LEARNING TO COMMUNICATE: TOWARD END-TO-END OPTIMIZATION OF MULTI-AGENT LANGUAGE SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems built on large language models have shown strong performance on complex reasoning tasks, yet most work focuses on agent roles and orchestration while treating inter-agent communication as a fixed interface. Latent communication through internal representations such as key–value caches offers a promising alternative to text-based protocols, but existing approaches do not jointly optimize communication with multi-agent reasoning. Therefore we propose DiffMAS, a training framework that treats latent communication as a learnable component of multi-agent systems. DiffMAS performs parameter-efficient supervised training over multi-agent latent trajectories, enabling agents to jointly learn how information should be encoded and interpreted across interactions. Across five backbones and six mathematical reasoning, scientific QA, code generation, and commonsense benchmarks, DiffMAS matches or exceeds single-agent inference, text-based multi-agent systems, and prior latent communication methods in every setting, achieving +26.7% on AIME24 and +20.2% on GPQA-Diamond, and it also improves decoding stability. Against single-agent and text-based pipelines fine-tuned on identical data with the same recipe, DiffMAS is better in 26 of 30 settings and never worse, while fine-tuning through the text channel lowers accuracy in 13 of 30 settings, consistent with our analysis that text communication lets training adapt only how messages are read, not how they are written.

1 INTRODUCTION

Multi-agent systems (MAS) built on large language models have emerged as a powerful paradigm for solving complex reasoning tasks, including mathematics, programming, and commonsense problem solving (Zhang et al., 2025a;b). Their effectiveness is commonly attributed to two complementary factors: *(i) system structure and agent role assignment* (Chen et al., 2024), which determine how agents are organized and specialized, and *(ii) inter-agent communication* (Chen et al., 2025; Shen et al., 2025), which governs how intermediate reasoning states and information are exchanged. By decomposing complex tasks (Wu et al., 2023; Hong et al., 2024; Qian et al., 2024; Li et al., 2023), multi-agent systems can outperform single-agent baselines. They do so when the benefit of decomposition, where each agent focuses on a narrower subtask and later agents check earlier ones, outweighs the cost of passing intermediate results between agents. That cost is set by the communication channel, which is the subject of this paper.

As multi-agent language systems continue to mature, recent research has increasingly explored how to systematically improve their performance through optimization (Wang et al., 2024; Zhang et al., 2025a;b; Chen et al., 2024). Existing efforts have primarily focused on improving agentic capabilities through fine-tuning or prompting, as well as refining workflow design through better role assignment, planning strategies, and execution structures (Shinn et al., 2023; Yao et al., 2023; Zhang et al., 2025b; Chen et al., 2024). These advances have led to more capable agents and more effective system structures.

However, such improvements largely treat multi-agent systems as modular compositions of optimized components, rather than systems whose interactions can themselves be learned (Zhuge et al., 2024; Shen et al., 2025; Chen et al., 2025). A natural next step is therefore to explore whether multi-agent

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

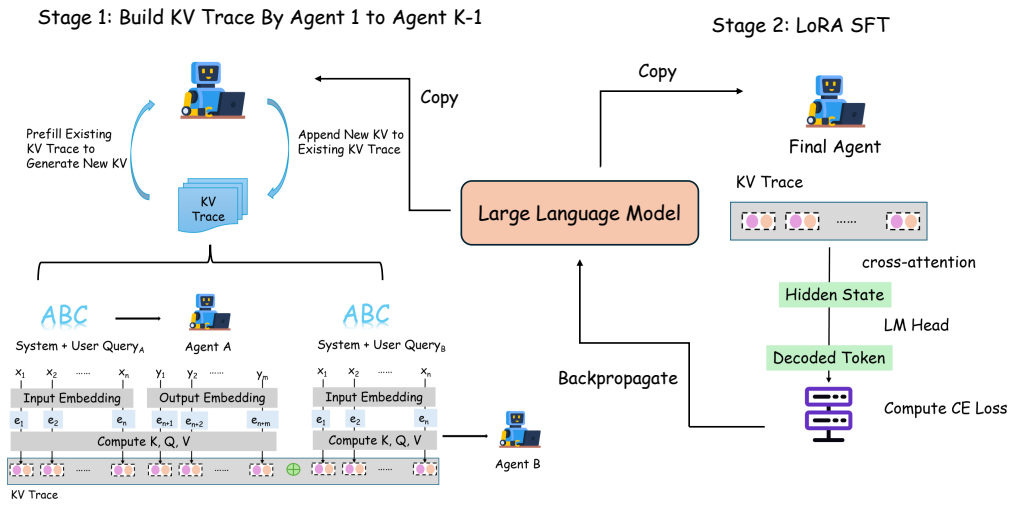


Figure 1: In Stage I, agents 1 to K–1 sequentially construct a shared KV trace by prefilling the existing cache and appending newly generated KV segments. These agents receive no separate supervision, but their KV segments stay in the computation graph. The accumulated KV trace serves as a latent communication medium across agents. In Stage II, the final agent performs autoregressive decoding on the prefilled KV cache. Attention over the KV trace produces hidden states, which are projected through the LM head to generate tokens. A single cross-entropy loss is applied to the final agent's output. Its gradient is backpropagated through the entire KV trace into one LoRA adapter shared by all agents, so the update changes both how upstream agents write the trace and how the final agent reads it. The backbone stays frozen.

systems can be optimized more holistically, allowing not only agent behaviors and workflows, but also their interactions to improve through learning (Motwani et al., 2025; Ma et al., 2025). Achieving this vision requires moving beyond fixed system interfaces toward interaction mechanisms that can be optimized together with agentic reasoning. As different tasks can place different demands on how intermediate reasoning states should be represented and preserved, effective communication strategies need to adapt to the task rather than relying on fixed protocols (Chen et al., 2025; Shen et al., 2025). In this context, enabling communication to become a learnable component represents a critical step towards more fully optimizable multi-agent reasoning systems.

A key obstacle to making communication learnable lies in the discrete nature of how agents currently exchange information. In most existing multi-agent systems, intermediate reasoning results must be decoded into natural language before being transmitted to downstream agents (Chen et al., 2023; Hong et al., 2024). While this design aligns naturally with the token-based interfaces of language models, it introduces a fundamental limitation: discrete message passing creates optimization boundaries between agents (Sukhbaatar et al., 2016; Foerster et al., 2016; Mordatch & Abbeel, 2018; Havrylov & Titov, 2017). This limitation is particularly restrictive as the performance of the system depends critically on how well intermediate reasoning results are preserved and interpreted across agents (Deng et al., 2023; Wei et al., 2023). More fundamentally, this reflects a mismatch between the continuous internal representations used during reasoning and the discrete messages used for inter-agent exchange (Zelikman et al., 2024; Hao et al., 2025; Fu et al., 2026; Zou et al., 2025). To make communication truly learnable and enable more end-to-end optimization of multi-agent systems, inter-agent information exchange must therefore operate over continuous representations that can support differentiable information flow.

Key–Value (KV) caches provide a natural mechanism for enabling such differentiable communication (Vaswani et al., 2023). As continuous internal representations produced by transformer attention layers, KV states can be directly reused by downstream agents without requiring discrete decoding or re-encoding (Kwon et al., 2023). This allows intermediate reasoning representations to be transmitted through a continuous channel that remains compatible with gradient-based learning. By treating KV caches as a latent communication interface, gradients can propagate not only within individual agents but also across agent boundaries, allowing communication behaviors to be optimized together with the underlying model (Motwani et al., 2025; Zhuge et al., 2024; Ma et al., 2025).

Therefore, we propose DiffMAS, a framework that treats KV-mediated latent communication as a trainable interface between agents and enables SFT-like training over multi-agent interaction trajectories. As illustrated in Figure 1, DiffMAS operates in two stages. In Stage I, upstream agents sequentially construct a shared KV trace by prefilling existing KV caches and appending newly generated KV segments, forming a continuous latent communication channel. In Stage II, the final agent performs autoregressive decoding conditioned on the accumulated KV trace, and supervised training is applied to optimize communication jointly with downstream reasoning. By training over full multi-agent interaction trajectories, the framework enables the model to learn communication with agentic reasoning. Our contributions are summarized as follows:

- We formulate inter-agent communication as an optimization problem in multi-agent language systems. We show that under text or otherwise non-differentiable communication, training can adapt only how the final agent interprets messages, whereas an append-only KV trace retains every agent's contribution without contraction and passes a depth-independent gradient to how upstream agents encode it (Proposition 3.1).
- We propose DiffMAS, a supervised training framework that treats KV-mediated latent communication as a differentiable and task-adaptive interface between agents.
- Across five backbones and six benchmarks, DiffMAS matches or exceeds all baselines in every setting, with gains of up to +26.7% on AIME24. Against single-agent and text-based pipelines fine-tuned on identical data, it is better in 26 of 30 settings and never worse, and it yields more stable decoding.

2 Related Work

2.1 Multi-Agent LLM Systems

LLM-based multi-agent systems improve complex reasoning by decomposing problems across specialized agents and structured interaction workflows. Prior work has explored improving agent capabilities and system design through role engineering, automated workflow construction, and architecture search (Wu et al., 2023; Li et al., 2023; Qian et al., 2024; Hong et al., 2024; Chen et al., 2024; Zhang et al., 2025b;a). Other studies focus on improving coordination through execution policies or self-configuration (Chen et al., 2025; Wang et al., 2025; Motwani et al., 2025; Ma et al., 2025).

Despite these advances, most multi-agent systems rely on natural language as a universal communication interface (Zhang et al., 2025b; Chen et al., 2023). While flexible, text-based communication requires internal reasoning states to be serialized into discrete tokens (Wei et al., 2023), which can limit information fidelity and prevent communication from being optimized jointly with agent reasoning (Zelikman et al., 2024; Zhuge et al., 2024; Ma et al., 2025). These limitations raise the question of whether richer internal representations could provide better communication mechanisms.

2.2 Latent Reasoning

In parallel, a growing body of work investigates latent or implicit reasoning in large language models, aiming to reduce reliance on explicit textual rationales. Implicit Chain-of-Thought distillation (Deng et al., 2023) shows that reasoning behaviors can be internalized into hidden representations, while approaches such as Quiet-STaR (Zelikman et al., 2024) and continuous latent reasoning frameworks (Hao et al., 2025; Zhu et al., 2025) demonstrate that models can reason effectively without emitting explicit thought tokens.

More recently, these ideas have been extended to multi-agent settings. Several works explore sharing internal representations—such as hidden states or KV caches—between agents to enable richer information exchange than text-based protocols (Zou et al., 2025; Fu et al., 2026; Du et al., 2026). Related studies analyze thought-level communication and aggregation across agents (Zheng et al., 2025; Fein-Ashley et al., 2025). These approaches demonstrate that latent communication can improve efficiency and expressivity by preserving fine-grained intermediate reasoning signals. However, existing latent communication methods either rely on training-free latent exchange (Zou et al., 2025) or learn representation alignment across models (Fu et al., 2026). In contrast, we

treat **communication itself as a learnable component and optimize it jointly with multi-agent reasoning.**

3 METHOD

3.1 PROBLEM SETTING AND OPERATOR VIEW

We propose DiffMAS, a framework for learning inter-agent communication in multi-agent language systems. We consider a system composed of K sequential agents that collaborate through intermediate reasoning steps. Unlike conventional MAS that treats communication as a fixed protocol, DiffMAS treats communication itself as part of the learnable computation.

From MAS to latent trace representation. Each agent produces intermediate latent representations consumed by later agents. Concretely, these are segments of key–value (KV) states generated during autoregressive decoding. We model communication as a sequence of latent blocks, where each stage appends a fixed number of blocks to a shared *latent trace*. This trace is the sole communication medium and is preserved without overwriting.

Latent block and trace spaces. Let $\mathcal{Z} \subseteq \mathbb{R}^d$ denote the latent block space, where each block $z_t \in \mathcal{Z}$ is a d-dimensional vector. Each stage emits T latent blocks. After j stages (i.e., after j agents have executed sequentially), the total number of blocks is $N_j \triangleq jT$. Thus, the stage-j trace space is $\mathcal{T}_j = \mathcal{Z}^{N_j}$, and $\mathcal{T}_0 \triangleq \{\mathbf{Z}_{1:0}\}$. We write the accumulated latent trace as $\mathbf{Z}_{1:N_j} = (z_1, \ldots, z_{N_j}) \in \mathcal{T}_j$.

Stage operators. For each stage $j \in \{1, \ldots, K\}$, let $A_\theta^{(j)}(\cdot; x, p_j) : \mathcal{T}_{j-1} \to \mathcal{T}_j$ denote a differentiable stage operator, where $x \in \mathcal{X}$ is the input and p_j is the stage-specific prompt, parameterized by θ, such that

$$\mathbf{Z}_{1:N_j} \;=\; A_\theta^{(j)}\big(\mathbf{Z}_{1:N_{j-1}};\, x, p_j\big). \tag{1}$$

The full trajectory is the composition $\mathbf{Z}_{1:N_K} = (A_\theta^{(K)} \circ \cdots \circ A_\theta^{(1)})(\mathbf{Z}_{1:0})$, so the entire multi-agent process is a single differentiable computation.

Within-stage dynamics. Inside each stage, $A_\theta^{(j)}$ runs T latent micro-steps. At step t it emits a block $z_t^{(j)}$ conditioned on x, p_j, the previous trace, and the blocks it has already emitted, and then updates its internal state residually. All maps are differentiable in θ; the explicit equations are in Appendix B.

3.2 TRAINING OBJECTIVE AND END-TO-END DIFFERENTIABILITY

After K stages, a decoder produces the output distribution $p_\theta\big(y|x, \{p_j\}_{j=1}^K\big) = \mathrm{Dec}_\theta(x, p_K, \mathbf{Z}_{1:N_K})$. Training minimizes the supervised negative log-likelihood $\mathcal{L}(\theta) = -\log p_\theta\big(y^\star \mid x, \{p_j\}_{j=1}^K\big)$. Since $\mathbf{Z}_{1:N_K}$ is obtained via compositions of differentiable operators, gradients of $\mathcal{L}$ propagate through all stages and micro-steps with respect to θ.

Practical instantiation. All stage operators and the decoder are instantiated with one pretrained transformer conditioned on stage-specific prompts. A single LoRA adapter shared by all stages is trained while the backbone stays frozen (Appendix F).

3.3 STRUCTURAL IMPLICATIONS OF NON-OVERWRITING COMMUNICATION

We compare DiffMAS with overwriting communication at the interface level. In overwriting systems, computation is transmitted through a fixed-dimensional carrier $h_j \in \mathbb{R}^{d_h}$ with update $h_j = F_\theta(h_{j-1}, x, p_j)$, so downstream stages access prior computation only through repeated re-encoding. DiffMAS instead lets stage j emit a segment $\mathbf{Z}^{(j)} \equiv \mathbf{Z}_{1:T}^{(j)} = G_\theta^{(j)}(\mathbf{Z}_{1:N_{j-1}})$ that conditions on the entire previous trace, and appends it:

$$\mathbf{Z}_{1:N_j} = [\mathbf{Z}_{1:N_{j-1}};\, \mathbf{Z}^{(j)}], \qquad \mathbf{Z}^{(j)} = [z_1^{(j)}, \ldots, z_T^{(j)}]. \tag{2}$$

Hence intermediate segments remain explicit in the final trace, while the overwriting carrier has fixed dimension d_h and the ambient dimension of $\mathbf{Z}_{1:N_j}$ grows as $N_j d$.

To compare the two interfaces on equal terms we use *total* derivatives, which include every path through later stages. Let $J_j \triangleq d\mathbf{Z}_{1:N_K}/d\mathbf{Z}^{(j)}$ be the total Jacobian of the final trace with respect to segment j, $\lambda_j \triangleq d\mathcal{L}/d\mathbf{Z}^{(j)}$ the total gradient reaching segment j, and $g_j \triangleq \partial\mathcal{L}/\partial\mathbf{Z}^{(j)}$ the gradient that the decoder places directly on segment j of the final trace (the j-th block of $\partial\mathcal{L}/\partial\mathbf{Z}_{1:N_K}$).

Proposition 3.1 (Interface-induced retention and gradient structure)**.** *Assume the overwriting transition is contractive, $\|\partial h_j/\partial h_{j-1}\|_2 \le \rho < 1$ for all j, and let θ be shared across stages. Then for every $j < K$:*

(i) ***Retention.*** *Overwriting contracts every stage: $\|dh_K/dh_j\|_2 \le \rho^{K-j}$. Concatenation never contracts: $\|J_j v\|_2 \ge \|v\|_2$ for every direction v, i.e., $\sigma_{\min}(J_j) \ge 1$.*

(ii) ***Gradient paths.*** *Overwriting: $\|d\mathcal{L}/dh_j\|_2 \le \rho^{K-j}\,\|d\mathcal{L}/dh_K\|_2$. Concatenation:*

$$\lambda_j \;=\; \underbrace{g_j}_{\textit{direct}} \;+\; \sum_{k>j} \Big(\frac{d\mathbf{Z}^{(k)}}{d\mathbf{Z}^{(j)}}\Big)^{\top} g_k, \tag{3}$$

where the direct term contains no product of stage Jacobians.

(iii) ***Shared parameters.*** *The gradient of the shared parameters decomposes over stages,*

$$\nabla_\theta \mathcal{L} \;=\; \nabla_\theta \mathcal{L}\big|_{\mathrm{Dec}} \;+\; \sum_{j=1}^{K} \big(\partial_\theta G_\theta^{(j)}\big)^{\top} \lambda_j. \tag{4}$$

Under overwriting, the stage-j term is at most $\|\partial_\theta F_\theta^{(j)}\|_2\,\rho^{K-j}\,\|d\mathcal{L}/dh_K\|_2$; under concatenation, it contains the depth-independent term $(\partial_\theta G_\theta^{(j)})^{\top} g_j$.

Implications. Part (i) states that the trace keeps every agent's contribution at full magnitude: a change in agent j's KV segment reaches the decoder undiminished, whereas a contractive carrier shrinks it by ρ^{K-j}. Part (ii) states that every upstream agent receives gradient through a path of length one, in addition to the paths through later agents. Part (iii) is the formal sense in which DiffMAS learns to *encode*: because the LoRA parameters are shared and every upstream emission receives the direct gradient g_j, the update to θ includes terms that change how upstream agents write the trace, not only how the final agent reads it. Under a contractive overwriting interface these upstream terms vanish geometrically, so training effectively adapts only the last stages. Under text communication, token sampling is non-differentiable, so $\lambda_j = 0$ for all upstream stages and only the interpretation term survives; the same holds if the upstream KV trace is detached. Proposition 3.1 is an interface-level result: it does not claim that the learned trace is semantically better, which we test empirically in Sections 4.2–5. The proof is in Appendix A.

Trade-offs. The direct term g_j is produced by the decoder's attention, and so is scaled by the attention mass the decoder places on segment j. Since these masses sum to at most one, a longer trace dilutes the signal each token receives (Remark A.2). Concatenation therefore replaces geometric decay ρ^{K-j} with attention dilution, which the decoder can reallocate but which grows with trace length. This is consistent with the effect of communication steps in Table 5.

4 Experiments

4.1 Experimental Setup

Tasks and Datasets. We conduct a comprehensive evaluation of DiffMAS across a diverse set of benchmarks spanning both reasoning-intensive and program synthesis tasks. Specifically, we evaluate on: (i) *Math and Science Reasoning*, including AIME 2024 (Maxwell-Jia, 2024), AIME 2025 (math-ai, 2025), and GPQA-Diamond (Rein et al., 2023), which require multi-step symbolic reasoning and precise numerical or categorical answers; (ii) *Commonsense Reasoning*, using OpenBookQA (Mihaylov et al., 2018), which tests structured reasoning over elementary scientific knowledge; and (iii) *Code Generation*, including HumanEval-Plus (Liu et al., 2023) and MBPP-Plus (Liu et al., 2023), which assess functional correctness and generalization in Python program synthesis.

Models and Baselines. We evaluate DiffMAS across multiple open-source large language models spanning diverse architectures and parameter scales. Our evaluation includes models from the Qwen3

Table 1: Accuracy (ACC, %) comparison across benchmarks. We report absolute accuracy and improvement over the single-agent baseline. Best results are in bold; ties are bolded jointly.

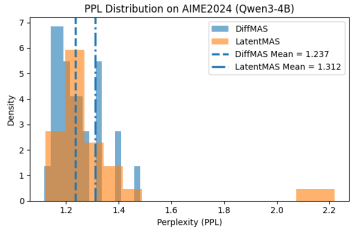
Model	Task	Text Communication		Training-free Latent Communication	Trained Latent Communication	
		Single	TextMAS	LatentMAS	C2C	DiffMAS
Qwen3-4B	AIME24	43.3%	46.7% (+3.4%)	50.0% (+6.7%)	0.0% (-43.3%)	**63.3% (+20.0%)**
	AIME25	43.3%	43.3% (+0.0%)	**50.0% (+6.7%)**	0.0% (-43.3%)	**50.0% (+6.7%)**
	GPQA-Diamond	42.4%	44.9% (+2.5%)	36.4% (-6.0%)	29.8% (-12.6%)	**46.4% (+4.0%)**
	HumanEval+	74.5%	76.3% (+1.8%)	78.9% (+4.4%)	33.3% (-41.2%)	**79.8% (+5.3%)**
	MBPP+	63.5%	69.3% (+5.8%)	70.6% (+7.1%)	43.9% (-19.6%)	**71.9% (+8.4%)**
	OpenBookQA	80.1%	81.8% (+1.7%)	77.6% (-2.5%)	79.4% (-0.7%)	**83.2% (+3.1%)**
Qwen3-8B	AIME24	50.0%	50.0% (+0.0%)	56.7% (+6.7%)	0.0% (-50.0%)	**76.7% (+26.7%)**
	AIME25	46.7%	53.3% (+6.6%)	53.3% (+6.6%)	0.0% (-46.7%)	**56.7% (+10.0%)**
	GPQA-Diamond	39.9%	43.4% (+3.5%)	45.5% (+5.6%)	41.4%(+1.5%)	**60.1% (+20.2%)**
	HumanEval+	74.5%	80.7% (+6.2%)	80.7% (+6.2%)	58.8%(-15.7%)	**81.5% (+7.0%)**
	MBPP+	64.8%	69.5% (+4.7%)	74.6% (+9.8%)	57.6%(-7.2%)	**74.8% (+10.0%)**
	OpenBookQA	83.6%	84.2% (+0.6%)	**85.8% (+2.2%)**	**85.8% (+2.2%)**	**85.8% (+2.2%)**

Table 2: Accuracy (%) comparison on additional backbones. We report absolute accuracy and improvement over the single-agent baseline. Best results are in bold; ties are bolded jointly.

Model	Task	Single	TextMAS	LatentMAS	DiffMAS
Ministral-3-8B	AIME24	50.0%	26.7% (-23.3%)	33.3% (-16.7%)	**73.3% (+23.3%)**
	AIME25	23.3%	23.3% (+0.0%)	33.3% (+10.0%)	**40.0% (+16.7%)**
	GPQA-Diamond	47.9%	51.0% (+3.1%)	46.4% (-1.5%)	**52.0% (+4.1%)**
	HumanEval+	75.4%	76.3% (+0.9%)	74.5% (-0.9%)	**77.2% (+1.8%)**
	MBPP+	69.8%	65.3% (-4.5%)	67.4% (-2.4%)	**71.1% (+1.3%)**
	OpenBookQA	81.0%	80.4% (-0.6%)	**82.2% (+1.2%)**	**82.2% (+1.2%)**
Qwen3-14B	AIME24	63.3%	63.3% (+0.0%)	66.7% (+3.4%)	**76.7% (+13.4%)**
	AIME25	56.7%	60.0% (+3.3%)	63.3% (+6.6%)	**66.7% (+10.0%)**
	GPQA-Diamond	48.5%	51.5% (+3.0%)	52.0% (+3.5%)	**53.0% (+4.5%)**
	HumanEval+	77.2%	81.5% (+4.3%)	86.8% (+9.6%)	**87.7% (+10.5%)**
	MBPP+	68.5%	72.8% (+4.3%)	75.7% (+7.2%)	**77.2% (+8.7%)**
	OpenBookQA	86.6%	86.2% (-0.4%)	88.8% (+2.2%)	**89.0% (+2.4%)**
DeepSeek-R1-Distill Qwen-32B	AIME24	66.7%	50.0% (-16.7%)	50.0% (-16.7%)	**70.0% (+3.3%)**
	AIME25	**50.0%**	**50.0% (+0.0%)**	43.3% (-6.7%)	**50.0% (+0.0%)**
	GPQA-Diamond	53.3%	56.5% (+3.2%)	55.1% (+1.8%)	**57.5% (+4.2%)**
	HumanEval+	80.7%	82.4% (+1.7%)	83.3% (+2.6%)	**88.5% (+7.8%)**
	MBPP+	73.8%	74.6% (+0.8%)	75.7% (+1.9%)	**78.5% (+4.7%)**
	OpenBookQA	**91.2%**	**91.2% (+0.0%)**	89.6% (-1.6%)	**91.2% (+0.0%)**

family (Yang et al., 2025) (4B, 8B, and 14B parameters), Ministral-3-8B (Liu et al., 2026), and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025), allowing us to assess performance across both mid-scale and large-scale reasoning models.

For baseline comparison, we evaluate DiffMAS against four representative paradigms: (i) *Single-model inference* (Single), where a single language model directly performs standard autoregressive generation; (ii) *Text-based multi-agent systems* (TextMAS), where multiple agents collaborate through explicit natural language communication; (iii) *Training-free latent multi-agent systems* (LatentMAS), where agents communicate through continuous latent representations without additional training; and (iv) *Cache-to-Cache communication* (C2C), where agents exchange KV-cache representations through learned fusion modules. We use the released C2C checkpoints, which were trained on about 500K examples from OpenHermes-2.5 (Teknium, 2023), mainly instruction-following data. Retraining C2C's fusion module on our small training sets did not produce a working model, so C2C results reflect out-of-domain use of the released checkpoints rather than a tuned competitor (Appendix I). In addition, to separate learned communication from supervised fine-tuning itself, Section 6 compares DiffMAS with *Single+SFT* and *TextMAS+SFT*, trained on exactly the same data with the same LoRA configuration and optimization schedule. Implementation details are in Appendix F and the inference setup in Appendix E.



(a) Perplexity, Qwen3-4B.



(b) Self-consistency, Qwen3-4B (top) and Qwen3-8B (bottom).

Figure 2: Decoding stability on AIME2024. (a) Distribution of per-problem perplexity for DiffMAS and LatentMAS; density indicates the number of problems in each perplexity range. (b) Number of correct answers out of four samples per problem.

Training Setup. We perform SFT of DiffMAS on high-performance GPU hardware using parameter-efficient LoRA adaptation. Experiments with Qwen3-4B, Qwen3-8B, and Ministral-3-8B are conducted on NVIDIA A40 GPUs, while Qwen3-14B and DeepSeek-R1-Distill-Qwen-32B are trained on NVIDIA H200 GPUs to accommodate their higher memory and computational requirements. Task-specific LoRA adapters are trained on small subsets of benchmark datasets to specialize the model communication for different reasoning domains. For mathematical and scientific reasoning, we fine-tune on 210 samples from Hendrycks Math (Hendrycks et al., 2021) for one epoch. For code generation, we train on 50 problems from HumanEval (Chen et al., 2021) for 10 epochs. These 50 problems are removed from HumanEval+ evaluation for all methods, leaving 114 problems. For commonsense reasoning, we use 700 samples from CommonsenseQA (Talmor et al., 2019) for one epoch. No OpenBookQA data is used in training, and OpenBookQA is evaluated on 1,000 questions sampled from its train split (Appendix E). We intentionally restrict training to small datasets to evaluate whether DiffMAS can learn effective latent communication with minimal supervision. Details of training setup are in Appendix D.

4.2 Main Results

Overall Performance. As shown in Tables 1 and 2, DiffMAS is best or tied-best in all 30 model–benchmark settings, spanning math and science reasoning, code generation, and commonsense reasoning, and it never falls below the training-free LatentMAS pipeline it builds on. Gains are largest on the hardest benchmarks: +26.7 points on AIME24 and +20.2 on GPQA-Diamond for Qwen3-8B, and +20.0 on AIME24 for Qwen3-4B. They persist at larger scales, e.g., +13.4 on AIME24 for Qwen3-14B and +7.8 on HumanEval+ for DeepSeek-R1-Distill-Qwen-32B. Gains are small on OpenBookQA, where all methods are near their ceiling.

The released C2C checkpoints perform poorly on AIME and code, including 0.0% on AIME. We attribute this to the mismatch between their instruction-following training data and long multi-step reasoning, and we do not draw conclusions from the C2C comparison.

5 Analysis

Decoding Stability Analysis via Perplexity. We measure decoding stability by the token-level perplexity (PPL) of the final agent's generated output, i.e., the exponentiated mean negative log-likelihood of its tokens. Because latent communication changes the attention context, PPL is a sensitive measure of how coherently the injected KV states are interpreted; lower PPL indicates more stable decoding.

We compare DiffMAS and LatentMAS on AIME 2024 using Qwen3-4B (Figure 2a). DiffMAS achieves lower mean perplexity (1.24 vs. 1.31) and a tighter distribution, with most values between 1.15–1.30, while LatentMAS exhibits a heavier tail with outliers exceeding 2.0, indicating unstable

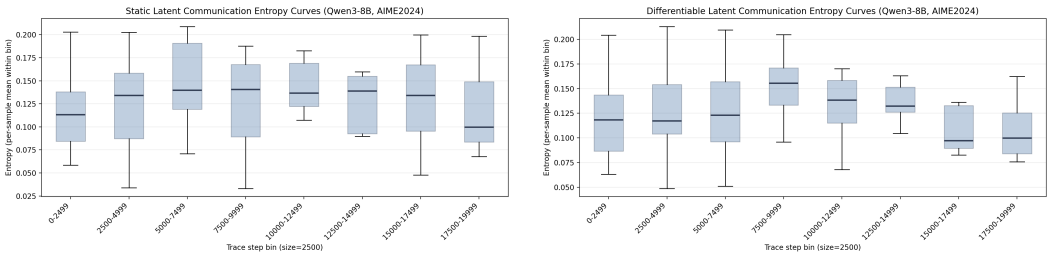


Figure 3: Token-level predictive entropy (top-25) of the final agent on AIME2024 under LatentMAS (left) and DiffMAS (right), averaged over 30 runs.

decoding caused by misaligned latent signals. This pattern also appears at the instance level: while both methods behave similarly on easier problems, LatentMAS shows frequent spikes on harder instances, whereas DiffMAS maintains consistently low perplexity, suggesting that supervised training leads to more structured encoding and interpretation of latent KV representations.

Self-Consistency in Inference. To measure stability under stochastic decoding, we sample each AIME 2024 problem four times and record the number of correct answers (0–4).

Figure 2b shows that DiffMAS consistently achieves higher self-consistency than LatentMAS and TextMAS across both Qwen3-4B and Qwen3-8B. DiffMAS displays a clear shift toward high-consistency outcomes (3–4 correct samples) across most problems, indicating more stable reasoning trajectories. In contrast, LatentMAS frequently produces low-consistency results (0–1 correct samples), suggesting brittle reasoning where small latent communication perturbations lead to divergent outcomes. TextMAS exhibits highly polarized behavior, with many consistent failures (0/4) and only occasional consistent successes (4/4). DiffMAS thus gains through more stable coordination rather than sporadic correct generations. Qualitative examples of both behaviors are given in Appendix J.

Token-Level Entropy Dynamics. To examine stability during decoding, we compute the entropy of the final agent's top-25 token distribution at each step, averaged over 30 runs (Figure 3). Entropy rises as decoding proceeds under both methods, but training-free latent injection shows higher variance and frequent spikes, whereas DiffMAS grows smoothly with lower peaks. This matches the mechanism of Section 3.3: in DiffMAS the upstream KV states stay in the computation graph and are shaped by the downstream loss, whereas under static injection the final agent must compensate for uncalibrated latent states on its own.

6 Ablation Study

Learning to Solve vs. Learning to Communicate. To separate learning communication from supervised adaptation, we fine-tune the single-agent and TextMAS pipelines on exactly the same data, with the same shared LoRA configuration and optimization schedule as DiffMAS (Single+SFT and TextMAS+SFT). As Table 3 shows, DiffMAS has the highest average accuracy on every backbone. Averaged over all five backbones, DiffMAS reaches 70.4%, compared with 63.8% for Single+SFT and 64.3% for TextMAS+SFT, while supervision alone lifts the untrained Single and TextMAS pipelines by only 1.4 points each. Per benchmark, it outperforms both SFT baselines in 26 of the 30 settings and is never below either of them (Table 7). The two ways of applying the same supervision also behave differently. Fine-tuning through the text channel is unreliable: TextMAS+SFT falls below untrained TextMAS in 13 of 30 settings. Fine-tuning through the latent channel never falls below its untrained counterpart, LatentMAS.

This contrast is what Proposition 3.1 and Remark A.3 predict. In TextMAS+SFT the upstream messages are sampled tokens, so the loss can only adapt how the final agent reads a fixed message distribution. In DiffMAS the same loss also reaches how the upstream agents write the trace. On Qwen3-8B, for example, the gains over TextMAS+SFT are largest on AIME25 (56.7% vs. 50.0%) and GPQA-Diamond (60.1% vs. 53.5%).

Table 3: Learning to solve vs. learning to communicate. Average accuracy (%) over the six benchmarks. +SFT baselines use the same training data, LoRA configuration, and schedule as DiffMAS. Per-benchmark results are in Table 7.

Model	Single	Single+SFT	TextMAS	TextMAS+SFT	LatentMAS	DiffMAS
Qwen3-4B	57.8	61.4	60.4	64.1	60.6	**65.8**
Qwen3-8B	59.9	64.5	63.5	68.6	66.1	**72.6**
Ministral-3-8B	57.9	55.0	53.8	51.1	56.2	**66.0**
Qwen3-14B	66.8	70.2	69.2	70.0	72.2	**75.0**
DeepSeek-R1-Distill-Qwen-32B	69.3	67.7	67.5	67.9	66.2	**72.6**
Mean	62.4	63.8	62.9	64.3	64.2	**70.4**

Table 4: DiffMAS vs. StitchMAS on Qwen3-8B (%); both use SFT.

Benchmark	DiffMAS	StitchMAS
GPQA-D	**60.1**	48.4
AIME24	**76.7**	60.0
AIME25	**56.7**	53.3

Table 5: Number of communication steps per agent for DiffMAS on AIME 2024 using Qwen3-8B.

# Steps	0	10	40	100	150	200
Acc. (%)	50.0	**76.7**	63.3	73.3	66.7	63.3

Stitched vs. Continuous Latent Communication under SFT. To isolate the effect of how the latent channel is constructed, we compare DiffMAS with **StitchMAS**, an SFT-based latent baseline in which each agent independently generates its KV states, which are then concatenated for downstream use. Both methods use SFT and differ only in how KV states are produced and integrated. As shown in Table 4, StitchMAS underperforms DiffMAS on Qwen3-8B, with large gaps on GPQA-Diamond (60.1% vs. 48.4%) and AIME2024 (76.7% vs. 60.0%). StitchMAS segments are generated independently and lack a shared positional frame. In DiffMAS, each agent conditions on the continuous trace written by the agents before it, with positions assigned contiguously across agents, which gives the cross-stage paths of Proposition 3.1(ii) through which the loss shapes how upstream agents encode.

Effect of Communication Steps. We vary the number of latent communication steps to test whether gains come from long latent trajectories or from a compact learned protocol (Table 5). A small number of steps yields a large improvement, from 50.0% at 0 steps to 76.7% at 10 steps. Longer traces bring no further gain, consistent with the attention-dilution trade-off of Section 3.3: as the trace grows, each trace token receives a smaller share of the decoder's attention and of the training signal. DiffMAS thus benefits from a compact protocol rather than from communication volume. The 10-step setting selected in this sweep is used unchanged for all models and benchmarks.

Topology and Cost. DiffMAS is not tied to the sequential chain. In a DAG-style workflow, two solver agents work in parallel on a parser's trace and a verifier reads both. With Qwen3-8B, DiffMAS reaches 76.7% on AIME24 and 58.0% on GPQA-Diamond under this workflow, close to the sequential version (76.7%, 60.1%) and well above LatentMAS (56.7%, 45.5%); details are in Appendix H. The gains also come at no extra inference cost. DiffMAS runs the same pipeline as LatentMAS with a mergeable LoRA adapter, and its wall-clock latency is essentially the same (410 seconds vs. 407 seconds per AIME24 query; Appendix F).

7 Conclusion

In this paper, we introduce DiffMAS, a supervised training framework that treats latent communication as a learnable operator within multi-agent interactions. Because the KV trace is append-only and the adapter is shared, a single loss on the final agent trains both how upstream agents encode and how the final agent interprets, which text-based fine-tuning cannot do. Across five backbones and six reasoning, coding, and commonsense benchmarks, DiffMAS matches or exceeds single-agent, text-based, and latent baselines in every setting, including baselines fine-tuned on identical data, and it improves decoding stability over training-free latent communication. More broadly, our results highlight latent communication as an important optimization target for multi-agent systems and motivate future work on fully differentiable MAS with end-to-end optimized interactions.

8 AI USE STATEMENT

We used generative AI tools to aid and polish the writing of this paper, including rewording, tightening the presentation, and improving the flow of the narrative. We also used them to help refine the formal statement and proof of Proposition 3.1 and the accompanying remarks in Appendix A; the authors checked every step of the proof. We also used Gemini-3-Flash-Preview to generate the reasoning traces used as commonsense training data; only traces with a correct final answer were kept (Appendix D). Apart from these uses, we did not use generative AI tools to generate research ideas, run or analyze experiments, or produce any reported numbers. All experimental results come from our own runs. We have reviewed all AI-assisted content and take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation, 2024. URL https://arxiv.org/abs/2309.17288.

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. Evaluating large language models trained on code, 2021. URL https://arxiv.org/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. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors, 2023. URL https://arxiv.org/abs/2308.10848.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system, 2025. URL https://arxiv.org/abs/2410.08115.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space, 2026. URL https://arxiv.org/abs/2511.09149.

Jacob Fein-Ashley, Dhruv Parikh, Rajgopal Kannan, and Viktor Prasanna. Mixture of thoughts: Learning to aggregate what experts think, not just what they say, 2025. URL https://arxiv.org/abs/2509.21164.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning, 2016. URL https://arxiv.org/abs/1605.06676.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models, 2026. URL https://arxiv.org/abs/2510.03215.

Google. Gemini 3 flash. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-flash, 2025. Vertex AI documentation. Accessed: 2026-03-29.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols, 2017. URL https://arxiv.org/abs/1705.11192.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet,

Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

Hao Ma, Tianyi Hu, Zhiqiang Pu, Boyin Liu, Xiaolin Ai, Yanyan Liang, and Min Chen. Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning, 2025. URL https://arxiv.org/abs/2410.06101.

math-ai. Aime 2025 dataset, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Maxwell-Jia. Aime 2024 dataset, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations, 2018. URL https://arxiv.org/abs/1703.04908.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. Malt: Improving reasoning with multi-agent llm training, 2025. URL https://arxiv.org/abs/2412.01928.

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. Chatdev: Communicative agents for software development, 2024. URL https://arxiv.org/abs/2307.07924.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2505.23352.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation, 2016. URL https://arxiv.org/abs/1605.07736.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2019. URL https://arxiv.org/abs/1811.00937.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Kun Wang, Guibin Zhang, ManKit Ye, Xinyu Deng, Dongxia Wang, Xiaobin Hu, Jinyang Guo, Yang Liu, and Yufei Guo. Mas^2: Self-generative, self-configuring, self-rectifying multi-agent systems, 2025. URL https://arxiv.org/abs/2509.24323.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-star: Language models can teach themselves to think before speaking, 2024. URL https://arxiv.org/abs/2403.09629.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet, 2025a. URL https://arxiv.org/abs/2502.04180.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025b. URL https://arxiv.org/abs/2410.10762.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration, 2025. URL https://arxiv.org/abs/2510.20733.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning, 2025. URL https://arxiv.org/abs/2507.06203.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs, 2024. URL https://arxiv.org/abs/2402.16823.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2025. URL https://arxiv.org/abs/2511.20639.

A Structural Properties of Communication Interfaces

Scope. The results in this appendix characterize the communication *interface*: how a change in, or a gradient on, an agent's emission propagates to the rest of the system. They make no claim about the semantic quality of the learned trace, about optimization speed, or about expressivity. The contraction assumption is used only for the overwriting comparator; nothing in the concatenation results depends on it.

A.1 Setup and notation

Concatenation (DiffMAS). Stage $j \in \{1, \dots, K\}$ emits a segment $\mathbf{Z}^{(j)} = G_\theta^{(j)}(\mathbf{Z}_{1:N_{j-1}}) \in \mathcal{Z}^T$, where $G_\theta^{(j)}$ absorbs the input x, the prompt p_j, and the T within-stage micro-steps of Appendix B. The trace is updated by (2), so $\mathbf{Z}_{1:N_K} = [\mathbf{Z}^{(1)}; \dots; \mathbf{Z}^{(K)}]$, and the loss is $\mathcal{L} = \ell\big(\mathrm{Dec}_\theta(\mathbf{Z}_{1:N_K})\big)$. Let P_k denote the block-insertion operator that places a segment into block k of the trace, so that $\mathbf{Z}_{1:N_K} = \sum_k P_k \mathbf{Z}^{(k)}$, $P_k^\top P_k = I$, and $P_k^\top P_{k'} = 0$ for $k \neq k'$.

We treat $\mathbf{Z}^{(j)}$ as a free input and propagate its effect forward with $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(j-1)}$ held fixed. For $k \geq j$, define the total cross-stage Jacobian $D_{k,j} \triangleq d\mathbf{Z}^{(k)}/d\mathbf{Z}^{(j)}$. By causality $D_{j,j} = I$, and by the chain rule

$$D_{k,j} = \sum_{i=j}^{k-1} \frac{\partial G_\theta^{(k)}}{\partial \mathbf{Z}^{(i)}} D_{i,j}, \qquad k > j, \tag{5}$$

where $\partial G_\theta^{(k)}/\partial \mathbf{Z}^{(i)}$ is the partial Jacobian of stage k with respect to block i of its input. Segments $k < j$ do not depend on $\mathbf{Z}^{(j)}$. Hence the total Jacobian of the final trace is

$$J_j \triangleq \frac{d\mathbf{Z}_{1:N_K}}{d\mathbf{Z}^{(j)}} = P_j + \sum_{k>j} P_k D_{k,j}. \tag{6}$$

Write $g \triangleq \partial\mathcal{L}/\partial\mathbf{Z}_{1:N_K}$ for the gradient the decoder places on the final trace, $g_k \triangleq P_k^\top g$ for its k-th block, and $\lambda_j \triangleq d\mathcal{L}/d\mathbf{Z}^{(j)}$ for the total gradient reaching segment j.

Overwriting (comparator). A carrier $h_j \in \mathbb{R}^{d_h}$ is updated by $h_j = F_\theta^{(j)}(h_{j-1})$, with $\mathcal{L} = \ell\big(\mathrm{Dec}_\theta(h_K)\big)$, and the transition is contractive:

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\|_2 \leq \rho < 1 \qquad \text{for all } j. \tag{7}$$

All norms are Euclidean on vectorized variables, and Jacobian norms are the induced $\ell_2 \to \ell_2$ operator norms.

A.2 Proof of Proposition 3.1

Proof. **(i) Retention.** *Overwriting.* By the chain rule $dh_K/dh_j = \prod_{i=j+1}^{K} \partial h_i/\partial h_{i-1}$, and by submultiplicativity and (7), $\|dh_K/dh_j\|_2 \le \prod_{i=j+1}^{K} \|\partial h_i/\partial h_{i-1}\|_2 \le \rho^{K-j}$.

Concatenation. Fix a direction v. By (6), $J_j v = P_j v + \sum_{k>j} P_k D_{k,j} v$. The terms occupy disjoint blocks of the trace, so they are mutually orthogonal and

$$\|J_j v\|_2^2 \;=\; \|v\|_2^2 + \sum_{k>j} \|D_{k,j} v\|_2^2 \;\ge\; \|v\|_2^2. \tag{8}$$

Hence $\sigma_{\min}(J_j) \ge 1$. The inequality holds for every θ and does not require any assumption on the stage maps $G_\theta^{(k)}$: segment j is itself a block of the output, and later stages can only add orthogonal blocks, never remove it.

(ii) Gradient paths. *Overwriting.* $d\mathcal{L}/dh_j = (dh_K/dh_j)^\top d\mathcal{L}/dh_K$, so by (i), $\|d\mathcal{L}/dh_j\|_2 \le \rho^{K-j}\|d\mathcal{L}/dh_K\|_2$.

Concatenation. $\lambda_j = J_j^\top g$. Substituting (6) and using $P_k^\top g = g_k$,

$$\lambda_j = P_j^\top g + \sum_{k>j} D_{k,j}^\top P_k^\top g = g_j + \sum_{k>j} D_{k,j}^\top g_k,$$

which is (3). The first term is the gradient the decoder places on block j and involves no stage Jacobian; every product of stage Jacobians in (5) enters only through the second term.

(iii) Shared parameters. Consider the computation graph whose nodes are the segments $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(K)}$ and the decoder output. By reverse-mode differentiation, the gradient with respect to a parameter shared across nodes is the sum over nodes of the node's local parameter Jacobian, transposed, applied to that node's total adjoint:

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}\big|_{\mathrm{Dec}} + \sum_{j=1}^{K} \big(\partial_\theta G_\theta^{(j)}\big)^\top \lambda_j,$$

where $\partial_\theta G_\theta^{(j)}$ is the Jacobian of stage j with respect to θ with its input trace held fixed. This is (4), and substituting (3) shows that the stage-j term contains $(\partial_\theta G_\theta^{(j)})^\top g_j$, which does not depend on $K - j$. For overwriting, the same argument gives stage-j terms $(\partial_\theta F_\theta^{(j)})^\top d\mathcal{L}/dh_j$, whose norm is at most $\|\partial_\theta F_\theta^{(j)}\|_2 \, \rho^{K-j} \, \|d\mathcal{L}/dh_K\|_2$ by (ii).

A.3 Remarks

Remark A.1 (When the direct path dominates). Part (ii) is a decomposition, not a lower bound on $\|\lambda_j\|_2$: the indirect paths could in principle cancel the direct term. By the triangle inequality,

$$\|\lambda_j - g_j\|_2 \;\le\; \sum_{k>j} \|D_{k,j}\|_2 \, \|g_k\|_2,$$

so $\lambda_j \approx g_j$ whenever later stages depend only weakly on segment j. In the opposite case the indirect paths themselves carry signal, since later stages read segment j directly through attention rather than through a fixed-dimensional carrier. By contrast, the retention statement (i) is an unconditional lower bound.

Remark A.2 (Attention dilution). The direct gradient g_j is computed by the decoder's attention. Consider a single attention head with one query q over trace tokens t with keys k_t and values v_t (possibly alongside other context tokens), $o = \sum_t \alpha_t v_t$ with $\alpha = \mathrm{softmax}(q^\top k_t/\sqrt{d_k})$. Then

$$\frac{\partial o}{\partial v_t} = \alpha_t I, \qquad \frac{\partial o}{\partial k_t} = \frac{\alpha_t}{\sqrt{d_k}} \, (v_t - o) \, q^\top,$$

so both the value path and the key path to token t are proportional to its attention weight α_t. Let $A_j \triangleq \sum_{t \in \text{segment } j} \alpha_t$ be the mass on segment j. Since $\sum_j A_j \le 1$, the average mass per segment is

at most $1/K$ and the average weight per trace token is at most $1/(KT)$. Heads, queries, and layers contribute additively, and each term has the same form. Concatenation thus replaces the geometric factor ρ^{K-j} with a dilution factor that is at most inverse-linear in trace length on average. Unlike ρ^{K-j}, this factor is set by learned attention and can be reallocated to informative segments. It also predicts that very long traces weaken the per-token training signal, which is consistent with every longer setting in Table 5 scoring below the 10-step setting. We present this as a consistent explanation, not as a proof of that trend.

Remark A.3 (Text and detached communication). If upstream agents communicate through sampled tokens, or if the upstream KV trace is detached from the computation graph, then $\lambda_j = 0$ for all $j < K$ and (4) reduces to the decoder term plus the final stage. Only interpretation is trained. The upstream sum in (4) is therefore exactly what separates DiffMAS from these variants, including TextMAS+SFT.

B Within-Stage Micro-Dynamics

Each stage j executes T micro-steps of differentiable computation. Let $s_t^{(j)} \in \mathbb{R}^m$ denote the internal state at micro-step t within stage j.

Initialization. The internal state is initialized as $s_0^{(j)} = \eta_\theta(x, p_j, \mathbf{Z}_{1:N_{j-1}})$.

Emission and update order. For each micro-step $t \in \{1, \dots, T\}$, the stage first emits a latent block

$$z_t^{(j)} = g_\theta\Big(s_{t-1}^{(j)}, x, p_j, \mathbf{Z}_{1:N_{j-1}}, \mathbf{Z}_{1:(t-1)}^{(j)}\Big), \tag{9}$$

which is appended to the trace. The state then evolves via a residual update:

$$s_t^{(j)} = s_{t-1}^{(j)} + f_\theta\Big(s_{t-1}^{(j)}, x, p_j, \mathbf{Z}_{1:N_{j-1}}, \mathbf{Z}_{1:t}^{(j)}\Big). \tag{10}$$

All maps η_θ, g_θ, and f_θ are differentiable in θ.

Relation to the stage operator. The operator $A_\theta^{(j)}$ is the macro-map induced by the initialization, emission, and update rules above.

C MAS Setup

To minimize confounding effects from specialized orchestration designs, we adopt a standard and widely used sequential multi-agent pipeline (Zou et al., 2025). Specifically, we use a sequential four-agent structure consisting of a **Planner**, **Critic**, **Refiner**, and **Solver**. This canonical setup is intentionally simple and commonly used in prior LLM-based multi-agent reasoning systems, allowing us to focus on the effect of the *communication medium* (text vs. latent KV states) rather than any particular MAS topology or role-engineering heuristics. Across all methods, we keep the agent roles, ordering, and role prompts fixed; the only difference is how intermediate information is transmitted between agents.

Planner. The Planner receives the input question and produces a concise step-by-step plan without generating the final answer.

Planner

You are a Planner Agent. Given an input question, design a clear, step-by-step plan for how to solve the question.
Question: {question}
Your outlined plan should be concise with a few bulletpoints for each step. Do not produce the final answer. Now output your plan to solve the question below:

Critic. The Critic evaluates the correctness and completeness of the Planner's plan and provides constructive feedback. The plan is provided via the inter-agent communication channel (text for TextMAS; latent KV representation for LatentMAS/DiffMAS).

Critic

Question: {question}
You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan. The plan information is provided in latent KV representation format. Review the plan and question and output: (1) original plan contents (2) constructive feedback on the original plan.
Format your response as follows: Original Plan: [Copy the provided Planner Agent's plan here] Feedback: [Your detailed feedback to improve the plan here]
Now, output your response below:

Refiner. The Refiner integrates the original plan and Critic feedback to produce a refined plan that is both correct and concise. It receives (i) latent-format information containing the previous plan with feedback, and (ii) the input question in text format.

Refiner

Question: {question}
You are a Refiner Agent to provide a refined step-by-step plan for solving the given question.
You are provided with: (1) latent-format information: a previous plan with feedback (2) text-format information: the input question you need to solve.
Based on the input, write a refined and improved plan to solve the question. Make sure your output plan is correct and concise.
Now, output your refined plan below:

Solver. The final agent produces the answer to the target question. It conditions on all accumulated information from upstream agents. Since upstream latent information can occasionally contain irrelevant content, we explicitly instruct the agent to ignore unhelpful context and output only the final answer.

Solver

Target Question: {question}
You are a helpful assistant. You are provided with latent information for reference and a target question to solve.
The latent information might contain irrelevant contents. Ignore it if it is not helpful for solving the target question.
You must reason step-by-step to solve the provided Target Question without outputting other irrelevant information.
Now, reason step by step and output the final answer inside [YOUR FINAL ANSWER].

D Training Setup

DiffMAS is trained directly within the multi-agent interaction loop, where inter-agent latent communication is treated as part of the model's conditioning context. During training, the backbone language model remains frozen, and only lightweight LoRA adapters are optimized, enabling parameter-efficient adaptation without modifying the pretrained base model.

For each training instance, intermediate agents first perform latent reasoning steps and produce continuous KV representations that encode their internal computation. These latent KV states are accumulated sequentially across agents and passed to downstream agents as a shared latent context, forming the full multi-agent reasoning trajectory. The final output agent conditions on this accumulated latent context together with its task-specific prompt and generates the target output autoregressively. Training is performed with teacher forcing on the target sequence, and the loss is computed as the negative log-likelihood of the target tokens conditioned on the full latent context and final prompt. Prompt tokens and padding positions are masked out from the loss, ensuring that gradients are driven only by the correctness of the final output. In this way, gradients propagate

through the latent communication pathway, allowing the model to jointly adapt how upstream agents encode information into KV states and how downstream agents interpret them.

We train task-specific LoRA adapters on small curated subsets of benchmark datasets in order to specialize latent communication for different reasoning domains. For mathematical reasoning, we construct a 210-sample training set from Hendrycks Math by uniformly sampling 30 samples from each of seven subsets: algebra, counting and probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus. This balanced composition ensures coverage across multiple types of symbolic and quantitative reasoning while keeping the training set intentionally small. For code generation, we randomly choose 50 samples from HumanEval. For commonsense reasoning, we use 700 samples from CommonsenseQA and collected synthetic reasoning traces generated by Gemini-3-Flash-Preview (Google, 2025). The box below contains the prompt we used for generating reasoning traces. We filter the generated reasoning traces by retaining only those for which Gemini produces the correct final answer, resulting in a final set of 700 high-quality traces that are used as supervision.

Gemini Prompt

Target Question: {question}
You are a helpful assistant. You must reason step-by-step to solve the provided Target Question without outputting other irrelevant information.
Your final answer must be selected from A,B,C,D,E. For example \boxed{A}. Do not add any other contents inside the box.
Now, reason step by step and output the final answer inside \boxed{YOUR_FINAL_ANSWER}.

Optimization is carried out using AdamW with a cosine learning-rate schedule and a linear warmup ratio of 0.03. LoRA parameters are trained with a fixed learning rate of 5×10^{-5}, using rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate 0.05, while the backbone model remains frozen. Gradients are accumulated over 64 micro-batches per optimization step, and gradient norms are clipped to value of 1.0.

E Inference Setup

Across all baselines and our method, we use the same decoding configuration. All LLM agents are run with a temperature of 0.6 and a top-p value of 0.95. The maximum output length for the final output agent is adjusted according to its task difficulty: 2,048 tokens for OpenBookQA; 4,096 tokens for MBPP+ and HumanEval+; 8,192 tokens for GPQA-Diamond; and 32,768 tokens for AIME 2024/2025. No additional sampling heuristics are introduced.

For evaluation coverage, we run experiments on the full benchmark sets of AIME 2024, AIME 2025, GPQA-Diamond, MBPP+, and HumanEval+, except that the 50 HumanEval problems used for training are removed from HumanEval+ for all methods (114 problems remain). For OpenBookQA, we evaluate on 1,000 questions sampled uniformly at random from the train split of the main subset. No OpenBookQA data is used for training (commonsense adapters are trained on CommonsenseQA), so these questions are unseen by every method. Because they are not the official test split, our OpenBookQA numbers are comparable across the methods in this paper but not to published test-set results. None of our conclusions rests on OpenBookQA alone.

F Implementation Details and Cost

Parameterization. A single LoRA adapter (rank $r = 8$, $\alpha = 16$, dropout 0.05) is inserted into the query, key, value, and output projections of every transformer block. It is shared by all agents, and the backbone is frozen. The +SFT baselines in Section 6 use the same adapter configuration, data, and schedule.

Gradient flow. Upstream agents generate their KV segments with gradient tracking enabled, and no stop-gradient is applied anywhere in the trace. The only loss is the token-level negative log-likelihood of the final agent's target output. It is backpropagated through the final agent's attention over the

trace, through every upstream agent's KV segment, and into the shared adapter. This is the setting analyzed in Proposition 3.1.

Positions. KV segments are appended with contiguous position indices across agents; positions are not reset per agent. The accumulated trace is therefore a single contiguous sequence under RoPE.

Hyperparameters. All models and benchmarks use the same configuration: 10 latent communication steps per upstream agent, together with the optimization settings of Appendix D. There is no per-model or per-benchmark tuning.

Answer extraction. For AIME, GPQA-Diamond, and OpenBookQA we use the deterministic rule-based extraction of LatentMAS (Zou et al., 2025). HumanEval+ and MBPP+ are scored with the official execution-based evaluation harness (Liu et al., 2023).

Cost. Training adds the cost of backpropagating through the multi-agent trace (hardware is listed in Section 4.1). At inference, DiffMAS runs exactly the LatentMAS pipeline, and the LoRA adapter can be merged into the backbone, so inference cost is unchanged. Table 6 confirms that wall-clock latency matches LatentMAS.

Table 6: Accuracy and wall-clock inference latency on Qwen3-8B.

Benchmark	Method	Acc. (%)	Runtime (s/query)
AIME2024	LatentMAS	56.7	407
	DiffMAS	**76.7**	410
AIME2025	LatentMAS	53.3	754
	DiffMAS	**56.7**	748

G Full Results for Matched-Training Baselines

Relative to the stronger of Single+SFT and TextMAS+SFT in each setting, DiffMAS is better in 26 settings and tied in 4 (Qwen3-4B AIME24 and HumanEval+, Qwen3-8B AIME24, DeepSeek-R1-Distill-Qwen-32B AIME25). The mean gain is 4.7 points and the median 1.8. A sign test over settings (26 wins, 0 losses) gives $p < 10^{-7}$, treating settings as independent, which is an approximation because they share backbones and benchmarks.

H Beyond Sequential Topologies

DiffMAS does not require a chain of agents: it operates on whatever KV trace the agents produce. To check that the gains are not specific to the Planner–Critic–Refiner–Solver chain, we evaluate a DAG-style workflow for mathematical and scientific reasoning, Parser $\rightarrow$ {Solver A $\|$ Solver B} $\rightarrow$ Verifier. The two solvers run in parallel on the parser's trace, and the verifier reads both before answering. Table 8 shows that DiffMAS with the DAG topology performs close to the sequential version and remains well above training-free LatentMAS. Optimizing the topology itself is left to future work.

I Notes on the C2C Baseline

C2C (Fu et al., 2026) fuses KV caches across models with a gated residual MLP module that was trained on about 500K communication examples from OpenHermes-2.5. DiffMAS instead adapts a LoRA on a few hundred multi-agent trajectories. We attempted to retrain C2C's fusion module with our training sets and recipe, but the resulting models collapsed to 0.0% on all six benchmarks for Qwen3-4B, Qwen3-8B, and Ministral-3-8B. A matched retraining would require building a large task-specific communication corpus, which is beyond the scope of this work. We therefore report the released checkpoints, applied out of domain. Their low scores, including 0.0% on AIME, should not be read as a tuned comparison, and none of our conclusions depend on C2C.

Table 7: Per-benchmark accuracy (%) of the matched-training comparison in Table 3. All three methods use the same training data, LoRA configuration, and schedule. Best results are in bold; ties are bolded jointly.

Model	Task	Single+SFT	TextMAS+SFT	DiffMAS
Qwen3-4B	AIME24	60.0	**63.3**	**63.3**
	AIME25	46.7	46.7	**50.0**
	GPQA-Diamond	36.9	43.9	**46.4**
	HumanEval+	75.4	**79.8**	**79.8**
	MBPP+	68.3	70.4	**71.9**
	OpenBookQA	81.1	80.6	**83.2**
Qwen3-8B	AIME24	66.7	**76.7**	**76.7**
	AIME25	50.0	50.0	**56.7**
	GPQA-Diamond	47.0	53.5	**60.1**
	HumanEval+	74.5	76.3	**81.5**
	MBPP+	64.8	73.8	**74.8**
	OpenBookQA	84.1	81.2	**85.8**
Ministral-3-8B	AIME24	30.0	16.7	**73.3**
	AIME25	23.3	20.0	**40.0**
	GPQA-Diamond	49.5	47.0	**52.0**
	HumanEval+	75.4	74.6	**77.2**
	MBPP+	70.1	66.7	**71.1**
	OpenBookQA	81.7	81.7	**82.2**
Qwen3-14B	AIME24	66.7	60.0	**76.7**
	AIME25	63.3	56.7	**66.7**
	GPQA-Diamond	52.5	52.0	**53.0**
	HumanEval+	78.9	86.8	**87.7**
	MBPP+	72.3	76.4	**77.2**
	OpenBookQA	87.8	88.4	**89.0**
DeepSeek-R1-Distill Qwen-32B	AIME24	53.3	53.3	**70.0**
	AIME25	**50.0**	**50.0**	**50.0**
	GPQA-Diamond	55.1	54.0	**57.5**
	HumanEval+	82.5	82.5	**88.5**
	MBPP+	74.6	76.7	**78.5**
	OpenBookQA	90.7	90.8	**91.2**

Table 8: DiffMAS under a DAG-style topology on Qwen3-8B (accuracy, %).

Method	AIME24	GPQA-Diamond
LatentMAS (sequential)	56.7	45.5
DiffMAS (DAG)	**76.7**	58.0
DiffMAS (sequential)	**76.7**	**60.1**

J Case Study

Case Study (LatentMAS on AIME2024)

Input Question: Let $\mathcal{B}$ be the set of rectangular boxes with surface area 54 and volume 23. Let r be the radius of the smallest sphere that can contain each of the rectangular boxes that are elements of $\mathcal{B}$. The value of r^2 can be written as $\frac{p}{q}$, where p and q are relatively prime positive integers. Find $p+q$.

LatentMAS Output:
Okay, let's try to solve this problem step by step. So, we have a tetrahedron ABCD with specific edge lengths. The question is about finding the equal distance from an interior point I to each of

the tetrahedron's faces. That distance is given in a particular form, and we need to find m + n, and p, then sum them.
First, I remember that in a tetrahedron, if there's a point equidistant from all four faces, that point is called the incenter. But wait, in a tetrahedron, the incenter exists only if the tetrahedron is isohedral or has some symmetry. But here, the given edge lengths seem symmetric. Let me check the edge lengths again: $AB = CD$, $CD = \sqrt{41}$, $AC = BD = \sqrt{80}$, $BC = AD$. So, the edges are symmetric in pairs. For example, $AB = CD$, $AC = BD$, $BC = AD$. This suggests that the tetrahedron is isosceles or maybe even a special type of tetrahedron, like a disphenoid...

[Attempt Solving Steps Omitted]

Alternatively, let me think about the volume. Let me try to compute the volume using the Cayley-Menger determinant. The formula for the volume of a tetrahedron with edge lengths AB = a, AC = b, AD = c, BC = d, BD = e, CD = f is given by the Cayley-Menger determinant:
V = sqrt(—CM— / 2 2 2

Text-based Communication. TextMAS exhibits the most stable inference behavior among the three paradigms. Since pretrained language models are optimized for next-token prediction over natural language, exchanging intermediate reasoning through text closely matches the model's training distribution. As a result, decoding tends to remain well-structured and logically organized.

However, this robustness comes at the cost of representational capacity. Intermediate reasoning must be compressed into discrete tokens before being passed to downstream agents. This serialization process inevitably introduces information loss, particularly for subtle intermediate computations or implicit contextual signals. In complex mathematical reasoning problems, we observe that downstream agents sometimes need to reconstruct missing details, which can lead to incomplete reasoning chains or reduced solution accuracy.

Training-free Latent Communication. LatentMAS removes this textual bottleneck by allowing agents to directly share KV caches. This enables downstream agents to attend to richer internal representations that preserve nearly all upstream reasoning information. In some cases, this additional expressivity allows agents to capture useful intermediate structure that would otherwise be lost in textual summaries.

However, we observe substantial instability in the reasoning process. Intermediate reasoning steps frequently drift off-topic, introduce irrelevant associations, or inconsistently track variables. These behaviors appear to stem from a distribution mismatch: pretrained transformers are not trained to interpret externally injected KV states as a semantic communication interface. Consequently, latent representations can function as structured but uncalibrated signals that disrupt attention patterns.

In the analyzed example, the drift is severe: although the input asks about rectangular boxes, the LatentMAS output solves a different AIME 2024 problem about a tetrahedron, before degenerating into repeated tokens. More generally, this manifests as chaotic reasoning trajectories in which correct intermediate ideas appear but are later abandoned or misused. Although some reasoning segments remain locally plausible, the global reasoning process often becomes incoherent, ultimately leading to incorrect final answers.

DiffMAS: Learned Latent Communication. DiffMAS addresses this stability–expressivity trade-off by explicitly learning how agents should communicate in latent space. Through supervised fine-tuning over full multi-agent latent reasoning trajectories, the model jointly learns both how intermediate agents should encode useful information into KV states and how downstream agents should interpret these representations.

In the same AIME example, DiffMAS produces substantially more coherent reasoning behavior. Intermediate steps remain focused on relevant constraints, variable definitions remain consistent across agents, and the reasoning trajectory follows a clear progression toward the solution. Compared to LatentMAS, we observe fewer attention failures and less reasoning drift.

Importantly, the final answer produced by DiffMAS is correct, demonstrating that supervised training aligns latent communication with task objectives rather than leaving it as an implicit and brittle interface. This suggests that the primary advantage of DiffMAS is not merely richer representations, but the ability to structure how those representations are generated and consumed across agents.

Summary of Observations. This case study highlights a fundamental tradeoff between stability and expressivity in multi-agent communication design. Text-based communication provides strong stability due to alignment with pretraining distributions but suffers from information compression. Training-free latent communication improves expressivity but introduces instability due to distribution mismatch. DiffMAS resolves this tension by learning a task-aligned latent communication protocol that preserves expressivity while maintaining stable decoding behavior.

These qualitative observations are consistent with our quantitative findings on decoding stability, perplexity, and self-consistency, which show that DiffMAS achieves both improved reasoning accuracy and more reliable multi-agent coordination.