8	2410	8728	**2400**	×3.6
HumanEval+	Acc.	75.0	79.7	**79.9**	↑ 0.2	74.4	80.5	**80.5**	↑ 0.0	76.8	81.1	**86.5**	↑ 5.4
	Token	2380	5987	**1775**	↓ 70.4%	2507	4593	**1866**	↓ 59.4%	2366	5934	**2042**	↓ 65.6%
	Speed	274	1044	**350**	×3.0	502	1619	**497**	×3.3	1084	4062	**1285**	×3.2

Table 2 | **Main results of LatentMAS on 6 general tasks under the Hierarchical MAS setting.** We report accuracy, token usage, and end-to-end speed, and highlight the performance gains following the same evaluation protocol as in Table 1.

Tasks	Metrics	Qwen3-4B			Improve	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Hierarchical MAS Setting													
ARC-E	Acc.	95.4	97.1	96.8	↓ 0.3	95.6	98.2	**98.3**	**↑ 0.1**	97.2	98.3	**98.7**	**↑ 0.4**
	Token	724	2054	**363**	**↓ 82.3%**	656	2237	**308**	**↓ 86.2%**	608	2752	**619**	**↓ 77.5%**
	Speed	369	2239	**591**	**×3.8**	404	3619	**1779**	**×2.0**	551	7102	**1884**	**×3.8**
ARC-C	Acc.	89.2	92.5	91.7	↓ 0.8	91.0	93.3	**93.9**	**↑ 0.6**	92.6	95.3	**95.5**	**↑ 0.2**
	Token	913	2674	**447**	**↓ 83.3%**	846	2854	**344**	**↓ 87.9%**	773	2167	**295**	**↓ 86.4%**
	Speed	97	1275	**299**	**×4.3**	266	2034	**714**	**×2.8**	338	4283	**1090**	**×3.9**
GSM8K	Acc.	82.4	89.4	88.4	↓ 1.0	81.1	90.4	89.5	↓ 0.9	83.7	90.8	**91.6**	**↑ 0.8**
	Token	1136	3098	**555**	**↓ 82.1%**	1280	2370	**353**	**↓ 85.1%**	1118	3021	**495**	**↓ 83.6%**
	Speed	469	1878	**360**	**×5.2**	449	1365	**702**	**×1.9**	536	3675	**1631**	**×2.3**
MedQA	Acc.	47.7	65.0	**67.3**	**↑ 2.3**	53.0	76.3	**77.0**	**↑ 0.7**	64.7	78.0	**78.3**	**↑ 0.3**
	Token	2134	6702	**1015**	**↓ 84.9%**	2098	6893	**1007**	**↓ 85.4%**	1746	5473	**899**	**↓ 83.6%**
	Speed	236	1495	**557**	**×2.7**	476	3387	**964**	**×3.5**	1360	7591	**1250**	**×6.1**
MBPP+	Acc.	63.5	69.3	**70.6**	**↑ 1.3**	64.8	71.9	**72.2**	**↑ 0.3**	68.5	73.0	**73.8**	**↑ 0.8**
	Token	1634	6782	**1339**	**↓ 80.3%**	2053	7703	**1264**	**↓ 83.6%**	1858	7458	**1187**	**↓ 84.1%**
	Speed	523	1766	**489**	**×3.6**	1064	3898	**1387**	**×2.8**	2410	9162	**2507**	**×3.7**
HumanEval+	Acc.	75.0	76.2	**79.3**	**↑ 3.1**	74.4	76.8	**78.0**	**↑ 1.2**	76.8	84.1	**86.6**	**↑ 2.5**
	Token	2380	8127	**1373**	**↓ 83.1%**	2507	8768	**1274**	**↓ 85.5%**	2366	8114	**1512**	**↓ 81.4%**
	Speed	274	931	**333**	**×2.8**	502	1809	**439**	**×4.1**	1084	3988	**1188**	**×3.4**

Table 3 | **Main results of LatentMAS on 3 reasoning-intensive tasks under both Sequential and Hierarchical MAS settings.** We report accuracy, token usage, and end-to-end speed, and highlight the performance gains following the same evaluation protocol as in Table 1.

Tasks	Metrics	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Sequential MAS Setting									
AIME24	Acc.	50.0	53.3	**56.7**	**↑ 3.4**	63.3	63.3	**66.7**	**↑ 3.4**
	Token	12891	38596	**8953**	**↓ 76.8%**	11263	32092	**10593**	**↓ 67.0%**
	Speed	421	2808	**688**	**×4.1**	1018	4554	**1149**	**×4.0**
AIME25	Acc.	46.7	53.3	**53.3**	**↑ 0.0**	56.7	60.0	**63.3**	**↑ 3.3**
	Token	14692	45088	**8699**	**↓ 80.7%**	11298	44618	**11402**	**↓ 74.4%**
	Speed	450	3150	**820**	**×3.8**	1040	5184	**1473**	**×3.5**
GPQA-Diamond	Acc.	39.9	43.4	**45.5**	**↑ 2.1**	48.5	51.5	**52.0**	**↑ 0.5**
	Token	6435	17986	**4571**	**↓ 74.6%**	5547	12676	**5454**	**↓ 57.0%**
	Speed	813	5771	**854**	**×6.8**	1043	9714	**1475**	**×6.6**
Hierarchical MAS Setting									
AIME24	Acc.	50.0	53.3	53.3	↑ 0.0	63.3	70.0	**73.3**	**↑ 3.3**
	Token	12891	42629	**7526**	**↓ 82.3%**	11263	29025	**10230**	**↓ 64.8%**
	Speed	421	3132	**776**	**×4.0**	1018	5718	**1089**	**×5.3**
AIME25	Acc.	46.7	50.0	50.0	↑ 0.0	56.7	66.7	66.7	↑ 0.0
	Token	14692	53929	**13230**	**↓ 75.5%**	11298	50003	**9527**	**↓ 80.9%**
	Speed	450	3488	**616**	**×5.7**	1040	6019	**1056**	**×5.7**
GPQA-Diamond	Acc.	39.9	43.0	**46.9**	**↑ 3.9**	48.5	52.0	**53.0**	**↑ 1.0**
	Token	6435	22450	**3395**	**↓ 84.9%**	5547	20931	**3606**	**↓ 82.8%**
	Speed	813	6108	**798**	**×7.7**	1043	9119	**1458**	**×6.3**

Study on Input-Output Alignment

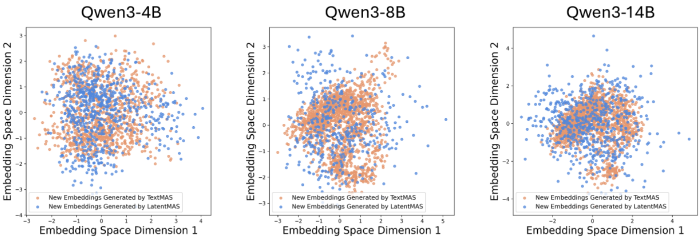

Figure 5 | **Illustration of the semantic meaning encoded by latent thoughts in LatentMAS.** Newly generated latent thought embeddings in LatentMAS largely cover the embedding space of text-based generated tokens, indicating semantic consistency and greater expressive capacity than discrete text.

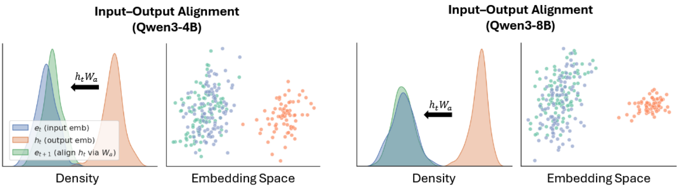

Figure 6 | **Effectiveness of the input-output alignment W_a on MedQA.** Unaligned output embeddings (h_t) drift away from the original input embeddings (e_t), while the aligned vectors (e_{t+1}) realign with e_t, demonstrating that W_a preserves embedding-space structure and prevents representation drift.

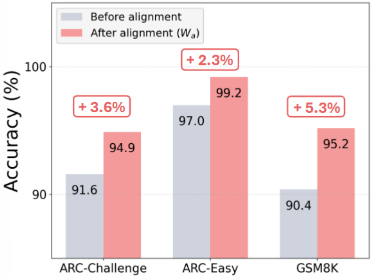

Figure 7 | Downstream performance before/after applying the input-output alignment W_a.

m	0	10	20	40	80	160
ARC-C	91.3	93.4	93.4	**94.9**	94.8	93.7
ARC-E	94.7	98.9	98.9	99.4	**99.6**	98.3
GSM8K	85.6	90.3	90.9	91.4	**92.0**	91.9

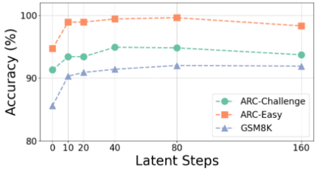

Figure 8 | Effectiveness of different latent step depths of LatentMAS on downstream performance.

Expressiveness of Latent Thoughts

Theorem 3.1 (Expressiveness of Latent Thoughts). *Under the Linear Representation Hypothesis on h (detailed in Assumption B.1), if the sequence of all latent thoughts with length m can be expressed losslessly through corresponding text-based reasoning, then the length of text (in tokens) needs to be at least $\Omega(d_h m / \log|\mathcal{V}|)$, where $|\mathcal{V}| > 1$ denotes the vocabulary size.*

Remark 3.2. Theorem 3.1 suggests that latent thoughts generation can be $O(d_h / \log|\mathcal{V}|)$ times more efficient than text-based reasoning. In addition, the expressiveness scales linearly with d_h, implying that larger models inherently exhibit greater latent reasoning capacity.

- This theorem suggests that, if the latent thoughts can be limited to a fixed set (combination of semantic basis), then to represent all latent thoughts with text token embeddings in a limited vocabulary size, there should exist a combination method.
- This theorem is only meaningful when we assume that multiple text token embeddings can be equivalent to a latent embedding during generation.

Reproduction Issue

The results in the table can not be fully reproduced. Here is my results on AIME 2024 under sequential setting:

- Single agent: 76.7%
- Text-MAS: 43.3%
- Latent-MAS (10 step): 66.7%
- Latent-MAS (20 step): 46.7%

Analysis

- Due to the uncertainty in decoding, latent reasoning is more determinstic than text token.
- The multi-agent setting is not reasonable, which causes lower performance than baseline.
- The latent steps, as a hyperparameter, have great impacts on the final performance.

Conclusion

Contribution

- Build a framework that combines latent reasoning and latent communication.
- Propose the input-output alignment method.

Problems

- The reproduction issue should be investigated.
- As an agent system, latent reasoning can not coexist with tool using. The workflow is not dynamic.