SLC: CONSTRUCTING SEMANTIC-AWARE LATENT COMMUNICATION FOR LLM-BASED MULTI-AGENT COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent communication has shown promising potential for efficient inter-agent communication by exchanging continuous model-internal representations. Yet bypassing natural language alone does not suffice to enable effective multi-agent collaboration. Existing approaches either share prefill-derived representations or propagate each latent response along with accumulated context. Our analysis shows that decoupling latent information flow from the underlying coordination topology alters how intermediate reasoning is propagated and composed across agents, limiting effective collaboration along the multi-agent topology. To bridge this gap, we propose Semantic-aware Latent Communication (SLC), a latent communication protocol that supports effective collaboration along the multi-agent topology. Specifically, SLC employs a reusable semantic-aware input-output distribution alignment for latent response generation and transfers only context-augmented KV caches with boundary-mediated latent integration. Extensive experiments across seven benchmarks demonstrate that SLC achieves competitive accuracy with TextMAS, while enabling up to $4.1\times$ faster inference and reducing input and output token usage by up to 82.6% and 79.6%, respectively.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have achieved remarkable success in solving complex tasks through task decomposition and inter-agent communication (He et al., 2025; Rasheeda et al., 2026). In these systems, communication topologies orchestrate interactions among agents, allowing them to build on upstream reasoning and results through natural language messages (Du et al., 2024; Qian et al., 2025; Guan et al., 2026). However, such *text-based communication* incurs substantial computation, as each message requires autoregressive decoding at the sender and prefilling of the same sequence at the receiver (Ramesh & Li, 2025), while converting internal representations into discrete tokens may omit useful information (Pham et al., 2024).

To reduce inference costs, recent work has explored latent communication, in which agents exchange continuous representations, including embeddings, hidden states, or key-value (KV) caches (Liu, 2026). Existing latent communication approaches mainly follow two paradigms: *latent response accumulation* (Pham et al., 2024; Zou et al., 2026), such as CIPHER (Pham et al., 2024) and LatentMAS (Zou et al., 2026), which generates continuous responses autoregressively and retains their representations alongside earlier interaction history; and *latent context sharing* (Fu et al., 2026; Ramesh & Li, 2025; Shi et al., 2026), which directly transfers internal representations obtained during input prefill-phase without generating intermediate responses, as in KVComm's selective KV sharing (Shi et al., 2026). While both paradigms bypass repeated inter-agent natural-language message decoding and re-encoding, the information flows induced by these paradigms are decoupled from the multi-agent topology that organizes agent coordination.

As illustrated in Figure 1, decoupling inter-agent information flow from the collaboration topology alters how intermediate reasoning is propagated and composed across agents. Under *text-based communication*, each agent builds on the response of its upstream neighbor, allowing intermediate reasoning to be propagated sequentially. In contrast, *latent context sharing* induces a star-like information flow, whereas *latent response accumulation* induces a complete-graph-style information

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.

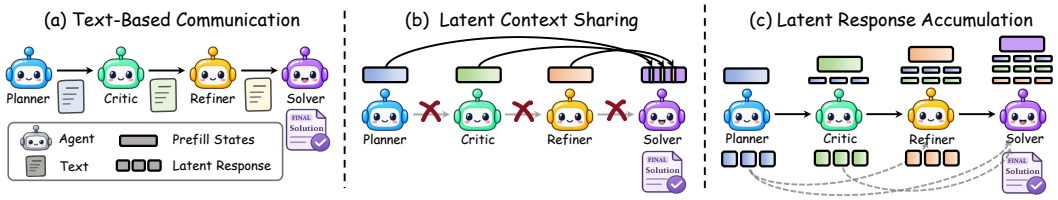
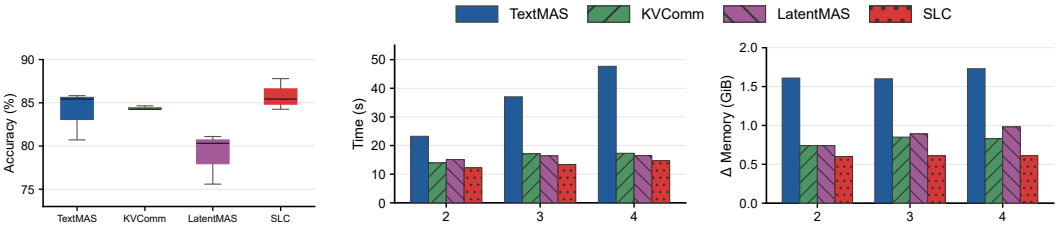


Figure 1: Comparison of three inter-agent communication paradigms induced information flow under a chain topology.



(a) Accuracy distributions **(b)** Per-sample inference time **(c)** Incremental peak GPU memory

Figure 2: Effectiveness and efficiency of representative communication methods on AQuA with the number of Qwen3-8B-based agents increases from 2 to 4 under a chain topology.

flow by carrying accumulated interaction history downstream. To examine the impact of this decoupling on multi-agent collaboration, we compare representative communication methods as the collaboration scale increases. As shown in Figure 2, existing latent communication methods achieve higher inference efficiency, yet fail to consistently realize corresponding collaboration gains as the number of agents increases. Thus we ask:

Can latent communication support effective collaboration along the multi-agent topology?

To bridge this gap, we propose **Semantic-aware Latent Communication (SLC)**, a latent communication protocol that effectively accelerate inter-agent communication while keeping the induced information flow aligned with underlying multi-agent topology. Specifically, SLC first calibrates a reusable hidden-state mapping which incorporates semantic information into latent response generation without per-step vocabulary computation. Each agent then communicates only the KV states of its newly generated response, together with a single sender-context boundary position, to its downstream neighbors, rather than propagating the full accumulated history. Through boundary-mediated latent integration, each receiver directly incorporates the transferred response into its local context without re-encoding, allowing subsequent generation to build on both local instructions and upstream reasoning along the prescribed topology.

Our contributions can be summarized as follows:

- We systematically analyze popular existing communication protocols in latent space and found these method fall short of coupling inter-agent communication with multi-agent coordination topology, limiting effective multi-agent collaboration.
- We propose SLC, a novel latent communication protocol that enables effective collaboration along the multi-agent topology. SLC combines semantic-aware latent response generation with topology-aligned latent context composition, propagating new latent responses downstream without recursive accumulation.
- Extensive experiments on mathematical and scientific reasoning, and code generation tasks show that SLC consistently outperforms existing latent communication baselines while achieving competitive performance with TextMAS. In particular, under communication-intensive multi-agent topology, SLC achieves up to $4.1\times$ faster inference while reducing system input and output tokens by up to 82.6% and 79.6% compared with TextMAS.

2 Related Work

LLM-Based Multi-Agent Systems. LLM-based multi-agent systems coordinate agents through communication topologies that structure their interactions and information flow during collaboration (Tran et al., 2025; Qi et al., 2026). Within sequential workflows, these exchanges allow agents to build on preceding outputs (Zhang et al., 2024; Hong et al., 2024), while multi-agent debate uses repeated exchanges to assess and refine responses (Du et al., 2024; Hu et al., 2025; Li et al., 2026). More recent work automates the organization of such interactions by using task feedback to select and refine agent configurations (Zhou et al., 2026; Hu et al., 2026) and adapting communication topologies to task requirements (Zhang et al., 2025a) or environmental conditions (Wu et al., 2026). Despite differences in how collaboration is organized, these approaches still rely on *text-based communication* along the designed multi-agent topology. This incurs additional inference costs, as every message forces the sender to autoregressively decode a token sequence and the receiver to prefill the same sequence, while converting internal states into discrete tokens may omit task-relevant details to downstream agents. Our work targets these communication limitations by exchanging intermediate responses through continuous model-internal representations rather than natural languages.

Latent Communication in LLM-Based MAS. Latent communication replaces inter-agent textual messages with continuous model-internal representations, including embeddings, hidden states, and KV caches (Du et al., 2026; Jin et al., 2026; Liu, 2026). One line of work explores sharing representations obtained during prefilling without generating intermediate responses. For example, AC combines intermediate activations across agents (Ramesh & Li, 2025), while KVComm shares selected prefill KV representations (Shi et al., 2026) and C2C learns to project and fuse KV caches across heterogeneous models (Fu et al., 2026). Although efficient, these *latent context sharing* schemes communicate only input-derived context and cannot convey newly synthesized responses from intermediate agents, limiting their ability to support progressive collaboration in tasks with dependent subtasks. Another line of work focuses on generating continuous responses and retaining their representations alongside previous interaction history for subsequent collaboration. CIPHER constructs continuous responses through probability-weighted token embeddings and incorporates them into the debate context (Pham et al., 2024). LatentMAS uses a linear matrix to map hidden states into latent inputs for iterative generation and transfers accumulated KV caches between agents (Zou et al., 2026). However, such *latent response accumulation* substantially expands downstream contexts, leading to repeated transmission of inherited information and increasing communication overhead as collaboration scales. In this paper, we introduce SLC, a topology-preserving latent communication paradigm that generates informative response representations while avoiding unnecessary accumulation of communication history.

3 Preliminaries and Problem Formulation

Autoregressive Generation in Transformers. Let $f_\theta(\cdot)$ denote the hidden state computation of a Transformer parameterized by θ (Vaswani et al., 2017). Given a token sequence $x_{\le t} = (x_1, \ldots, x_t)$, the input embedding matrix $W_{\text{in}} \in \mathbb{R}^{|\mathcal{V}| \times d_h}$ maps it to $E_{\le t} = [e_1, \ldots, e_t] \in \mathbb{R}^{t \times d_h}$, where $\mathcal{V}$ is the vocabulary. The model computes $[h_1, \ldots, h_t] = f_\theta(E_{\le t})$, where $h_t \in \mathbb{R}^{1 \times d_h}$ is the final-layer hidden state at position t. The next token is generated as

$$x_{t+1} = \mathcal{D}\big(h_t W_{\text{out}}^\top\big)\,, \tag{1}$$

where $W_{\text{out}} \in \mathbb{R}^{|\mathcal{V}| \times d_h}$ is the output projection matrix, and $\mathcal{D}(\cdot)$ converts vocabulary logits into a discrete token. The generated token is then mapped through W_{in} to obtain the next input embedding.

LLM-based Multi-Agent Systems. An LLM-based multi-agent system can be represented as a directed communication graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where edges specify information flows under the designed topology $\mathcal{G}$. The inter-agent communication consists of two key computations: *sender-side response generation* and *receiver-side context composition* (Guan et al., 2026). Each agent generates a response conditioned on its system prompt and received messages, while each receiver integrates responses from its incoming neighbors into its context for subsequent generation.

Latent Communication in LLM-based MAS. Latent communication exchanges continuous representations derived from internal activations rather than decoded text. Following LatentMAS (Zou et al., 2026), we consider latent response generation that replaces token decoding and re-embedding

through a linear mapping $W \in \mathbb{R}^{d_h \times d_h}$:

$$e_{t+1} = h_t W, \tag{2}$$

The embedding e_{t+1} is then appended to $E_{\leq t}$, and f_θ computes the next hidden state h_{t+1}. Repeating this process for m steps yields a latent response $H_W = [h_{t+1}, \ldots, h_{t+m}]$. Within this setting, we study how to determine W for more effective latent response generation and how to integrate the latent response into receivers' contexts under the communication topology $\mathcal{G}$.

4 METHODOLOGY

In this section, we present **SLC**, a semantic-aware latent communication framework for LLM-based MAS. As illustrated in Figure 4, SLC comprises two key stages: sender-side sematic-aware latent response generation and receiver-side latent context composition. We first introduce how to generate semantic-aware latent responses (Section 4.1), then describe latent context composition, including context-augmented KV transfer and boundary-mediated latent integration (Section 4.2). Finally, we analyze the framework communication complexity (Section 4.3).

4.1 SEMANTIC-AWARE LATENT RESPONSE GENERATION

Direct hidden state mapping enables efficient autoregressive generation in latent space. However, the performance gap observed in Figure 2(a) suggests that such efficiency does not directly translate into effective latent responses. This motivates us to examine the representations induced during latent response generation.

Representation Analysis. Using TextMAS as a reference, we compare the hidden states generated by LatentMAS and SLC via linear Centered Kernel Alignment (CKA) (Cortes et al., 2012; Kornblith et al., 2019), which measures agreement between centered inter-example similarity structures and is invariant to orthogonal transformations and isotropic scaling. We conduct this analysis with Qwen3-8B on AQuA and HumanEval+, focusing on the first agent, the Planner, to avoid differences introduced by upstream agent messages. At each layer, we compute CKA across matched questions separately for each of the averaged first five generation step scores. Figure 3 shows limited structural agreement between LatentMAS and TextMAS across layers. This observation motivates us to incorporate semantic information into the fitting of W, rather than relying on direct input-output alignment alone. As reflected by the performance improvement in Figure 2(a), this semantic-aware alignment leads to more effective latent response generation. Correspondingly, the SLC-LatentMAS comparison in Figure 3 shows that semantic-aware alignment induces distinct representational structures from direct hidden state mapping. More importantly, SLC exhibits consistently higher CKA with TextMAS than LatentMAS on both datasets, particularly in intermediate layers, indicating that these representational changes better instruct the fitting of W.

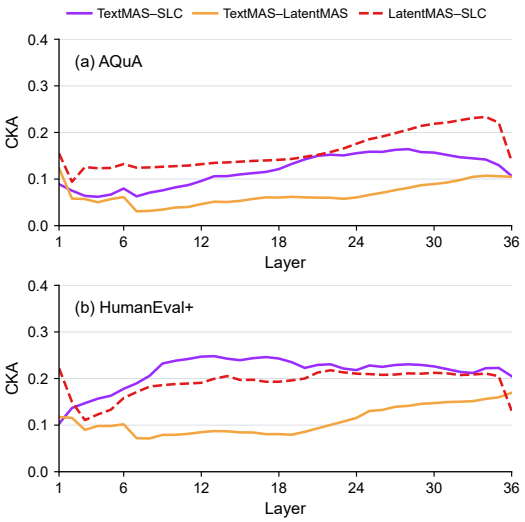


Figure 3: Layer-wise similarities on AQuA and HumanEval+, averaged over the first five generation steps.

Semantic-Aware Input-Output Distribution Alignment. Given K final-layer hidden states $\{h^{(r)}\}_{r=1}^{K}$ obtained from N calibration responses generated from unlabeled task-related inputs, we construct a semantic coefficient matrix $S \in \mathbb{R}^{K \times |\mathcal{V}|}$. For each hidden state, we retain the top-k probabilities from $\mathrm{softmax}(h^{(r)} W_{\mathrm{out}}^{\top})$, set the remaining entries to zero, and renormalize the retained weights. The resulting sparse coefficients capture multiple plausible tokens and their relative probabilities, retaining richer contextual information than a single-token representation (Pham et al., 2024;

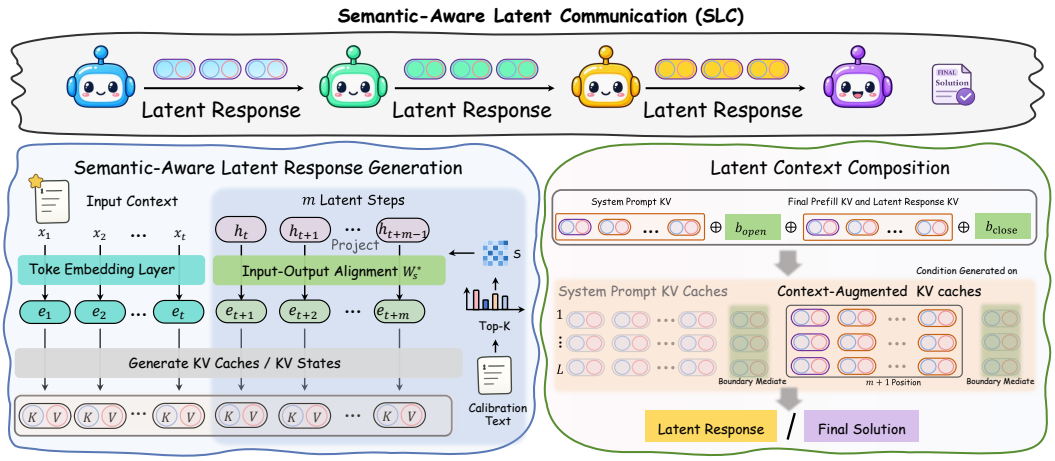


Figure 4: The overview of proposed SLC framework.

Zhang et al., 2025b). Applying the same coefficients to both matrices yields paired combinations SW_{out} and SW_{in}. We determine the semantic-aware alignment matrix by solving

$$W_s^\star = \arg\min_{W_s} \left\{ \frac{1}{K} \left\| S\left(\beta W_{\text{out}} W_s - W_{\text{in}}\right) \right\|_F^2 + \lambda \|W_s\|_F^2 \right\}, \tag{3}$$

where $\beta > 0$ is an output-side scale factor and $\lambda > 0$ controls the regularization strength. Under the shared semantic-coefficient assumption, the fitting term upper-bounds the squared Wasserstein distance between the empirical distributions formed by the rows of SW_{in} and $\beta SW_{\text{out}} W_s$. The objective admits the unique closed-form solution

$$W_s^\star = \left(\frac{\beta^2}{K} (SW_{\text{out}})^\top (SW_{\text{out}}) + \lambda I_{d_h} \right)^{-1} \frac{\beta}{K} (SW_{\text{out}})^\top (SW_{\text{in}}). \tag{4}$$

The theoretical justification, derivation, and parameter settings are provided in Appendix B. The fitted matrix maps a final-layer hidden state h to $hW_s^\star$, which serves as the Transformer's input embedding at the next latent generation step.

4.2 Latent Context Composition

We next describe how to incorporate generated latent response into downstream agents' contexts. Together, these steps connect sender's generation to receiver's reasoning while retaining sender context and avoiding recursive history transfer.

Context-Augmented KV Transfer. Building on the semantic-aware alignment in Section 4.1, SLC uses the KV states of each agent's newly generated latent response as the primary communication content, rather than forwarding the entire accumulated cache. Beyond response content, we provide a contextual cue to help the receiver associate each message with its sender's role and task context, analogous in function to explicit sender-role annotations in TextMAS. For this purpose, SLC prepends the KV states at the sender's final prefill position to the response KV. Computed under the sender's complete local prompt and any received messages, these states supplement the response with sender context at the cost of only one additional cached position.

For sender A_j, let t_j index this final prefill position. Let $\mathcal{C}_{j,t_j}$ denote the layer-wise KV at position t_j, and $\{\mathcal{C}_{j,t}\}_{t_j+1}^{t_j+m}$ the collection of layer-wise KV at the following m latent positions. The outgoing message is

$$Z_j = \mathcal{C}_{j,t_j} \oplus \{\mathcal{C}_{j,t}\}_{t=t_j+1}^{t_j+m} = \left\{ \left([K_{j,t_j}^{(l)}; \ldots; K_{j,t_j+m}^{(l)}], [V_{j,t_j}^{(l)}; \ldots; V_{j,t_j+m}^{(l)}] \right) \right\}_{l=1}^{L}, \tag{5}$$

where $K_{j,t}^{(l)}$ and $V_{j,t}^{(l)}$ denote the key and value at position t in layer l, and L is the number of Transformer layers. The operator $\oplus$ denotes ordered concatenation along the sequence dimension,

applied separately to keys and values at each layer. Each outgoing message therefore consists of the m response positions and one additional position for sender context, keeping its length at $m + 1$ regardless of the sender's prompt length or accumulated upstream history.

Boundary-Mediated Latent Integration. To distinguish upstream references from local instructions while jointly conditioning subsequent generation on both, SLC encloses incoming messages between fixed boundary token sequences b_{open} and b_{close}. Let P_i denote receiver A_i's complete local prompt, including its role, query, task requirements, and output constraints. We concatenate messages from its direct upstream agents $A_{j_1}, \ldots, A_{j_r}$ in $\mathcal{G}$ according to the prescribed workflow order, forming the logical input context

$$\mathcal{X}_i = P_i \oplus b_{\text{open}} \oplus (Z_{j_1} \oplus \cdots \oplus Z_{j_r}) \oplus b_{\text{close}}. \tag{6}$$

The receiver first prefills P_i together with b_{open}, directly inserts the received messages as KV blocks without re-encoding, and then prefills b_{close} using the updated cache. The final position of b_{close} attends to the local prompt and all inserted messages, producing a receiver-side hidden state jointly conditioned on both sources of information. We apply $W_s^\star$ to this state to obtain the first latent input, initiating response generation conditioned on the complete context $\mathcal{X}_i$ (Section 4.1). Subsequent latent steps reuse and extend the resulting cache, maintaining access to the local prompt and upstream references throughout generation.

4.3 Communication Complexity Analysis

We compare SLC with cumulative forwarding over one round in the fixed order $A_1 \prec \cdots \prec A_n$, where A_n is the final aggregator and each preceding agent contributes one local block of size ℓ. Both protocols use the same topology $\mathcal{E}_{\text{chain}} \subseteq \mathcal{E} \subseteq \mathcal{E}_{\text{complete}}$, between the chain and the complete directed acyclic graph (DAG). Each agent executes once, and each edge carries one message. Let T count block transmissions across edges, including duplicate copies, so the total volume is $T\ell$. In SLC, each edge carries only the sender's local message Z_j (Equation 5), giving $n - 1 \leq T_{\text{non}} = |\mathcal{E}| \leq n(n-1)/2$. Cumulative forwarding concatenates local and received blocks without deduplication or truncation, producing one transmitted copy per nonempty directed path. Hence, $n(n-1)/2 \leq T_{\text{acc}} \leq 2^n - n - 1$. For both protocols, the chain and complete forward DAG attain the lower and upper bounds, respectively. Thus, SLC's communication volume is $\Theta(n\ell)$ on the chain and $\Theta(n^2\ell)$ on the complete DAG, versus $\Theta(n^2\ell)$ and $\Theta(2^n\ell)$ for cumulative forwarding. Avoiding recursive history replication keeps SLC's communication proportional to the number of edges without altering the topology.

5 Experiments

5.1 Experimental Setup

We briefly summarize the experimental setup here, with further details in Appendix C.

Benchmarks. We evaluate on seven benchmarks across three categories: (1) *Mathematical Reasoning*: GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), and AIME (Art of Problem Solving, 2025); (2) *Scientific Reasoning*: ARC-Challenge (ARC-C) (Clark et al., 2018) and WinoGrande (WinoG) (Sakaguchi et al., 2020); and (3) *Code Generation*: MBPP+ and HumanEval+ (Liu et al., 2023). Detailed benchmark descriptions are provided in Appendix C.1.

Models and Baselines. We use Qwen3-8B and Qwen3-14B (Team, 2025) as backbone models. We conduct all comparisons under two multi-agent interaction topologies, *Chain* (Zhang et al., 2024) and *CompleteGraph* (Qian et al., 2025), and evaluate each communication method under both settings. Within each topology, we use text-based multi-agent communication (TextMAS) as the primary reference and compare SLC with representative latent communication methods, including the latent context sharing method KVComm (Shi et al., 2026) and the latent response transfer methods CIPHER (Pham et al., 2024) and LatentMAS (Zou et al., 2026).

Implementation Details. For SLC, we set $N = 50$, $k = \{16, 32, 64\}$, use $m = \{40, 50, 60\}$ latent steps per response, and instantiate b_{open} and b_{close} as the token sequences obtained by encoding `<latent reference>` and `</latent reference>`, respectively. The alignment matrix is computed once for each calibration configuration and reused during inference. We set all LLM

Table 1: Main result under four-agent system of *Chain* topology. We report accuracy (%) for reasoning tasks and Pass@1 (%) for code generation; and total number of system output per sample ($\#Output$), including tokens and latent representations. For accuracy and Pass@1, the best results within each model are **bold** and the second-best results are underlined.

	Math Reasoning			Scientific Reasoning		Code Generation		
Method	GSM8K	AQuA	AIME	ARC-C	WinoG	MBPP+	HumanEval+	**Avg.**
Qwen3-8B								
TextMAS	92.42	86.22	21.67	93.43	73.48	67.46	**80.49**	73.60
# Output	1,161	1,712	4,976	1,255	822	966	1,110	1,715
KVComm	87.36	77.82	18.33	62.23	67.51	59.64	51.09	60.57
# Output	176	187	1,883	11	53	244	83	377
CIPHER	92.19	82.68	**23.33**	93.26	72.93	61.64	78.05	72.01
# Output	1,345	1,997	4,181	1,385	1,275	1,138	1,651	1,853
LatentMAS	90.90	73.62	16.67	92.83	71.27	43.39	53.66	63.19
# Output	412	670	2,332	479	418	175	264	678
SLC (ours)	**93.56**	**87.80**	**23.33**	**94.11**	**75.30**	**69.05**	78.05	**74.46**
# Output	428	649	1,806	412	322	168	194	568
Qwen3-14B								
TextMAS	94.16	88.19	23.33	95.14	80.82	69.84	**84.15**	76.52
# Output	1,311	2,148	5,542	1,479	1,299	1,667	1,766	2,173
KVComm	85.44	74.80	11.67	89.33	66.93	56.08	68.29	64.65
# Output	245	485	1,540	173	164	62	100	396
CIPHER	94.09	83.86	21.67	94.88	79.87	71.16	83.54	75.58
# Output	1,384	2,283	5,135	1,770	1,425	1,921	2,001	2,274
LatentMAS	74.45	50.00	3.33	73.55	50.91	17.72	17.07	41.00
# Output	779	1,436	3,362	680	971	1,376	1,581	1,455
SLC (ours)	**94.62**	**88.58**	**25.00**	**95.56**	**80.90**	**72.75**	82.93	**77.19**
# Output	361	538	1,915	403	271	174	280	563

agents with a temperature of 0.7, a top-p of 0.8, and the maximum length of 4,096 tokens. All experiments are conducted on NVIDIA RTX-5090 GPU. Please see Appendix C.2 for more details.

5.2 Main Results

What existing latent methods save in tokens, they lose elsewhere: context sharing loses the agents' reasoning contributions, and response accumulation loses efficiency as history piles up across agents. SLC shows this loss is not inevitable: by passing on only each agent's new conclusions, it achieves the best accuracy on most tasks with linearly growing communication cost, running 4.1× faster than TextMAS with 83.9% less memory at four agents.

What is saved in tokens is lost in latent communication. Latent context sharing is among the cheapest: AC and KVComm require 3,350–4,382 s of online time and at most 0.85 GiB of additional memory, compared with 5,734–13,344 s and 1.61–3.84 GiB for TextMAS. However, because agents exchange only prefill-derived representations, adding agents contributes no new reasoning, and accuracy stays flat (84.25–84.65)for KVComm.

However, Latent response accumulation fails in two opposite ways. CIPHER keeps accuracy close to TextMAS (82.28-84.65) but is nearly as slow (5,825-10,874 s), and its memory overhead grows to 7.39 GiB with four agents, about twice that of TextMAS, as soft-token context accumulates. LatentMAS reduces online time to up to 4,430 s, but its accuracy drops from 80.31 to 55.12 as agents are added. SLC avoids both failure modes: it maintains around 87.40 accuracy at every scale while requiring up to 3,504 s of online time and 0.60-0.62 GiB of memory overhead. With four agents, it is the fastest method in online time. These results support a simple principle: bypassing text should not bypass agents' reasoning contributions. Effective latent communication must transmit

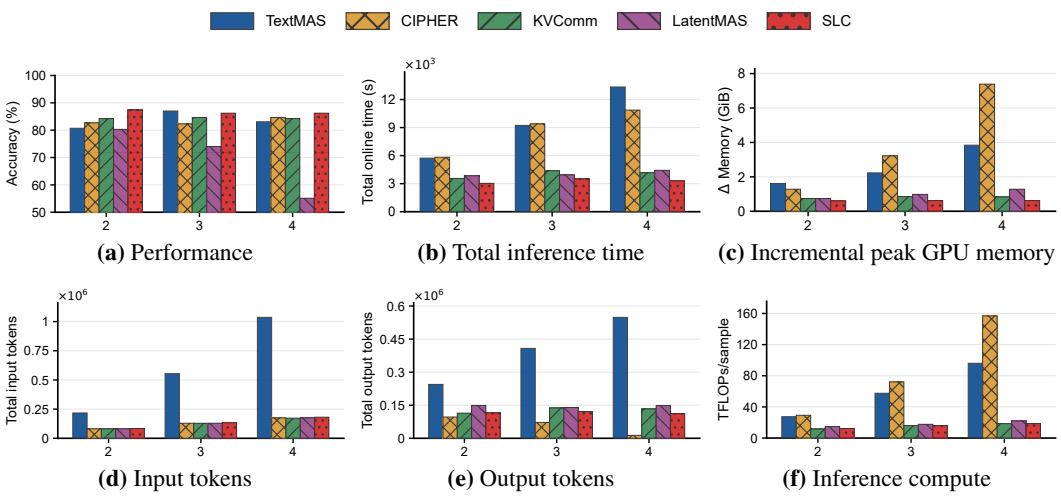


Figure 5: Effectiveness-efficiency trade-offs of compared methods as the number of agents increases on the AQuA dataset. measured using a single GPU with batch size one.

what each agent newly concludes, as latent response methods do, but without accumulating history across agents, which is precisely the combination that SLC realizes.

The trade-off between accuracy and efficiency To examine the practical implications of these designs, we vary the number of Qwen3-8B agents up 4 agents under the complete-graph topology on AQuA and measure accuracy, online inference time, and peak GPU memory (see Figure 5). The three latent communication protocols exhibit distinct trade-offs.

Scalable communication on dense multi-agent coordination topology. Complementing the trade-off analysis in figure 5, which details how communication cost scales under the 4 agent based complete graph topology in which agents interact 6 times. Because every downstream agent re-encodes all upstream text, TextMAS input tokens grow superlinearly from 217K to 1.04M as agents increase from two to four. SLC transmits only $m + 1$ positions per edge (Section 4.3), so its input tokens grow approximately linearly from 84K to 181K and its memory overhead stays constant. With four agents, SLC reduces online inference time by $4.1\times$, input tokens by 82.6%, text output tokens by 79.6%, and 74.0% when latent positions are counted. Moreover, peak memory overhead is reduced by 83.9% relative to TextMAS. More significantly, the end-to-end speedup is $2.5\times$.

Not what the sender knows, but where it left off The efficiency of SLC comes from shortening the computational path between collaborating agents. In text-based communication, every intermediate exchange introduces a full decode–transmit–prefill cycle: the sender autoregressively materializes its internal state as tokens, and the receiver subsequently processes these tokens again to recover a representation for downstream reasoning. As collaboration becomes deeper, this conversion is repeatedly inserted between otherwise continuous model computations. SLC removes this repeated representation round trip. By transferring intermediate information through the latent interface and integrating it directly into the receiver's inference process, the system avoids unnecessary intermediate decoding and reduces the amount of newly materialized context that must be processed and stored.

Importantly, SLC does not achieve efficiency by replacing the underlying reasoning model with a cheaper approximation; it reduces the overhead between reasoning stages while largely preserving the computation within each agent. This distinction becomes increasingly important as the collaboration graph grows. In a chain of multiple agents, textual communication repeatedly pays the cost of materializing and reconstructing intermediate messages at each communication boundary. SLC instead turns these boundaries into lightweight computational interfaces. The resulting efficiency gain is therefore not tied to a single faster operation, but to eliminating a recurring systems cost that compounds across multi-agent collaboration.

5.3 Ablation Study

We further ablate each component of SLC to understand its contribution. As shown in Table 2, removing semantic alignment consistently degrades performance across all three benchmarks by 3% to 6% , and removing sender context also leads to mild drop in accuracy. Taken together, these results suggest that the accuracy gap is not a cost of going latent, but a cost of what is left behind. Text is the medium through which agents convey their reasoning, not the reasoning itself; latent communication should discard the verbalization while preserving the computation it carries.

Table 2: Component ablation of SLC on Qwen3-14B for AQuA, WinoGrande, and MBPP+.

Variant	AQuA	WinoGrande	MBPP+
SLC (full)	**88.58**	**80.90**	**72.75**
w/o semantic alignment	82.28	76.56	69.58
w/o sender context	86.61	78.53	72.49
w/o boundary integration	85.43	80.43	70.90

5.4 Latent Step Analysis

We vary the number of latent steps m from 10 to 100 to examine how much latent computation each agent needs to convey its contribution (Figure 6). The answer depends on the task. On WinoGrande, accuracy is largely insensitive to m. We attribute this to the nature of the task: each WinoGrande instance asks the model to resolve an ambiguous reference by choosing between two options, a judgment that rests on a single commonsense inference rather than an extended chain of reasoning. What an agent newly concludes is therefore compact, essentially a preference between two candidates, and a short latent message suffices to carry it; additional steps add capacity the task does not need, which likely explains the non-monotonic fluctuation. On MBPP+, by contrast, Pass@1 improves steadily from 65.9 at $m = 10$ to 70.4 at $m = 90$, where it saturates. Writing a correct program requires a sequence of interdependent decisions, including choosing an algorithm, structuring control flow, and handling edge cases, and MBPP+ tightens this requirement with substantially more test cases than the original MBPP, so partial or loosely specified solutions are penalized. An agent's contribution here is structured and multi-part, and longer latent messages give it room to convey more of that structure before saturation. Together, these results suggest that the latent budget an agent needs scales with the richness of what it concludes.

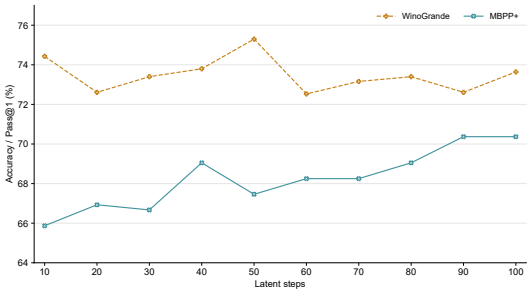


Figure 6: Effectiveness under different latent-step depths on Qwen3-8B in chain topology.

6 Conclusion

We view inter-agent communication from a different perspective: communication need not preserve the sender's text, but only needs to preserve the ability of the receiver to continue the computation. Under this view, an intermediate message, whether textual or latent, is not the object to be reconstructed, but a computational boundary connecting two otherwise separate inference processes. SLC therefore learns a lightweight interface across this boundary, transforming the sender's intermediate state into a form that can participate directly in the receiver's subsequent computation. This perspective shifts latent communication from representation transmission toward computation continuation.

Discussion: Do multi-agent gains come from coordination or communication? Our analysis suggests that multi-agent gains depend on the coupling between agents coordination and inter-agent communication: the topology defines inter-agent dependencies, while the communication mechanism determines how effectively these dependencies are realized. Future MAS design should therefore consider not only agent configurations and interaction topologies, but also communication mechanisms, potentially selecting them adaptively according to the task and coordination structure.

Ethics statement

This study raises no ethical issues. No human subjects or sensitive personal data were involved in the experiments.

Reproducibility statement

We provide sufficient information to allow the results reported in this paper to be reproduced. All experiments were conducted using publicly available datasets along with open-source models and code. Implementation details, including data selection, model architectures, and hyperparameters, are provided in Appendix.

References

Art of Problem Solving. AIME problems and solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

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. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=zj7YuTE4t8.

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. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129, 2026.

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. In *International Conference on Learning Representations*, volume 2026, pp. 43130–43158, 2026.

Yao Guan, Lin Wang, Zhihui Lu, Ziyi Wang, Wenzhu Yan, and Qiang Duan. MOC: Multi-order communication in LLM-based multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=wyynWicO5s.

Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Tianyu Hu, Zhen Tan, Song Wang, Huaizhi Qu, and Tianlong Chen. Multi-agent debate for LLM judges with adaptive stability detection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Vusd1Hw2D9.

Yuntong Hu, Yuting Zhang, Matthew Trager, Yi Zhang, Shuo Yang, Wei Xia, and Stefano Soatto. Evomas: Evolutionary generation of multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026.

Haibo Jin, Peng Kuang, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reuseable latent building blocks for multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=CzShhpY2qU.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Han Li, Yuling Shi, Shaoxin Lin, Xiaodong Gu, Heng Lian, Xin Wang, Yantao Jia, Tao Huang, and Qianxiang Wang. Swe-debate: Competitive multi-agent debate for software issue resolution. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, pp. 1380–1392, 2026.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Yingzhuo Liu. Beyond tokens: a unified framework for latent communication in llm-based multi-agent systems. *arXiv preprint arXiv:2606.05711*, 2026.

Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pp. 406–413. Cambridge University Press, 1955.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations*, volume 2024, pp. 51899–51928, 2024.

Shihao Qi, Jie Ma, Rui Xing, Wei Guo, Xiao Huang, Zhitao Gao, Jianhao Deng, Jun Liu, Lingling Zhang, Bifan Wei, et al. Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in llm-based multi-agent systems. *arXiv preprint arXiv:2605.14892*, 2026.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=W6RPXUUFic.

Zeeshan Rasheeda, Muhammad Waseema, Kai-Kristian Kemella, Mika Saari, and Pekka Abrahamsson. Llm-based multi-agent systems for code generation: A multi-vocal literature review. *arXiv preprint arXiv:2604.16321*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Xiangyu Shi, Marco Chiesa, Gerald Q Maguire Jr, and Dejan Kostic. Kvcomm: Enabling efficient llm communication through selective kv sharing. In *International Conference on Learning Representations*, volume 2026, pp. 84546–84573, 2026.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tongtong Wu, Yanming Li, Ziye Tang, Chen Jiang, Linhao Luo, Guilin Qi, Shirui Pan, and Gholamreza Haffari. CARD: Towards conditional design of multi-agent topological structures. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JgvJdICc6P.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=LpE54NUnmO.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=ByQdHPGKgU.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, 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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

A THE USE OF LARGE LANGUAGE MODELS

Large language models (LLMs) were used solely to assist with language polishing, stylistic refinement, and minor formatting to improve the clarity, coherence, and readability of the manuscript. They were not used to formulate research questions, develop methodological or theoretical ideas, conduct experiments, generate experimental results, or interpret findings. All scientific content, analyses, conclusions, and interpretations remain the sole responsibility of the authors.

B THEORETICAL JUSTIFICATION AND SOLUTION OF SEMANTIC-AWARE ALIGNMENT

We first justify the semantic-aware alignment objective in Equation 3 and then derive its closed-form solution in Equation 4.

B.1 THEORETICAL JUSTIFICATION OF THE ALIGNMENT OBJECTIVE

In SLC, *Semantic-Aware Input-Output Distribution Alignment* incorporates semantic information into the fitting of the input-output mapping, while retaining the efficiency of direct hidden-state mapping.

Following Zou et al. (2026), we assume that the corresponding input and output representations share the same semantic coefficients over the vocabulary.

Assumption B.1 (Shared Semantic Coefficients)**.** *Let P_σ be a distribution over semantic coefficient vectors $\sigma \in \mathbb{R}^{1\times|\mathcal{V}|}$ satisfying $\|\sigma\|_2 \le 1$. For $\sigma \sim P_\sigma$, the corresponding input and output representations are*

$$e = \sigma W_{\text{in}}, \qquad h = \beta\sigma W_{\text{out}}, \tag{7}$$

where $\beta > 0$ accounts for the output scale. The same coefficient vector σ is thus shared between the two representations.

For an input-output alignment matrix $W \in \mathbb{R}^{d_h\times d_h}$, let P_e and $P_{\hat{e},W}$ denote the distributions of the input representation e and the aligned representation $\hat{e} = hW$, respectively. We measure their discrepancy using the Wasserstein distance:

$$\mathcal{W}^2(P_{\hat{e},W}, P_e) \coloneqq \inf_{\gamma\in\Gamma(P_{\hat{e},W},P_e)} \mathbb{E}_{(\hat{e},e)\sim\gamma}\left[\|\hat{e}-e\|_2^2\right], \tag{8}$$

where $\Gamma(P_{\hat{e},W}, P_e)$ is the set of joint distributions with marginals $P_{\hat{e},W}$ and P_e.

Proposition B.1 (Upper Bound under Shared Semantic Coefficients)**.** *Under Assumption B.1, for any candidate semantic-aware alignment matrix $W_s \in \mathbb{R}^{d_h\times d_h}$, the shared coefficient vector σ induces a coupling between $P_{\hat{e},W_s}$ and P_e, yielding*

$$\mathcal{W}^2(P_{\hat{e},W_s}, P_e) \le \mathbb{E}_{\sigma\sim P_\sigma}\left[\|\sigma\left(\beta W_{\text{out}}W_s - W_{\text{in}}\right)\|_2^2\right]. \tag{9}$$

Proof. For a fixed W_s, draw $\sigma \sim P_\sigma$ and form the pair $(\hat{e}, e) = (\beta\sigma W_{\text{out}}W_s, \sigma W_{\text{in}})$. Let γ_σ denote the resulting joint distribution. Since the same coefficient vector σ is used to form both representations, for any measurable set $B \subseteq \mathbb{R}^{d_h}$, its marginals satisfy

$$\gamma_\sigma(B\times\mathbb{R}^{d_h}) = \Pr_{\sigma\sim P_\sigma}\left(\beta\sigma W_{\text{out}}W_s \in B\right) = P_{\hat{e},W_s}(B),$$
$$\gamma_\sigma(\mathbb{R}^{d_h}\times B) = \Pr_{\sigma\sim P_\sigma}\left(\sigma W_{\text{in}} \in B\right) = P_e(B).$$

Hence, $\gamma_\sigma \in \Gamma(P_{\hat{e},W_s}, P_e)$. Evaluating the transport cost at this admissible coupling gives

$$\begin{aligned}
\mathcal{W}^2(P_{\hat{e},W_s}, P_e) &\le \mathbb{E}_{(\hat{e},e)\sim\gamma_\sigma}\left[\|\hat{e}-e\|_2^2\right] \\
&= \mathbb{E}_{\sigma\sim P_\sigma}\left[\|\beta\sigma W_{\text{out}}W_s - \sigma W_{\text{in}}\|_2^2\right] \\
&= \mathbb{E}_{\sigma\sim P_\sigma}\left[\|\sigma\left(\beta W_{\text{out}}W_s - W_{\text{in}}\right)\|_2^2\right] \\
&\le \mathbb{E}_{\sigma\sim P_\sigma}\left[\|\sigma\|_2^2\,\|\beta W_{\text{out}}W_s - W_{\text{in}}\|_F^2\right] \\
&\le \|\beta W_{\text{out}}W_s - W_{\text{in}}\|_F^2 .
\end{aligned}$$

□

The final inequality gives the objective used by LatentMAS (Zou et al., 2026), which no longer depends on P_σ. SLC instead uses the expectation in Equation 9 to determine W_s, thereby retaining the dependence on the semantic coefficient distribution.

B.2 Solving the Semantic-Aware Alignment Matrix W_s

Based on Proposition B.1, we determine W_s by minimizing a regularized version of the Wasserstein upper bound

$$\min_{W_s} \left\{ \mathbb{E}_{\sigma \sim P_\sigma} \left[\|\sigma \left(\beta W_{\text{out}} W_s - W_{\text{in}}\right)\|_2^2 \right] + \lambda \|W_s\|_F^2 \right\}, \tag{10}$$

where $\lambda > 0$ and $\beta > 0$. Let $\mathcal{L}_{\text{sem},\lambda}(W_s)$ denote the objective function in Equation 10.

Second-Moment Formulation. To derive a closed-form solution for W_s, we express this objective as a matrix quadratic form. Since the fitting term is quadratic in σ, its dependence on P_σ is fully captured by the uncentered second moment

$$C_\sigma = \mathbb{E}_{\sigma \sim P_\sigma}[\sigma^\top \sigma] \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}. \tag{11}$$

By construction, $C_\sigma \succeq 0$. Using the trace identity for a squared norm, the objective in Equation 10 takes the equivalent form

$$\mathcal{L}_{\text{sem},\lambda}(W_s) = \operatorname{tr}\left[(\beta W_{\text{out}} W_s - W_{\text{in}})^\top C_\sigma (\beta W_{\text{out}} W_s - W_{\text{in}}) \right] + \lambda \|W_s\|_F^2. \tag{12}$$

The off-diagonal entries $(C_\sigma)_{uv} = \mathbb{E}_\sigma[\sigma_u \sigma_v]$ retain cross terms between different vocabulary coefficients.

Proposition B.2 (Closed-Form Solution of Semantic-Aware Input-Output Distribution Alignment)**.** *For fixed $C_\sigma \succeq 0$, $\beta > 0$, and $\lambda > 0$, the optimization problem in Equation 10 has the unique minimizer*

$$W_s^\star = \left(\beta^2 W_{\text{out}}^\top C_\sigma W_{\text{out}} + \lambda I_{d_h} \right)^{-1} \beta W_{\text{out}}^\top C_\sigma W_{\text{in}}. \tag{13}$$

Proof. Differentiating Equation 12 with respect to W_s yields

$$\nabla_{W_s} \mathcal{L}_{\text{sem},\lambda}(W_s) = 2\beta W_{\text{out}}^\top C_\sigma (\beta W_{\text{out}} W_s - W_{\text{in}}) + 2\lambda W_s. \tag{14}$$

Setting the gradient to zero gives the normal equation

$$\left(\beta^2 W_{\text{out}}^\top C_\sigma W_{\text{out}} + \lambda I_{d_h} \right) W_s = \beta W_{\text{out}}^\top C_\sigma W_{\text{in}}. \tag{15}$$

Since $C_\sigma \succeq 0$ and $\lambda > 0$, the coefficient matrix is positive definite. The objective is therefore strictly convex, and solving this system yields the unique minimizer in Equation 13. □

Weighted Pseudo-inverse Interpretation. To interpret how semantic weighting modifies the alignment operator, we examine the limit as $\lambda \downarrow 0$, keeping C_σ and β fixed. Let $C_\sigma^{1/2}$ denote the symmetric positive-semidefinite square root of C_σ. The fitting term can be written as $\|C_\sigma^{1/2}(\beta W_{\text{out}} W_s - W_{\text{in}})\|_F^2$. For each nonzero singular value s_j of $C_\sigma^{1/2} W_{\text{out}}$, the ridge solution applies the factor $\beta s_j / (\beta^2 s_j^2 + \lambda)$, which converges to $1/(\beta s_j)$ as $\lambda \downarrow 0$. Each column of the minimizer has zero component in the null space of $C_\sigma^{1/2} W_{\text{out}}$. Consequently, $\lim_{\lambda \downarrow 0} W_s^\star = \frac{1}{\beta} (C_\sigma^{1/2} W_{\text{out}})^\dagger C_\sigma^{1/2} W_{\text{in}}$, where $\dagger$ denotes the Moore-Penrose pseudo-inverse (Penrose, 1955). This limit is the minimum-Frobenius-norm solution of the unregularized problem, including when $C_\sigma^{1/2} W_{\text{out}}$ is rank deficient.

B.3 Empirical Solution of W_s

To obtain an empirical solution, we construct semantic coefficient vectors from model predictions in task-related contexts. Let $\{h^{(r)}\}_{r=1}^K$ denote a collection of final-layer hidden states. For each hidden state, we define

$$\hat{\sigma}_r = \text{TopKNormalize}_k \left(\text{softmax} \left(h^{(r)} W_{\text{out}}^\top \right) \right). \tag{16}$$

Here, TopKNormalize_k retains the k largest probabilities and renormalizes them to sum to one, so that $\|\hat{\sigma}_r\|_2 \leq 1$. Stacking these coefficients gives $S = [\hat{\sigma}_1; \ldots; \hat{\sigma}_K] \in \mathbb{R}^{K \times |\mathcal{V}|}$. The rows of S define

an empirical distribution over semantic coefficient vectors, with second moment $\widehat{C}_\sigma = S^\top S/K$. Replacing C_σ with $\widehat{C}_\sigma$ in Equation 12 yields

$$\widehat{\mathcal{L}}_{\mathrm{sem},\lambda}(W_s) = \frac{1}{K}\left\|S\left(\beta W_{\mathrm{out}} W_s - W_{\mathrm{in}}\right)\right\|_F^2 + \lambda \|W_s\|_F^2, \tag{17}$$

which is the objective used in Equation 3.

For fixed S, $\beta > 0$, and $\lambda > 0$, applying Proposition B.2 to the empirical objective gives the unique minimizer

$$\widehat{W}_s^\star = \left(\frac{\beta^2}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{out}}) + \lambda I_{d_h}\right)^{-1} \frac{\beta}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{in}}). \tag{18}$$

Scale and Regularization. To set the output-side scale, we choose

$$\beta = \frac{\frac{1}{K}\sum_{r=1}^{K} \|h^{(r)}\|_2}{\frac{1}{|\mathcal{V}|}\sum_{x\in\mathcal{V}} \|W_{\mathrm{in},x}\|_2}, \tag{19}$$

where $W_{\mathrm{in},x}$ denotes the input embedding of token x. This choice matches the mean norm of $h^{(r)}/\beta$ to the mean input-embedding norm. We keep β fixed throughout alignment fitting.

For regularization, we assume $SW_{\mathrm{out}} \neq 0$ and set

$$\lambda = \rho \lambda_{\max}\left(\frac{\beta^2}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{out}})\right), \tag{20}$$

where $\rho > 0$ controls the regularization strength relative to the spectral scale of the empirical alignment problem.

Proposition B.3 (Dependence on Scale and Regularization)**.** *Fix S, W_{out}, and W_{in} with $SW_{\mathrm{out}} \neq 0$. For any $\beta > 0$ and $\rho > 0$, let $\lambda = \rho\lambda_{\max}(\frac{\beta^2}{K}(SW_{\mathrm{out}})^\top(SW_{\mathrm{out}}))$ as in Equation 20. Then the empirical objective has a unique minimizer, and*

$$\kappa_2\left(\frac{\beta^2}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{out}}) + \lambda I_{d_h}\right) \leq \frac{1+\rho}{\rho}, \tag{21}$$

where $\kappa_2(\cdot)$ denotes the spectral condition number. Moreover, defining

$$M_s = \frac{1}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{out}}), \qquad B_s = \frac{1}{K}(SW_{\mathrm{out}})^\top (SW_{\mathrm{in}}), \tag{22}$$

the empirical solution satisfies

$$\beta\widehat{W}_s^\star = \left(M_s + \rho\lambda_{\max}(M_s) I_{d_h}\right)^{-1} B_s. \tag{23}$$

For fixed S, W_{out}, W_{in}, and ρ, $\beta\widehat{W}_s^\star$ is independent of β. Hence, β affects the fitted mapping only through an overall $1/\beta$ scaling, without changing the relative regularization specified by ρ.

Proof. Let $\mu_s = \beta^2\lambda_{\max}(M_s)$. Since $M_s \succeq 0$ and $SW_{\mathrm{out}} \neq 0$, we have $\mu_s > 0$. With $\lambda = \rho\mu_s$, the eigenvalues of $\beta^2 M_s + \lambda I_{d_h}$ lie in $[\rho\mu_s, (1+\rho)\mu_s]$. Hence, the coefficient matrix is positive definite, giving a unique minimizer, and its spectral condition number satisfies Equation 21.

To show how β scales the solution, we substitute $\lambda = \rho\beta^2\lambda_{\max}(M_s)$ into Equation 18 to obtain

$$\begin{aligned}\widehat{W}_s^\star &= \left[\beta^2\left(M_s + \rho\lambda_{\max}(M_s) I_{d_h}\right)\right]^{-1} \beta B_s \\ &= \frac{1}{\beta}\left(M_s + \rho\lambda_{\max}(M_s) I_{d_h}\right)^{-1} B_s.\end{aligned}$$

□

Time Complexity of Solving W_s. Given S and β, computing SW_{out} and SW_{in} requires $\mathcal{O}(Kkd_h)$ operations, since each row of S has k nonzero entries. Forming M_s and B_s then requires $\mathcal{O}(Kd_h^2)$ operations. Computing $\lambda_{\max}(M_s)$ and solving the resulting $d_h \times d_h$ regularized linear system for $\widehat{W}_s^\star$ together require $\mathcal{O}(d_h^3)$ operations. Therefore, the time complexity of computing the alignment matrix in Equation 18 is $\mathcal{O}\big(Kkd_h + Kd_h^2 + d_h^3\big)$.

Table 3: Overview of the evaluation benchmarks.

Category	Dataset	Eval. Size	Type	Metric
Mathematical Reasoning	GSM8K	1,319	Number	Acc.
	AIME	60	Number	Acc.
	AQuA	254	Multi-choice	Acc.
Scientific Reasoning	ARC-Challenge	1,172	Multi-choice	Acc.
	WinoGrande	1,267	Multi-choice	Acc.
Code Generation	MBPP+	378	Python Code	Pass@1
	HumanEval+	164	Python Code	Pass@1

C Experimental Setup

This appendix supplements Section 5.1 with dataset descriptions, baseline configurations, and implementation details.

C.1 Evaluation Benchmarks

We evaluate on seven benchmarks covering mathematical reasoning, scientific reasoning, and code generation. Table 3 summarizes their evaluation sizes, answer formats, and metrics. Below, we describe the tasks and document the corresponding data sources.

Mathematical Reasoning. GSM8K (Cobbe et al., 2021) contains human-written grade-school mathematical word problems requiring multi-step arithmetic reasoning. We evaluate all 1,319 questions in the `test` split of the `main` configuration.[1] AQuA (Ling et al., 2017), also known as AQuA-RAT, consists of algebraic word problems with five answer choices and natural-language rationales. We evaluate 254 questions from the public release, using the correct-option labels as reference answers.[2] AIME (Art of Problem Solving, 2025) consists of problems from the 2024 and 2025 American Invitational Mathematics Examination (AIME), covering algebra, geometry, number theory, and combinatorics. Each year contains 30 problems, with each problem requiring an integer answer between 0 and 999. We evaluate all 30 problems from each year using their public releases: the `train` split of AIME24[3] and the `train` split of the `default` configuration of AIME25.[4]

Scientific Reasoning. ARC-Challenge (Clark et al., 2018) contains multiple-choice grade-school science questions selected because both information-retrieval and word-co-occurrence baselines answered them incorrectly during dataset construction. We evaluate all 1,172 questions in the `test` split of the `ARC-Challenge` configuration.[5] WinoGrande (Sakaguchi et al., 2020) is a Winograd-style commonsense reasoning benchmark in which models choose between two candidate completions for a sentence. We evaluate all 1,267 labeled questions in the `validation` split of the `winogrande_debiased` configuration.[6]

Code Generation. Both benchmarks use the expanded test suites of EvalPlus (Liu et al., 2023), with functional correctness assessed by executing generated programs against the benchmark tests. MBPP+ builds on the Mostly Basic Python Problems benchmark (Austin et al., 2021), requiring the generation of short Python programs from natural-language problem descriptions. We evaluate all 378 tasks in the `test` split of the public MBPP-Plus release.[7] HumanEval+ extends HumanEval (Chen et al., 2021), in which each task requires implementing a Python function from its

[1] `https://huggingface.co/datasets/openai/gsm8k`
[2] `https://huggingface.co/datasets/deepmind/aqua_rat`
[3] `https://huggingface.co/datasets/HuggingFaceH4/aime_2024`
[4] `https://huggingface.co/datasets/yentinglin/aime_2025`
[5] `https://huggingface.co/datasets/allenai/ai2_arc`
[6] `https://huggingface.co/datasets/allenai/winogrande`
[7] `https://huggingface.co/datasets/evalplus/mbppplus`

signature and docstring. We evaluate all 164 tasks in the `test` split of the public HumanEval-Plus release.[8]

C.2 Implementation Details

We implement SLC using HuggingFace Transformers with Qwen3-8B, and Qwen3-14B (Team, 2025) as backbone models. We adopt four agents organized in a chain and complete graph topology: Planner, Critic, Refiner, and Solver, detailed prompt are shown in Figure 7. Each of the first three agents performs m latent reasoning steps, while the Solver generates the final textual answer.

For semantic alignment, we collect final-layer hidden states at all response positions from N TextMAS trajectories. Calibration uses only the inputs and does not access reference answers. We retain and renormalize the top k vocabulary probabilities to construct semantic anchors, and fit a separate alignment matrix W_s for each agent. The alignment matrices and calibrated output-side scales are fixed during inference and reused across all latent steps. We set$\rho = 10^{-4}$ to control the regularization strength λ(equation 20).

For inter-agent communication, each sender constructs a message containing the KV caches of its final prompt position and its m locally generated latent positions, yielding $m+1$ positions per Transformer layer. The receiver concatenates the incoming message with its local prompt-prefix cache along the sequence dimension and processes the remaining prompt through the 'past_key_values' interface, with corresponding attention masks and position IDs. Each sender forwards only this local message, rather than its entire accumulated cache. In the chain topology, messages are received from the immediately preceding agent; in the complete-graph topology, messages from all direct predecessors are concatenated in a fixed order.

Metrics. We report accuracy for multiple-choice tasks, and Pass@1 for code generation task. We also report two efficiency metrics: Online Computational Cost (TFLOPs/sample), which is computed as $C_{\text{online}} = \frac{F_{\text{online}}}{10^{12}N}$, where F_{online} is the estimated number of floating-point operations accumulated across all agents and evaluation samples, and N is the number of evaluation samples. One-time initialization and offline calibration are excluded. Lower values indicate less estimated online computation per sample. Incremental Peak GPU Memory (Δ Memory), which is computed as $\Delta M = \frac{\max(0, M_{\text{peak}} - M_{\text{resident}})}{2^{30}}$ GiB, where both memory quantities are measured in bytes. M_{resident} denotes the GPU memory occupied by the loaded model before evaluation. Thus, ΔM measures the additional memory required, lower values indicate smaller incremental GPU memory requirements.

D CKA Analysis Set Up

We adopt linear Centered Kernel Alignment (CKA (Cortes et al., 2012; Kornblith et al., 2019) to examine the representational relationships among TextMAS, LatentMAS, and SLC. CKA measures the agreement between the similarity structures of two sets of representations, allowing us to assess whether SLC retains representational structure associated with text-based reasoning.

We conduct this analysis under a four-agent chain topology and focus on the first agent, the Planner. The Planner receives the same question and initial prompt across methods, without conditioning on messages from preceding agents. In contrast, downstream agents may receive method-dependent upstream messages, introducing additional variation in their inputs and representations. Analyzing the Planner therefore reduces confounding effects from differences in inter-agent communication and provides a more controlled comparison. Specifically, we extract the Planner's representations at the first five generation positions and compute CKA separately at each Transformer layer using matched questions.

For a fixed generation position and layer, let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{N \times d}$ denote the representations from two methods, where corresponding rows refer to the same question, N is the number of common valid questions, and d is the representation dimension. We construct the linear Gram matrices $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$

[8] https://huggingface.co/datasets/evalplus/humanevalplus

Chain topology: Planner → Critic → Refiner → Solver

System Prompt: You are a helpful assistant.

Planner:
You are a Planner Agent. Design a clear, concise, step-by-step plan for solving the input question.

Input Question:
{question}

Critic:
You are a Critic Agent. Evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan. Review the plan and question and output:
(1) original plan contents
(2) constructive feedback on the original plan.

Input Question:
{question}

{previous_information}

Refiner:
You are a Refiner Agent. Produce a correct, concise, and improved step-by-step plan for solving the input question. Use the available information from previous agents to correct errors and fill in missing steps.

Input Question:
{question}

{previous_information}

Solver:
You are the final Solver Agent. Solve the target question while using information from previous agents only as reference.

Target Question:
{question}

{previous_information}

{task_instruction}

Figure 7: Prompts used for the chain-structured multi-agent system.

and $\mathbf{L} = \mathbf{Y}\mathbf{Y}^\top$. Their alignment is defined as

$$\mathrm{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\mathcal{D}_{\mathrm{HSIC}}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathcal{D}_{\mathrm{HSIC}}(\mathbf{K}, \mathbf{K})\mathcal{D}_{\mathrm{HSIC}}(\mathbf{L}, \mathbf{L})}}, \tag{24}$$

where $\mathcal{D}_{\mathrm{HSIC}}$ is the empirical Hilbert–Schmidt independence criterion. We use its original, non-debiased estimator:

$$\mathcal{D}_{\mathrm{HSIC}}(\mathbf{K}, \mathbf{L}) = \frac{1}{(N-1)^2} \operatorname{tr}(\mathbf{KHLH}), \tag{25}$$

where $\mathbf{H} = \mathbf{I}_N - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ is the centering matrix. Each CKA score is computed over the common valid question set, rather than by averaging separately computed mini-batch scores.

E ADDITIONAL RESULTS

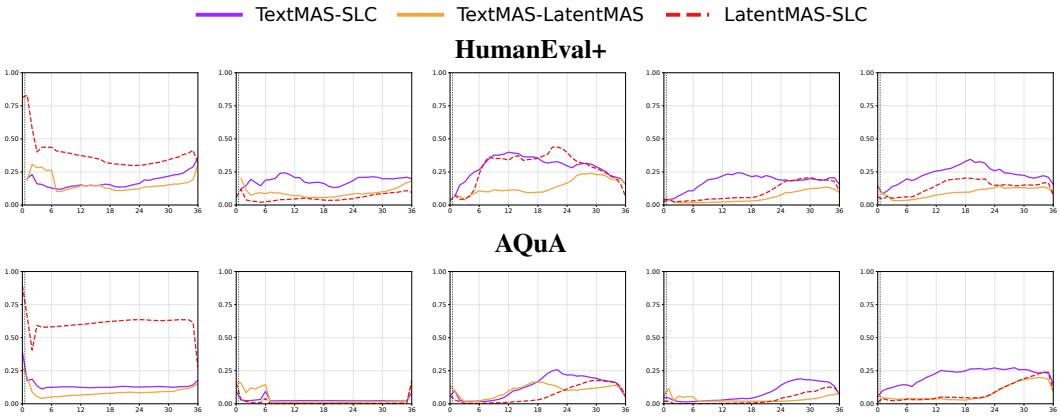


Figure 8: Layer-wise pairwise representation similarity measured by linear CKA on HumanEval+ and AQuA. Each row corresponds to one dataset, and columns show the first five generation positions from left to right. Each panel compares TextMAS-SLC, TextMAS-LatentMAS, and LatentMAS-SLC. The horizontal axis denotes representation depth, with position 0 representing the input representation and positions 1-36 corresponding to Transformer hidden layers, and the vertical axis reports CKA.

Table 4: Sensitivity analysis of ρ and corresponding λ of each agent on AQuA.

ρ	Planner λ	Critic λ	Refiner λ	Accuracy (%)
10^{-6}	6.02×10^{-3}	4.16×10^{-3}	6.52×10^{-3}	85.43
10^{-5}	6.02×10^{-2}	4.16×10^{-2}	6.52×10^{-2}	87.40
10^{-4}	6.02×10^{-1}	4.16×10^{-1}	6.52×10^{-1}	**87.80**
10^{-3}	6.02	4.16	6.52	85.43
10^{-2}	6.02×10^{1}	4.16×10^{1}	6.52×10^{1}	85.04

Table 5: Sensitivity analysis of Top-k on AQuA.

Top-k	1	2	4	8	16	32	64	128
Accuracy (%)	85.04	87.01	85.43	86.22	83.46	**87.80**	84.65	86.22

Table 6: Detailed comparison of different communication methods with varying numbers of agents under Qwen3-8b-based CompleteGraph multi-agent topology. TextMAS is shaded as the reference baseline. For each non-TextMAS method, the gray numbers report the relative improvement over TextMAS. Positive values indicate improvement.E2E represents the end-to-end time. Output position denotes total number of system output, including tokens and latent representations.

Method	Acc. ↑	E2E Time ↓	Inference Time ↓	ΔPeak Mem. ↓	TFLOPs/sample ↓	Input Token ↓	Output Token ↓	Output Pos. ↓
2 Agents								
TextMAS	80.71	5761.17	5733.53	1.61	27.47	217,135	244,492	244,492
KVComm	84.25	3612.02	3565.71	0.74	11.67	81,636	114,085	114,085
Improve	+4.39%	+37.30%	+37.81%	+54.04%	+57.53%	+62.40%	+53.34%	+53.34%
LatentMAS	80.31	3895.22	3845.63	0.74	14.76	81,636	149,122	159,282
Improve	-0.50%	+32.39%	+32.93%	+54.04%	+46.28%	+62.40%	+39.01%	+34.85%
CIPHER	82.68	5852.91	5825.40	1.28	29.24	81,636	96,574	237,800
Improve	+2.44%	-1.59%	-1.60%	+20.50%	-6.45%	+62.40%	+60.50%	+2.74%
SLC	87.40	3695.68	3036.45	0.60	12.23	83,668	115,329	125,489
Improve	+8.29%	+35.85%	+47.04%	+62.73%	+55.47%	+61.47%	+52.83%	+48.67%
3 Agents								
TextMAS	87.01	9262.82	9235.08	2.23	57.54	553,974	407,722	407,722
KVComm	84.65	4428.16	4381.62	0.85	16.02	128,538	137,454	137,454
Improve	-2.71%	+52.19%	+52.55%	+61.88%	+72.16%	+76.80%	+66.29%	+66.29%
LatentMAS	74.02	3984.76	3935.11	0.98	17.69	129,046	139,288	159,608
Improve	-14.93%	+56.98%	+57.39%	+56.05%	+69.26%	+76.71%	+65.84%	+60.85%
CIPHER	82.28	9440.78	9394.93	3.22	72.10	129,046	71,823	380,265
Improve	-5.44%	-1.92%	-1.73%	-44.39%	-25.31%	+76.71%	+82.38%	+6.73%
SLC	86.22	4931.70	3503.93	0.62	15.95	133,110	121,437	141,757
Improve	-0.91%	+46.76%	+62.06%	+72.20%	+72.29%	+75.97%	+70.22%	+65.23%
4 Agents								
TextMAS	83.07	13371.80	13343.86	3.84	96.03	1,036,443	548,600	548,600
KVComm	84.25	4216.33	4169.74	0.83	18.46	172,646	133,278	133,278
Improve	+1.42%	+68.47%	+68.75%	+78.39%	+80.77%	+83.34%	+75.71%	+75.71%
LatentMAS	55.12	4479.79	4430.04	1.28	21.92	174,678	148,357	178,837
Improve	-33.65%	+66.50%	+66.80%	+66.67%	+77.17%	+83.15%	+72.96%	+67.40%
CIPHER	84.65	10901.31	10873.59	7.39	156.86	174,678	13,228	452,604
Improve	+1.90%	+18.48%	+18.51%	-92.45%	-63.35%	+83.15%	+97.59%	+17.50%
SLC	86.22	5341.36	3288.35	0.62	18.54	180,774	112,066	142,546
Improve	+3.79%	+60.06%	+75.36%	+83.85%	+80.69%	+82.56%	+79.57%	+74.02%