MANY MINDS, ONE PASS: MULTIPLEXING COLLABORATIVE REASONING VIA SUPERPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems (MASs) enable emergent system-level intelligence, driving a paradigm shift in problem solving toward collaborative reasoning. However, standard MAS reasoning procedures process each query independently, incurring substantial redundant computation and token expenditure as query volume grows and creating a bottleneck to large-scale deployment. To address this challenge, we propose REMUX, a novel token-efficient collaborative reasoning framework that uses reasoning *superposition* to process multiple queries concurrently in a single pass through a MAS. Given a batch of queries, REMUX encodes their reasoning signals in approximately mutually orthogonal subspaces and superposes them in a shared latent space. By preserving near-orthogonality throughout reasoning, REMUX enables a demultiplexing agent to *de-multiplex* the signals with minimal cross-query interference and recover the corresponding answers. Extensive experiments demonstrate that, by multiplexing up to 20 queries concurrently, REMUX reduces token consumption by a factor of 10.1–16.3 and achieves a $1.9\times$–$10\times$ speedup over state-of-the-art baselines while maintaining comparable task accuracy. The code for REMUX will be released publicly.

1 INTRODUCTION

Multi-agent systems (MAS) built upon large language models (LLMs) have emerged as a powerful paradigm for complex reasoning tasks. By assigning specialized roles, such as planner, critic, and refiner, to multiple agents, MAS decomposes problems into manageable sub-tasks and aggregates diverse perspectives, yielding substantial gains in mathematical problem-solving (Rizvi-Martel et al., 2026; Zhou et al., 2025), scientific analysis (Ghareeb et al., 2026; Xiong et al., 2026), and commonsense reasoning (Tu et al., 2026; Alazraki et al., 2026). However, these improvements come at a significant token cost, as every query traverses the complete multi-agent pipeline and the majority of generated tokens are consumed by intermediate steps, such as drafting, commenting, and revising proposals, rather than the final answer. Consequently, the practicality of MAS is still limited by token consumption, which grows sharply with the number of participating agents (Zhang et al., 2025a; Chen et al., 2026).

Recent work reduces token consumption by shifting reasoning and inter-agent communication from text into continuous latent spaces. Representative approaches reason directly over hidden states rather than tokens (Hao et al., 2025; Gozeten et al., 2026), compress inter-agent messages into latent vectors (Li et al., 2026; Liu, 2026; Du et al., 2026), or combine both in a fully latent framework (Zou et al., 2026). These methods nevertheless retain a common *one-to-one* paradigm, in which each query's reasoning trajectory is represented by a separate latent vector sequence. Two observations suggest that these sequences could instead be shared across queries within a batch. First, queries are processed by the same set of agents under a common collaboration topology. Second, high-dimensional LLM latent vectors have greater representational capacity than a single reasoning step requires (Zhang et al., 2025b), suggesting the potential to encode multiple reasoning signals simultaneously. Prior work on data multiplexing (Murahari et al., 2022; Menet et al., 2023) demonstrates that multiple inputs can be superposed into a shared representation with the same dimensionality as an individual input and processed concurrently in a single forward pass through a neural network. MASs offer a particularly promising setting for such superposition: their long reasoning pipelines contain many intermediate steps with substantial representational redundancy. Sharing latent representations across queries throughout these pipelines could therefore amortize token costs and yield

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.

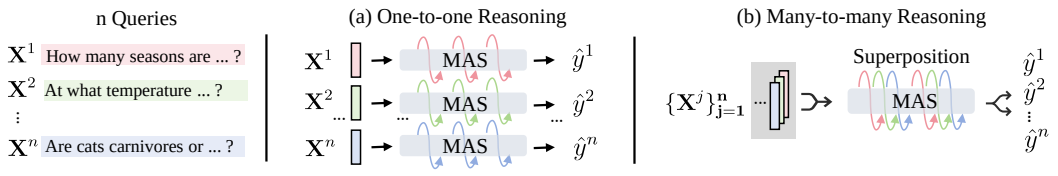


Figure 1: Schematic of our superposition method compared with one-to-one reasoning in MAS.

substantial savings. Building on these observations, we ask: **Can a multi-agent system reason over multiple queries using a shared sequence of latent vectors, without increasing their dimensionality, while still recovering a separate answer for each query?**

We propose REMUX, a framework that processes the collaborative reasoning of n queries in one pass via *multiplexing*. As illustrated in Fig. 1(b), REMUX projects a batch of queries into a shared latent space, sums them into a single input, and lets the first $m-1$ agents reason over this superposition state. The final agent then separates the shared trajectory and generates a textual answer for each query. As a result, the intermediate reasoning is performed once per batch instead of once per query. Notably, REMUX is also applicable to single agent setting when $m = 2$.

To enable REMUX, we address three technical challenges.

The first challenge lies in preserving query-specific semantics when combining token embeddings from multiple queries into a shared representation. Naively superposing these embeddings can introduce cross-query interference and distort their semantic information. To address this challenge, REMUX employs a tailored multiplexing function that maps token embeddings from different queries into nearly orthogonal subspaces of a shared representation space. This construction establishes a *continuous superposition space* that supports reasoning over superposed representations while preserving the semantic features of individual queries. We provide a theoretical proof of this near-orthogonality property for LLM token embeddings in Appendix A.1.

The second challenge lies in mitigating interference at intermediate steps along the reasoning trajectories. We find that, even when input representations are near-orthogonal, the excessive information from superposed queries induces computational interference, leading to representation drift that compounds as reasoning progresses (Fig. 3(b)). To address this issue, we enforce per-step alignment to anchor reasoning prefixes in the continuous superposition space. Specifically, we introduce a Continuous Supervised Objective (CSO) that constrains these prefixes to match ground-truth superposed signals constructed from standard one-to-one reasoning trajectories, thereby suppressing representation drift with reduced cross-query interference.

The third challenge lies in generating query-specific answers from superposed states once reasoning is complete. Standard LLM decoding strategies cannot directly recover individual answers from these states as they lack an explicit mapping from superposed representations to query-specific semantic representations. To address this challenge, we store the complete superposed reasoning trajectories in the KV cache. We then use the original queries to construct query-specific attention vectors that extract the corresponding reasoning traces from the cached trajectories for final answer generation. This disentanglement mechanism is differentiable end to end, enabling joint training of the entire multi-agent system using standard gradient descent.

We conduct experiments under two typical multi-agent settings (sequential and hierarchical) across four LLM families, including Qwen3-4B/8B/14B, Ministral-3-8B, Mistral-Nemo-12B, Moonlight-16B-A3B, and DeepSeek-R1-Distill-Qwen series (8B/14B/32B). Evaluations of REMUX are conducted on 8 MAS datasets covering mathematical problem solving, logical reasoning and code generation. Empirical results demonstrate that REMUX reduces token overhead by 10.1–$16.3\times$ and achieves a 1.9–$10\times$ speedup over state-of-the-art baselines, while retaining comparable accuracy.

Our contributions are summarized as follows:

- **A New Paradigm for Token-Efficient Reasoning**: To the best of our knowledge, REMUX is the first latent collaboration framework to multiplex multiple queries in a single pass through a MAS, substantially reducing token consumption and accelerating inference.

- **Multiplexing for Collaborative Reasoning**: We introduce mechanisms including query orthogonalization, superposition space alignment, and trajectory disentanglement to enable multiplexed reasoning across queries, bridging the gap between data multiplexing and collaborative reasoning.
- **Broad Compatibility**: REMUX is compatible with various MAS architectures and backbone models while remaining independent of and complementary to existing token-saving techniques.

2 RELATED WORKS

Token Saving. Existing approaches can be broadly grouped into three major categories: (i) *Continuous chain-of-thought* (Hao et al., 2025; Zhang et al., 2025b; Zhu et al., 2025; Gozeten et al., 2026) performs reasoning in a continuous latent space (instantiated as hidden states or concept tokens) rather than textual tokens, where each latent step consolidates multiple textual steps and thus shortens the reasoning sequence. Such consolidation is practicable because a high-dimensional continuous embedding can encode multiple relational concepts simultaneously. (ii) *Communication management* compresses messages relayed between agents into a few latent vectors (Li et al., 2026) and even eliminates text entirely, enabling agents to communicate in a continuous latent space (Liu, 2026; Du et al., 2026). By transmitting latent vectors directly, agents convey what would otherwise require tens or hundreds of text tokens. (iii) *Latent collaboration* (Zou et al., 2026) provides a fully latent framework that unifies reasoning and communication within the same continuous space, yielding further savings. However, none of these methods reduces token consumption through multiplexed processing, an orthogonal approach that motivates the design of REMUX.

Data Multiplexing. Recent work (Sun et al., 2022; Elhage et al., 2022) investigates *superposition*, whereby neural networks encode more features than there are dimensions in their representation space. Related work by Murahari et al. (2022), Murahari et al. (2023), Menet et al. (2023), and Song et al. (2026) introduces data multiplexing techniques that enable small-scale discriminative neural networks to process multiple inputs in a single forward pass. These approaches multiplex multiple input embeddings into a single superposed representation, allowing the network to handle concurrent inputs within a single forward pass and thereby substantially improving throughput. However, these methods are not directly applicable to multiplexed reasoning in MASs, where generative large language models serve as collaborating agents. To bridge this gap, REMUX addresses the aforementioned challenges and, to the best of our knowledge, is the first framework designed to multiplex collaborative reasoning across queries, offering a new paradigm for token-efficient multi-agent reasoning.

3 PRELIMINARIES

In this section, we first revisit the standard one-to-one pipeline for collaborative reasoning in multi-agent systems. Building on this foundation, we introduce the task that REMUX aims to address.

3.1 STANDARD COLLABORATIVE REASONING PIPELINE

Let $\mathcal{M} = \{\mathbf{A}_i\}_{i=1}^{m}$ be a multi-agent system (MAS) comprising m LLM-based agents. Given a batch of queries $\mathcal{B} = \{\mathbf{X}^j\}_{j=1}^{n}$, the system $\mathcal{M}$ orchestrates agent interactions to collaboratively produce the final answer for each $\mathbf{X}^j$ at inference time. This pipeline consists of the following two modules.

Autoregressive Reasoning. Let $\mathbf{E} \in \mathbb{R}^{v \times d}$ denote the vocabulary embedding matrix, where each row corresponds to a d-dimensional token embedding. Each agent $\mathrm{A}_i \in \mathcal{M}$ reasons autoregressively over $\mathbf{X}^j$. At step t, A_i outputs the last-layer hidden state conditioned on the reasoning prefix $\hat{\mathbf{e}}^j_{i,<t}$:

$$\mathbf{h}^j_{i,t} = \mathrm{A}_i\left(\cdot \,\middle|\, \hat{\mathbf{e}}^j_{i,<t}, \mathbf{X}^j\right), \tag{1}$$

where token embedding $\hat{\mathbf{e}}^j_{i,t} \in \mathbb{R}^d$ is decoded from $\mathbf{h}^j_{i,t} \in \mathbb{R}^d$ via the LM head and vocabulary lookup. In the autoregressive loop, $\hat{\mathbf{e}}^j_{i,t}$ serves as the prefix input for the next reasoning step. After

c reasoning steps, A_i produces the discrete token-embedding sequence $\mathbf{R}_i^j = \left[\hat{\mathbf{e}}_{i,1}^j, \ldots, \hat{\mathbf{e}}_{i,c}^j\right]$. Following prior work (Gozeten et al., 2026; Zhu et al., 2025), we assume for simplicity that the answer depends only on the last-step token $\hat{\mathbf{e}}_{i,c}^j$, the same framework naturally extends to the sequence $\mathbf{R}_i^j$.

Multi-agent Collaboration. In MAS, collaborative reasoning relies not only on the input query $\mathbf{X}^j$ but also on $\hat{\mathbf{e}}_{i-1,c}^j$, the outputs of the preceding agents. We therefore reformulate Eq. 1 as a conditional recursive process. The last-step (i.e., c-th step) hidden state of the i-th agent is given by:

$$\mathbf{h}_{i,c}^j = \begin{cases} \mathrm{A}_i\left(\cdot \,\middle|\, \hat{\mathbf{e}}_{i,<c}^j, \mathbf{X}^j\right) & i = 1, \\ \mathrm{A}_i\left(\cdot \,\middle|\, \hat{\mathbf{e}}_{i,<c}^j, \mathbf{X}^j, \hat{\mathbf{e}}_{i-1,c}^j\right) & 1 < i \leq m. \end{cases} \tag{2}$$

where $\hat{\mathbf{e}}_{i,c}^j$ is obtained from $\mathbf{h}_{i,c}^j$ following the same decoding procedure defined above. The final predictive answer $\hat{\mathbf{e}}_{m,c}^j$ from the m-th agent encapsulates the cumulative result of collaborative reasoning, as its generation is conditioned on the outputs of all preceding agents. Notably, our discussion is not limited to any specific MAS. Instead, we model the collaboration process via message passing, which can be naturally extended to various architectures, e.g., sequential and hierarchical.

3.2 Problem Formalization

Given n queries $\mathcal{B} = \{\mathbf{X}^j\}_{j=1}^n$ with $\mathbf{X}^j \in \mathbb{R}^{k \times d}$, a multi-agent system $\mathcal{M} = \{A_i\}_{i=1}^m$ solves them collaboratively. In the best case, the standard reasoning pipeline has a token cost $\text{Token}_{\text{text}} = \mathcal{O}(nmc)$, achieved when each agent is called exactly once, where c denotes the average reasoning length per agent. Our goal is to learn a superposition-based reasoning mapping:

$$\mathcal{F}_{\text{sup}} \colon (\mathcal{M}, \mathcal{B}) \mapsto \{\hat{\mathbf{e}}_{m,c}^j\}_{j=1}^n, \tag{3}$$

such that $\text{Token}_{\text{sup}} \ll \text{Token}_{\text{text}}$ while minimizing task loss between predictions and ground truth:

$$\sum_{j=1}^{n} \text{loss}\left(\hat{\mathbf{e}}_{m,c}^j, \mathbf{e}_{m,c}^j\right). \tag{4}$$

4 Methodology

Building upon the task introduced in Section 3.2, we now present the overall design of ReMux, as illustrated in Fig. 2. ReMux consists of three key components: (i) a superposition-based reasoning method that concurrently processes a batch of queries in continuous space (Section 4.1); (ii) an alignment strategy applied at each reasoning step to mitigate cross-query interference (Section 4.2); and (iii) a disentangling strategy that separates the superimposed trajectories for answer generation (Section 4.3). A theoretical justification of multiplexing is provided in Appendix A.

4.1 Reasoning by Superposition for Batch Queries

The standard pipeline operates in a discrete semantic space, which makes it impossible to simultaneously represent multiple semantic concepts within one input embedding. To address this limitation, ReMux introduces a continuous superposition space, which is defined as follows.

Definition 4.1 (Continuous Superposition Space)**.** Let $\mathcal{E}_{I,\cdot} = \{\mathbf{e}^j\}_{j=1}^n \subset \mathbb{R}^d$ denote a subset of token embeddings from the vocabulary embedding matrix $\mathbf{E}$. Given a collection of linear transformations $\{\phi^j\}_{j=1}^n$, we define the *continuous superposition space* $\mathcal{S}$ as the set of uniform weighted sums over the transformed embeddings:

$$\mathcal{S} = \left\{ \sum_{j=1}^{n} \frac{1}{n} \phi^j\left(\mathbf{e}^j\right) \;\middle|\; \mathbf{e}^j \in \mathcal{E}_{I,\cdot} \right\} \subset \mathbb{R}^d, \tag{5}$$

where each $\phi^j(\mathbf{e}) := \mathbf{g}^j \odot \mathbf{e}$ is the Hadamard product with a fixed random vector $\mathbf{g}^j$.

Theorem 4.1 (Near-orthogonality)**.** *Consider a set of queries $\{\mathcal{X}^j\}_{j=1}^n$ in the continuous superposition space, where each query is projected onto the subspace $\mathcal{V}^j = \text{span}\left(\{\phi^j(\mathbf{e}) \mid \mathbf{e} \in \mathcal{X}^j\}\right)$, such that $\mathcal{V}^j$ is nearly orthogonal to $\mathcal{V}^{j'}$ for all $j \neq j'$.*

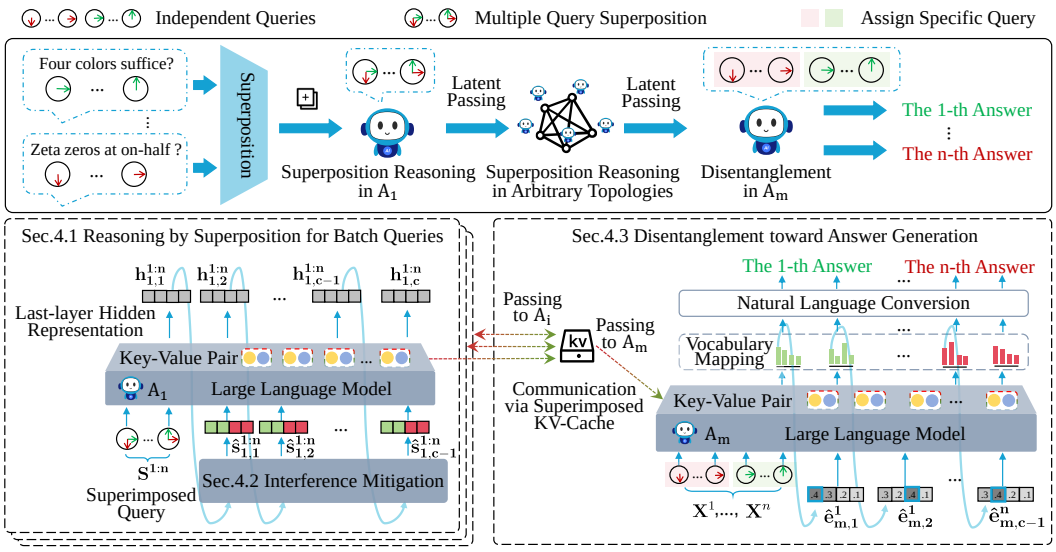


Figure 2: Overview of REMUX. For the first $m-1$ agents, REMUX conducts reasoning by projecting a batch of queries into the continuous superposition space (details in Sec. 4.1). To mitigate interference, the inputs at each reasoning step are aligned to the continuous superposition space (details in Sec. 4.2). REMUX accumulates the complete superposition reasoning trajectory within the KV cache, which the final agent leverages to generate the explicit final answer (details in Sec. 4.3).

Theorem 4.1 shows that token embeddings from distinct input queries are projected into mutually nearly orthogonal, distinguishable sub-regions, with the full proof provided in Appendix A.1.

Reasoning in Superposition. Inspired by Theorem 4.1, REMUX feeds a set of superimposed queries into the LLM backbone to compute them concurrently. This capability stems from the linear projection of self-attention, which natively supports projecting superimposed signals onto approximately separate subspaces. Detailed theoretical justifications are provided in Appendix A.2. Specifically, given a batch of queries $\mathcal{B} = \{\mathbf{X}^j\}_{j=1}^n$ with each query $\mathbf{X}^j = [\mathbf{x}_1^j, \ldots, \mathbf{x}_k^j] \in \mathbb{R}^{k\times d}$ represented as a sequence of token embeddings, REMUX first projects these queries from the semantic space into the continuous superposition space:

$$\mathbf{S}^{1:n} = \sum_{j=1}^{n} \phi^j(\mathbf{X}^j), \tag{6}$$

where $\mathbf{S}^{1:n} = [\mathbf{s}_1^{1:n}, \ldots, \mathbf{s}_k^{1:n}] \in \mathbb{R}^{k\times d}$ denotes the latent sequence shared across n queries. Conditioned on the superimposed query $\mathbf{S}^{1:n}$ and the prefix $\mathbf{h}_{i,<t}^{1:n}$, A_i conducts autoregressive reasoning:

$$\mathbf{h}_{i,t}^{1:n} = \mathrm{A}_i\big(\cdot \,\big|\, \mathbf{h}_{i,<t}^{1:n}, \mathbf{S}^{1:n}\big), \tag{7}$$

where $\mathbf{h}_{i,t}^{1:n} \in \mathbb{R}^d$ denotes the superimposed last-layer hidden embedding of agent A_i at reasoning step t, which is concatenated to the reasoning prefix for the next step. Overall, the above procedure described by Eqs. 6 and 7 reshapes the autoregressive reasoning process within a continuous latent space, such that the output at each step encapsulates reasoning across n queries.

4.2 INTERFERENCE MITIGATION

Despite the near-orthogonality of superimposed input queries after applying the transformations $\{\phi^j\}_{j=1}^n$, interference may still occur. This is because superimposed hidden representations tend to drift over successive reasoning steps, resulting in missing reasoning trajectories for certain queries.

We identify that the key to mitigating interference is to align input prefix to the continuous superposition space at every reasoning step, thereby preserving information from multiple reasoning trajectories in the shared representations. To this end, we first employ a learnable matrix $\mathbf{W}_a \in \mathbb{R}^{d\times d}$

that transforms the last-layer hidden representations $\mathbf{h}_{i,t}^{1:n}$ to input prefix $\hat{\mathbf{s}}_{i,t}^{1:n}$. Accordingly, Eq. 7 is reformulated as:

$$\mathbf{h}_{i,t}^{1:n} = \mathrm{A}_i\big(\cdot \,\big|\, \hat{\mathbf{s}}_{i,<t}^{1:n}, \mathbf{S}^{1:n}\big), \quad \text{where} \quad \hat{\mathbf{s}}_{i,t}^{1:n} = \mathbf{h}_{i,t}^{1:n}\mathbf{W}_a \subset \mathcal{S}. \tag{8}$$

To provide effective learning signals for $\mathbf{W}_a$, we propose a *Continuous Supervised Objective (CSO)* that aligns input prefix $\hat{\mathbf{s}}_{i,t}^{1:n}$ toward the continuous superposition space. Our training procedure operates on two parallel pipelines. In the *teacher* pipeline, standard one-to-one inference is performed across n queries, and the resulting reasoning outputs $\{\mathbf{e}_{i,t}^{j}\}_{j=1}^{n}$ are superimposed to form the supervision signal, as formulated by $\mathbf{s}_{i,t}^{1:n} = \sum_{j=1}^{n} \phi^j\big(\mathbf{e}_{i,t}^{j}\big)$. In the *student* pipeline, we run the superposition inference and apply cosine distance loss to optimize representation consistency between the input prefix $\hat{\mathbf{s}}_{i,t}^{1:n}$ and the supervision signal $\mathbf{s}_{i,t}^{1:n}$. The CSO objective function is formally expressed as follows:

$$\mathcal{L}_{\text{CSO}} = \sum_{t=1}^{c} \mathrm{Cosine}\big(\hat{\mathbf{s}}_{i,t}^{1:n}, \mathbf{s}_{i,t}^{1:n}\big). \tag{9}$$

With the CSO objective, REMUX ensures that superposition reasoning incorporates trajectories for all queries, preventing drift toward certain queries. A visualization of the alignment effectiveness is provided in Fig. 3 (b).

4.3 DISENTANGLEMENT TOWARD ANSWER GENERATION

Given the superposed output $\hat{\mathbf{s}}_{i,c}^{1:n}$ from the i-th agent, REMUX first conducts inter-agent collaboration via superposition communication and deploys a disentangling strategy at the m-th agent to generate answers for each query.

Communication via Superimposed Key-Value (KV) Cache. Directly passing the last-layer latent output as the context to the next agent A_{i+1} results in the loss of intermediate-layer superposition states, which provide critical information for trajectory disentanglement. To preserve the complete superposed computational trajectory, we copy the KV cache from A_i to A_{i+1} for communication, since the KV cache acts as dynamic working memory that stores intermediate states(Fu et al., 2026; Zou et al., 2026). The KVCache-Centric communication is formalized as:

$$\mathrm{A}_{i+1}(\cdot|\hat{\mathbf{s}}_{i+1,<c}^{1:n}, \mathbf{S}^{1:n}, \hat{\mathbf{s}}_{i,c}^{1:n}) = \mathrm{A}_{i+1}(\cdot|\hat{\mathbf{s}}_{i+1,<c}^{1:n}, \mathbf{S}^{1:n}, \mathbf{K}_i^{1:n}, \mathbf{V}_i^{1:n}), \tag{10}$$

where $\hat{\mathbf{s}}_{i,c}^{1:n}$ denotes the context obtained from the outputs of preceding agent A_i, while $\mathbf{K}_i^{1:n}$ and $\mathbf{V}_i^{1:n}$ represent the Key-Value (KV) cache across all layers of A_i. Eq. 10 shows that the KV cache generated by agents can serve as output messages for inter-agent collaboration, and the theoretical validity of this equivalence follows directly from Theorem 3.3 established in (Zou et al., 2026).

Built upon the above implicit communication principle in Eq. 10 and alignment strategy in Eq. 8, we propagate complete reasoning trajectories among agents to realize latent collaboration:

$$\hat{\mathbf{s}}_{i,c}^{1:n} = \mathbf{h}_{i,c}^{1:n}\,\mathbf{W}_a, \qquad \mathbf{h}_{i,c}^{1:n} = \begin{cases} \mathrm{A}_i\big(\cdot \,\big|\, \hat{\mathbf{s}}_{i,<c}^{1:n},\, \mathbf{S}^{1:n}\big), & i = 1, \\ \mathrm{A}_i\big(\cdot \,\big|\, \hat{\mathbf{s}}_{i,<c}^{1:n},\, \mathbf{S}^{1:n},\, \mathbf{K}_{i-1}^{1:n},\, \mathbf{V}_{i-1}^{1:n}\big), & 1 < i \le m. \end{cases} \tag{11}$$

Answer Generation From Eq. 11, we obtain the complete superimposed reasoning trajectories and cache them as $\{\mathbf{K}_{m-1}^{1:n}, \mathbf{V}_{m-1}^{1:n}\}$. To disentangle this superposition state, we deploy an independent query disentanglement strategy on the final agent A_m.

Let $\mathcal{B} = \{\mathbf{X}^j\}_{j=1}^{n}$ be a batch of independent queries, and let $\mathbf{Q}^j = \mathbf{W}_Q\mathbf{X}^j$ represent the projection of the j-th query within the attention layer, with $\mathbf{W}_Q$ denotes the query projection matrix. We compute query-aware attention scores by matching $\mathbf{Q}^j$ against the past key matrix $\mathbf{K}_{m-1}^{1:n}$:

$$\alpha^j = \mathrm{softmax}\left(\frac{\mathbf{Q}^j(\mathbf{K}_{m-1}^{1:n})^\top}{\sqrt{d}}\right) \approx \mathrm{softmax}\left(\frac{\mathbf{Q}^j(\mathbf{K}_{m-1}^{j})^\top}{\sqrt{d}}\right), \tag{12}$$

where $1/\sqrt{d}$ denotes the scaling factor, and $\mathrm{softmax}$ refers to the normalized function. This second approximation is valid since each $\mathbf{Q}^j$ focuses on its most relevant key $\mathbf{K}^j$, such that

$\mathbf{Q}^j(\mathbf{K}^{j'})^\top \ll \mathbf{Q}^j(\mathbf{K}^j)^\top$ for all $j' \neq j$. We use the query-aware attention distribution α^j to retrieve target values from the superimposed past value $\mathbf{V}_{m-1}^{1:n}$. This operation effectively disentangles the superimposed reasoning trajectories encoded in past KV cache. Detailed theoretical derivations and error bounds for the disentanglement mechanism are provided in Appendix A.2. The disentanglement of reasoning trajectories from superposition is visualized in Appendix F.1.

Building on the above disentanglement principle, we formulate the answer generation procedure as:

$$\hat{\mathbf{e}}_{m,c}^j \sim \hat{\mathbf{h}}_{m,c}^j = \mathrm{A}_m(\cdot|\hat{\mathbf{e}}_{m,<c}^j, \mathbf{X}^j, \mathbf{K}_{m-1}^{1:n}, \mathbf{V}_{m-1}^{1:n}), \quad 1 < j \leq n, \tag{13}$$

where $\hat{\mathbf{e}}_{m,c}^j$ is the answer for the j-th query, which is obtained from $\hat{\mathbf{h}}_{m,c}^j$ via standard decoding. Furthermore, we adopt single-agent reasoning as an ablation baseline to verify that our disentanglement strategy facilitates effective reuse of historical reasoning trajectories, with corresponding results reported in Table 2.

4.4 Training and Inference

With the explicit outputs $\{\hat{\mathbf{e}}_{m,t}^j\}_{j=1}^n$ in Eq. 13, we extract the final predictions $\{\hat{y}^j\}_{j=1}^n$ for the n queries within a batch. Taking the ground-truth answers $\{y^j\}_{j=1}^n$ from the dataset, we fine-tune the backbone with the loss function below:

$$\begin{aligned} \mathcal{L}_{\text{Total}} &= \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{CSO}}, \\ \text{where:}\quad \mathcal{L}_{\text{CE}} &= -\frac{1}{n}\sum_{j=1}^{n}\sum_{c=1}^{C} y_{j,c}\log(\hat{y}_{j,c}). \end{aligned} \tag{14}$$

Here $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss between predictions and groundtruth answers, and $\mathcal{L}_{\text{CSO}}$ is the continuous supervised fine-tuning loss introduced in Eq. 9. We adopt LoRA (Mangrulkar et al., 2022) for parameter-efficient fine-tuning. The LoRA adapters are injected into the self-attention projection layers: *q_proj*, *k_proj*, *v_proj*, and *o_proj*. The original pretrained weights remain frozen during training, and only the low-rank matrices of these target modules are optimized. Additional training details can be found in Appendix E.

During inference, REMUX first feeds a batch of queries into the superposition reasoning pipeline formulated in Eq. 11 to perform latent collaboration across $m-1$ agents. It then disentangles these superposed reasoning trajectory to produce explicit answers for all queries in the batch, following the procedure in Eq. 13.

Complexity Analysis. Let n, m, c and d denote the number of queries, agents, reasoning steps per agent, and embedding dimension, respectively. The token, time, and space overhead of REMUX are reduced from $\mathcal{O}(nmc)$, $\mathcal{O}(nmcd)$, and $\mathcal{O}(nmcd)$ to $\mathcal{O}(nc + mc - c)$, $\mathcal{O}(2mcd - 2cd + ncd)$, and $\mathcal{O}(mcd - cd + ncd)$, respectively. This indicates that as the number of queries n scales up, REMUX achieves higher efficiency in token, time, and space overhead than the one-to-one methods. A detailed complexity analysis can be found in Appendix B.

5 Evaluation

5.1 Experimental Setup

We briefly describe our experimental setup, which is divided into four parts. (i) **Backbones.** We evaluate REMUX across diverse backbones: Qwen3-4B/8B/14B, Ministral-3-8B, Mistral-Nemo-12B, Moonlight-16B-A3B, and DeepSeek-R1-Distill-Qwen series (8B/14B/32B). (ii) **Baselines.** We compare REMUX against four baselines: *Single-Agent*, a single agent performing text-based reasoning; *Sequential TextMAS* (Zhao et al., 2026), which adopts the chain-of-agents design with text-mediated collaboration; *Hierarchical TextMAS* (Wang et al., 2025), where domain-specialized agents collaborate through a summarizer with text; and *LatentMAS* (Zou et al., 2026), a pure latent MAS that processes each query independently in continuous space. (iii) **Multi-agent Architectures.** To ensure fair comparison, we follow the same multi-agent architectures (*sequential* and *hierarchical*), with detailed descriptions provided in Appendix D.2. (iv) **Benchmarks.** We evaluate REMUX on eight benchmarks spanning three categories: Math & Science Reasoning (GSM8K (Cobbe et al., 2021), AIME25 (Zhang & Math-AI, 2025), GPQA (Rein et al., 2023), and MedQA (Yang et al.,

Tasks	Metrics	Qwen3-4B				Qwen3-8B				Qwen3-14B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	92.7	92.4	92.1	0.6 ↓	92.8	92.1	92.2	0.1 ↓	98.2	98.0	98.1	0.1 ↑
	Token	7205	3251	302	**10.8**×	7326	3146	295	**10.7**×	6981	3139	257	**12.2**×
	Time	20.5	3.0	0.7	**4.3**×	21.4	4.1	0.9	**4.5**×	27.5	6.2	1.2	**5.1**×
ARC-C	Acc.	87.5	90.4	90.3	0.1 ↓	93.2	93.5	92.7	0.8 ↓	95.2	94.7	94.1	0.6 ↓
	Token	7685	3734	365	**10.2**×	7623	3695	361	**10.2**×	7459	3698	337	**11.0**×
	Time	15.7	3.1	0.7	**4.4**×	21.8	6.2	0.9	**6.8**×	26.6	9.3	1.1	**8.5**×
GSM8K	Acc.	83.7	87.9	87.5	0.4 ↓	91.4	92.8	93.0	0.2 ↑	92.5	93.1	92.6	0.5 ↓
	Token	7155	4900	461	**10.6**×	7083	4875	458	**10.6**×	6981	4881	420	**11.6**×
	Time	18.4	6.8	1.1	**6.2**×	20.6	6.9	1.1	**6.3**×	25.7	9.0	1.3	**6.9**×
MedQA	Acc.	64.5	65.3	65.2	0.1 ↓	77.1	77.9	78.0	0.1 ↑	78.0	77.2	77.6	0.4 ↑
	Token	12806	6135	607	**10.1**×	12806	6083	585	**10.4**×	11572	6059	560	**10.8**×
	Time	61.7	9.8	1.5	**6.5**×	46.6	10.1	1.8	**5.6**×	56.3	10.0	2.0	**5.0**×
MBPP+	Acc.	71.3	71.3	71.0	0.3 ↓	72.0	73.0	73.4	0.4 ↑	73.5	73.8	73.4	0.4 ↓
	Token	12714	4377	419	**10.4**×	12174	4392	410	**10.7**×	11769	4306	407	**10.6**×
	Time	26.5	4.4	0.7	**6.3**×	40.1	7.2	1.3	**5.5**×	64.6	8.6	1.5	**5.7**×
HumanEval+	Acc.	78.0	78.0	77.5	0.5 ↓	76.8	76.8	77.5	0.7 ↑	84.8	83.1	83.5	0.4 ↑
	Token	23592	4402	437	**10.1**×	22347	4395	420	**10.5**×	20584	4356	419	**10.4**×
	Time	25.1	10.7	2.9	**3.7**×	33.1	12.8	3.0	**4.3**×	47.0	15.6	3.5	**4.5**×
GPQA	Acc.	39.0	41.5	41.3	0.2 ↓	43.0	45.6	45.7	0.1 ↑	47.8	47.8	48.1	0.3 ↑
	Token	32657	22782	1623	**14.0**×	31429	23165	1590	**14.5**×	30840	21968	1584	**13.9**×
	Time	136.4	28.6	6.5	**4.4**×	157.0	40.1	6.8	**5.9**×	189.3	58.7	7.0	**8.4**×
AIME25	Acc.	46.0	49.0	50.0	1.0 ↑	46.0	50.0	50.7	0.7 ↑	50.0	57.0	57.5	0.5 ↑
	Token	37601	22883	1734	**13.1**×	36486	22176	1659	**13.4**×	34065	22047	1683	**13.1**×
	Time	431.8	56.8	7.1	**8.0**×	447.0	69.7	7.6	**9.2**×	558.3	73.0	8.0	**9.1**×
Average	Acc.	70.3	72.0	71.9	0.1 ↓	74.0	75.2	75.4	0.2 ↑	77.5	78.1	78.1	**0.0**
	Token	17677	9058	744	**12.2**×	17159	8991	722	**12.4**×	16281	8807	708	**12.4**×
	Time	92.0	15.4	2.7	**5.8**×	98.5	19.6	2.9	**6.7**×	124.4	23.8	3.2	**7.4**×

Table 1: Main results of our method on eight datasets under the sequential MAS setting. We report three metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), and average time consumption per query ("Time"). Improvements over SOTA baselines use ratios of (SOTA baselines/Ours). The superposition count n of REMUX is fully reported in Appendix F.7.

2025)), Commonsense Reasoning (ARC-E (Clark et al., 2018) and ARC-C (Clark et al., 2018)), and Code Generation (MBPP+ (Liu et al., 2023) and HumanEval+ (Liu et al., 2023)). We provide the detailed experimental setup, training procedures, and prompt descriptions in Appendix D, Appendix E, and Appendix G, respectively.

5.2 OVERALL PERFORMANCE

We first evaluate REMUX under sequential MAS with three Qwen3 backbones, measuring accuracy, token usage and inference time, with results in Table 1. Across eight tasks, REMUX achieves comparable accuracy to the state-of-the-art LatentMAS (Zou et al., 2026), slashing token consumption by 10.1–$14.5\times$ and accelerating inference by 4.3–$9.2\times$ without compromising task accuracy. Unlike prior one-to-one paradigms, these pronounced efficiency gains stem from our ability to superpose multiple samples within a single forward pass. To evaluate generality, we further extend our experiments to hierarchical MAS and diverse backbones (detailed in Appendix F.7), where REMUX yields 10.3–$16.3\times$ gains in token efficiency and 1.9–$10.0\times$ inference speedups. Notably, our method is also compatible with Mixture of Experts (MoE) backbones, since expert activation only replaces the feed-forward network modules rather than the attention mechanism, which underpins our superposition. Accordingly, we validate REMUX on the MoE backbone Moonlight-16B-A3B, with results provided in Appendix F.7.

5.3 IN-DEPTH ANALYSES ON REMUX

Impact of Superposition Count. To further evaluate the impact of the superposition count (n) on both accuracy and token consumption, we vary its value within each batch. As shown in Fig. 3(a), we observe that task accuracy degrades for all datasets as n grows excessively. This is because superimposed representations induce substantial interference in the superposition space, and we provide a theoretical error bound in Appendix A.3. Meanwhile, we also find that increasing n

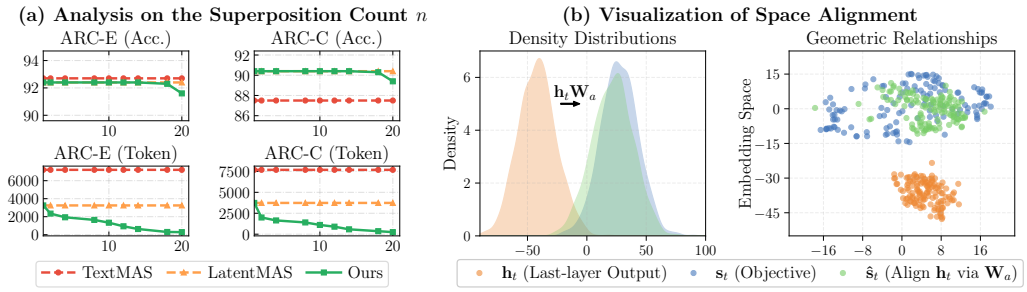


Figure 3: (a) Accuracy and token consumption on ARC-E and ARC-C as the superposition count n, shown on the x-axis, increases. (b) Visualization of space alignment effects in density distributions and geometric relationships.

within a batch reduces token consumption, because the reasoning trajectories of all queries share the same latent sequence. Overall, there exists a trade-off between accuracy and token consumption. In Appendix C, we present an efficient algorithm for determining the value of n.

Ablation Studies. To evaluate the effectiveness of the proposed components in REMUX, we design five ablation variants by removing each component, and report the results of the first two datasets in Table 2. Results for the remaining datasets can be found in Appendix F.2. We find that our proposed mechanisms, including the mapping ϕ, superimposed KV-Cache communication, interference mitigation strategy $\mathbf{W}_a$&$\mathcal{L}_{CSO}$ and model fine-tune $\mathcal{L}_{Total}$, all contribute to the overall model performance. In particular, comparisons against *Single-Agent* verify that the superimposed reasoning paradigm of REMUX preserves the inherent advantages of collaborative intelligence in multi-agent systems. In Appendix F.3, we present single-agent comparisons across more diverse backbones.

Variants	Datasets					
	ARC-E			ARC-C		
	Acc.	Token	Time	Acc.	Token	Time
REMUX	92.1	302	0.7	90.3	365	0.7
Single-Agent	83.5	156	0.2	78.0	289	0.2
w/o ϕ	86.6	302	0.7	82.4	365	0.7
w/o KV-comm.	84.0	481	0.9	78.3	394	0.8
w/o $\mathbf{W}_a$&$\mathcal{L}_{CSO}$	88.9	302	0.7	85.6	365	0.7
w/o $\mathcal{L}_{Total}$	74.8	302	0.7	62.0	365	0.7

Table 2: Ablation studies on REMUX.

Effectiveness on Space Alignment. To evaluate the effectiveness of space alignment in REMUX, we visualize the density distributions and geometric relationships of the superimposed representations. As shown in Fig. 3, the aligned representation $\hat{\mathbf{s}}_t$ exhibits consistency with the supervision signal $\mathbf{s}_t$ after the $\mathbf{W}_a$ transformation. This consistency indicates that the superimposed reasoning trajectories capture the reasoning of all n queries. More results and detailed discussions are provided in Appendix F.6.

Visualization of Full Reasoning Traces. We perform empirical evaluations with Qwen3-14B under both MAS architectures (sequential and hierarchical), visualizing the complete trajectories of collaborative reasoning in the embedding space. The results are provided in Appendix F.1. We observe that the superposed reasoning trajectories are effectively disentangled into their respective trajectories at the final agent. This is because the introduced individual queries encourage the attention to focus on the specific problems. In addition, a parameter analysis of reasoning steps and the associated discussion are provided in Appendix F.4.

6 CONCLUSION

In this paper, we propose REMUX, a novel *token-efficient* collaborative reasoning framework designed to reduce token consumption in multi-agent systems via superposition. Extensive experiments demonstrate that REMUX reduces token consumption by a factor of 10.1–16.3 and achieves a 1.9–10× speedup compared to state-of-the-art baselines, while retaining competitive task accuracy. In future work, we plan to extend REMUX to dynamic and heterogeneous multi-agent scenarios and explore its applicability to more complex tasks to further improve scalability and generalization.

AI USE STATEMENT

In this work, we used generative AI tools to polish proof writing, refine the wording of hypotheses, and assist with translation. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, and interpreting results. Generating synthetic data sets, and cleaning and reformatting datasets are not applicable to this work. We have manually inspected and reconfirmed that the LLM-based work contains no fabrication or plagiarism. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Lisa Alazraki, Lihu Chen, Ana Brassard, Joe Stacey, Hossein A. Rahmani, and Marek Rei. AgentCoMa: A compositional benchmark mixing commonsense and mathematical reasoning in real-world scenarios. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8383–8410, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.380. URL https://aclanthology.org/2026.acl-long.380/.

Yulang Chen, Haoxuan Peng, Jinyan Liu, Zichen Wen, Dongrui Liu, and Linfeng Zhang. AgentSlimming: Towards efficient and cost-aware multi-agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 30064–30086, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1387. URL https://aclanthology.org/2026.acl-long.1387/.

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, 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, 2021. URL https://arxiv.org/abs/2110.14168.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129, San Diego, California, United States, July 2026. Association for Computational Linguistics. URL https://aclanthology.org/2026.acl-long.1248/.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

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*, 2026. URL https://iclr.cc/virtual/2026/poster/10010020.

Ali E. Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Dmytro Shved, Gavin J. Gyimesi, Jon M. Laurent, Samantha M. Wright, Muhammed T. Razzak, Andrew D. White, Silvia C. Finnemann, Michaela M. Hinks, and Samuel G. Rodriques. A multi-agent system for automating scientific discovery. *Nature*, 655(8122):497–505, Jul 2026. ISSN 1476-4687. doi: 10.1038/s41586-026-10652-y. URL https://doi.org/10.1038/s41586-026-10652-y.

Tieliang Gong, Yuxin Dong, Shujian Yu, and Bo Dong. Computationally efficient approximations for matrix-based rényi's entropy. *IEEE Transactions on Signal Processing*, 70:6170–6184, 2022. doi: 10.1109/TSP.2022.3233724.

Alperen Gozeten, Muhammed Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 82881–82914, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/85ec26ef94c3acb4c195e905df1ff4f7-Paper-Conference.pdf.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Workshop on Reasoning and Planning for Large Language Models, ICLR2025*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://iclr.cc/virtual/2022/poster/6319.

Yiping Li, Zhiyu An, and Wan Du. When less latent leads to better relay: Information-preserving compression for latent multi-agent llm collaboration, 2026. URL https://arxiv.org/abs/2604.13349.

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, 2023. URL https://arxiv.org/abs/2305.01210.

Yingzhuo Liu. Beyond tokens: a unified framework for latent communication in llm-based multi-agent systems, 2026. URL https://arxiv.org/abs/2606.05711.

Yizhou Liu, Ziming Liu, and Jeff Gore. Superposition yields robust neural scaling. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 159269–159305. Curran Associates, Inc., 2025. doi: 10.52202/085713-5320. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/e97ac22927560eb2de6b658498cbc575-Paper-Conference.pdf.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 39553–39565. Curran Associates, Inc., 2023. doi: 10.52202/075280-1718.

Vishvak Murahari, Carlos Jimenez, Runzhe Yang, and Karthik Narasimhan. Datamux: Data multiplexing for neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17515–17527. Curran Associates, Inc., 2022. doi: 10.52202/068431-1273. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6fc46679a7ba2ec82183cf01b80e5133-Paper-Conference.pdf.

Vishvak Murahari, Ameet Deshpande, Carlos Jimenez, Izhak Shafran, Mingqiu Wang, Yuan Cao, and Karthik Narasimhan. MUX-PLMs: Data multiplexing for high-throughput language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4540–4554, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.301. URL https://aclanthology.org/2023.findings-emnlp.301/.

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.

Michael Rizvi-Martel, Satwik Bhattamishra, Neil Rathi, Guillaume Rabusseau, and Michael Hahn. Benefits and limitations of communication in multi-agent reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 144216–144256, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e9372abc370b8302459abdce4bdce5a5-Paper-Conference.pdf.

Yixin Song, Guangchi Liu, Xiangyu Xu, Shaofeng Li, Zhen Ling, Yiwei Wang, and Yujun Cai. MIMO-LP: A Multi-Input Multi-Output Framework for Subgraph-based Link Prediction. In *Forty-third International Conference on Machine Learning*, 2026.

Rémy Sun, Alexandre Ramé, Clément Masson, Nicolas Thome, and Matthieu Cord. Towards efficient feature sharing in MIMO architectures. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2696–2700, 2022. doi: 10.1109/CVPRW56347.2022.00303.

Geng Tu, Dingming Li, Jun Huang, and Ruifeng Xu. Consensus-Driven Multi-Agent Cognitive Reasoning for Enhancing the Emotional Intelligence of Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17751–17759, 2026. doi: https://doi.org/10.1609/aaai.v40i21.38832.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 2026. URL https://webapps.math.uci.edu/~rvershyn/papers/HDP-book/HDP-2.pdf.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, pp. 33944–33963, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5434be94e82c54327bb9dcaf7fca52b6-Paper-Conference.pdf.

Xiaoyu Xiong, Yuqi Ren, and Deyi Xiong. EvoSci: A Bio-Inspired Multi-Agent Framework for the Evolution of Scientific Discovery. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9846–9878, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.447. URL https://aclanthology.org/2026.acl-long.447/.

Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. Llm-medqa: Enhancing medical question answering through case studies in large language models, 2025. URL https://arxiv.org/abs/2501.05464.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bbc461518c59a2a8d64e70e2c38c4a0e-Paper-Conference.pdf.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 168990–169012. Curran Associates, Inc., 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f7396d1c54d51416958d63e285377103-Paper-Conference.pdf.

Jiaxing Zhao, Hongbin Xie, Yuzhen Lei, Xuan Song, Zhuoran Shi, Lianxin Li, Shuangxue Liu, Linguo Xie, and Haoran Zhang. Cochain: Balancing insufficient and excessive collaboration in llm agent workflows, 2026. URL https://arxiv.org/abs/2505.10936.

Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. ReSo: A reward-driven self-organizing LLM-based multi-agent system for reasoning tasks. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15979–15998, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.808. URL https://aclanthology.org/2025.emnlp-main.808/.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 79931–79963. Curran Associates, Inc., 2025. doi: 10.52202/085713-2670. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/72c363c2a573ca2128bd176d3317696b-Paper-Conference.pdf.

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. In *The 43rd International Conference on Machine Learning (ICML26)*, 2026. URL https://icml.cc/virtual/2026/poster/61180.

A Theoretical Analysis

This section provides a formal theoretical justification for REMUX. We begin by introducing the *continuous superposition space* (Definition A.1), in which token embeddings from different queries are projected into mutually near-orthogonal subspaces, so that cross-query interference vanishes (Theorem A.1). Building on this near-orthogonality, we then establish why superposed reasoning is both computable and recoverable. We prove that the Transformer-based backbone operating on the superimposed input approximates the average of the per-query computations (Theorem A.2), which implies that the joint superimposed forward pass is approximately equivalent to computing each query in isolation and averaging the outputs, so that a single shared latent sequence faithfully carries every per-query reasoning trajectory. We further prove that the attention mechanism can disentangle the independent reasoning trajectory corresponding to each query from the superposition states (Theorem A.3). Finally, we derive a theoretical generalization bound for REMUX and analyze how the total approximation error accumulates as more queries are superposed (Appendix A.3).

A.1 Proof for Near-Orthogonality

The standard pipeline operates in a discrete semantic space, which makes it impossible to simultaneously represent multiple semantic concepts within one input embedding. To address this limitation, REMUX introduces a continuous superposition space, which is defined as follows.

Definition A.1 (Continuous Superposition Space)**.** Let $\mathcal{E}_{I,\cdot} = \{\mathbf{e}_i\}_{i\in I} \subset \mathbb{R}^d$ denote a subset of token embeddings from the vocabulary embedding matrix $\mathbf{E}$. Given a collection of linear transformations $\{\phi^j\}_{j=1}^n$, we define the *continuous superposition space* $\mathcal{S}$ as the set of uniform weighted sums over the transformed embeddings:

$$\mathcal{S} = \left\{ \sum_{j=1}^{n} \frac{1}{n} \phi^j\big(\mathbf{e}^j\big) \;\middle|\; \mathbf{e}^j \in \mathcal{E}_{I,\cdot} \right\} \subset \mathbb{R}^d, \tag{15}$$

where each mapping $\phi^j(\cdot)$ is the Hadamard product with a random vector $\mathbf{g}^j \in \mathbb{R}^d$, i.e., $\phi^j(\mathbf{e}) = \mathbf{g}^j \odot \mathbf{e}$. The entries of each $\mathbf{g}^j$ are sampled independently from the standard normal distribution $\mathcal{N}(0,1)$ once and for all, and remain fixed throughout training and inference.

Theorem A.1 (Near-orthogonality)**.** *Let $\{\mathcal{X}^j\}_{j=1}^n$ be a set of queries, and let $\mathcal{V}^j = \mathrm{span}\big(\{\phi^j(\mathbf{e}) \mid \mathbf{e} \in \mathcal{X}^j\}\big)$ denote the subspace onto which each query $\mathcal{X}^j$ is projected in the continuous superposition space. Then $\mathcal{V}^{j'}$ is nearly orthogonal to $\mathcal{V}^j$ for all $j' \neq j$. Formally, for any $\delta \in (0,1)$, with*

probability at least $1-\delta$ *over the one-time draw of* $\{\mathbf{g}^j\}$*:*

$$\frac{\left|\langle \phi^j(\mathbf{e}),\, \phi^{j'}(\mathbf{e}')\rangle\right|}{\|\phi^j(\mathbf{e})\|\,\|\phi^{j'}(\mathbf{e}')\|} \;\le\; \varepsilon, \qquad \varepsilon = O\left(\mu_{\max}\sqrt{\frac{\log(nL/\delta)}{d}}\right), \tag{16}$$

holds for all $\mathbf{e}\in\mathcal{X}^j$ *and* $\mathbf{e}'\in\mathcal{X}^{j'}$*, where* L *denotes the number of tokens per query and* $\mu_{\max}$ *denotes the maximal coherence of the token embeddings. In particular, since the embedding dimension* d *of modern LLMs is large and the maximal coherence* $\mu_{\max}$ *is a small constant independent of* d*, the error* ε *decays at the rate* $O\Big(\mu_{\max}\sqrt{\log(nL/\delta)/d}\Big)$ *and vanishes as* d *grows.*

Proof. Let $\phi^j(\mathbf{e}^j) = \mathbf{g}^j \odot \mathbf{e}^j$ as in Definition A.1. The token embeddings are deterministic, and the only randomness comes from the one-time draw of $\{\mathbf{g}^j\}_{j=1}^n$. The proof consists of two parts. We first show that the cross-query inner product vanishes in expectation, and then bound the deviation of any single realization from this expectation.

For two transformed embeddings from different queries $j \neq j'$, define their inner product as:

$$Z \;=\; \langle \mathbf{g}^j \odot \mathbf{e}^j,\, \mathbf{g}^{j'} \odot \mathbf{e}^{j'}\rangle \;=\; \sum_{k=1}^{d} g_k^j\, g_k^{j'}\, e_k^j\, e_k^{j'}. \tag{17}$$

Since $\mathbf{g}^j$ and $\mathbf{g}^{j'}$ are independent zero-mean Gaussian vectors, we have:

$$\mathbb{E}[Z] \;=\; \sum_{k=1}^{d} e_k^j\, e_k^{j'}\, \mathbb{E}\big[g_k^j\big]\, \mathbb{E}\big[g_k^{j'}\big] \;=\; 0, \tag{18}$$

therefore the cross-query correlation vanishes in expectation.

We next bound the deviation of a single realization. Since the target is a cosine in Eq. 16, we control its numerator and denominator separately. For the numerator, the idea is to separate the randomness from the deterministic embeddings. Writing Z as the bilinear form $Z = (\mathbf{g}^j)^\top \mathbf{A}\, \mathbf{g}^{j'}$ with $\mathbf{A} = \operatorname{diag}\big(e_k^j e_k^{j'}\big)_{k=1}^d$ collects all randomness into the two independent Gaussian vectors and all embedding dependence into $\mathbf{A}$. The Hanson–Wright inequality (Vershynin, 2026) controls the tail of such a bilinear form through the norms of $\mathbf{A}$, which for this diagonal matrix reduce to $\|\mathbf{A}\|_F \le \|\mathbf{e}^j\|_\infty \|\mathbf{e}^{j'}\|$ and $\|\mathbf{A}\| \le \|\mathbf{e}^j\|_\infty \|\mathbf{e}^{j'}\|_\infty$. Hence, with probability at least $1-\delta$:

$$|Z| \;\lesssim\; \|\mathbf{e}^j\|_\infty \|\mathbf{e}^{j'}\| \sqrt{\log(1/\delta)} \;+\; \|\mathbf{e}^j\|_\infty \|\mathbf{e}^{j'}\|_\infty \log(1/\delta), \tag{19}$$

where $\lesssim$ hides universal constants. This bounds the numerator of the cosine entirely by deterministic embedding quantities.

For the denominator, we need to show that the random projection does not shrink the embeddings, so that the projected norms stay comparable to the original ones. The squared projected norm $\|\mathbf{g}^j \odot \mathbf{e}^j\|^2 = \sum_k (g_k^j)^2 (e_k^j)^2$ is again a quadratic form in $\mathbf{g}^j$ with mean $\|\mathbf{e}^j\|^2$, so the same inequality bounds its fluctuation by $O\big(\|\mathbf{e}^j\|_\infty \|\mathbf{e}^j\| \sqrt{\log(1/\delta)}\big)$ up to a lower-order term. In terms of the maximal coherence $\mu_{\max} = \max_{j,\mathbf{e}} \sqrt{d}\, \|\mathbf{e}\|_\infty / \|\mathbf{e}\|$, this fluctuation is at most half of the mean whenever $\mu_{\max}^2 \log(1/\delta) \lesssim d$, giving $\|\mathbf{g}^j \odot \mathbf{e}^j\| \ge \|\mathbf{e}^j\|/\sqrt{2}$ and similarly for $\mathbf{e}^{j'}$.

Combining the two bounds, we divide the numerator bound by these norm lower bounds and use $\|\mathbf{e}\|_\infty \le \mu_{\max} \|\mathbf{e}\|/\sqrt{d}$ to obtain:

$$\frac{|Z|}{\|\mathbf{g}^j \odot \mathbf{e}^j\|\, \|\mathbf{g}^{j'} \odot \mathbf{e}^{j'}\|} \;\lesssim\; \mu_{\max} \sqrt{\tfrac{\log(1/\delta)}{d}} + \mu_{\max}^2\, \tfrac{\log(1/\delta)}{d}, \tag{20}$$

where the second term is of lower order whenever $\mu_{\max}^2 \log(1/\delta) \lesssim d$. Hence, with probability at least $1-\delta$, the cosine form of the cross-query inner product satisfies:

$$\frac{\left|\langle \mathbf{g}^j \odot \mathbf{e}^j,\, \mathbf{g}^{j'} \odot \mathbf{e}^{j'}\rangle\right|}{\left\|\mathbf{g}^j \odot \mathbf{e}^j\right\|\, \left\|\mathbf{g}^{j'} \odot \mathbf{e}^{j'}\right\|} \;\lesssim\; \mu_{\max} \sqrt{\tfrac{\log(1/\delta)}{d}}, \tag{21}$$

so a single realization deviates from exact orthogonality by at most this amount per pair.

Finally, we extend the bound to all pairs. Applying the above with failure probability on the order of $\delta/(n^2L^2)$ per pair and taking a union bound over the at most n^2L^2 pairs of transformed embeddings and the nL norm-concentration events, we obtain that, under the regime condition $\mu_{\max}^2 \log(nL/\delta) \lesssim d$, with probability at least $1-\delta$:

$$\frac{\left|\langle \mathbf{g}^j \odot \mathbf{e}^j,\, \mathbf{g}^{j'} \odot \mathbf{e}^{j'} \rangle\right|}{\left\|\mathbf{g}^j \odot \mathbf{e}^j\right\| \left\|\mathbf{g}^{j'} \odot \mathbf{e}^{j'}\right\|} \;\le\; \varepsilon, \qquad \varepsilon = O\left(\mu_{\max}\sqrt{\frac{\log(nL/\delta)}{d}}\right), \tag{22}$$

simultaneously for all pairs.

Hence the cross-query inner product need not be exactly zero in a single realization, but its cosine form decays as $O\Big(\mu_{\max}\sqrt{\log(nL/\delta)/d}\Big)$. The error is of order $O(\mu_{\max}/\sqrt{d})$ up to logarithmic factors and vanishes as the dimension d grows.

□

A.2 Why ReMux Works

This section provides a theoretical justification for the two core operations of ReMux, namely computing multiple queries in superposition and disentangling query-specific outputs from the superposed cache. We first show in Theorem A.2 that a batch of queries, once projected into mutually decorrelated subspaces by a fixed random projection, can be processed jointly by the Transformer backbone in a single forward pass, so that the superposed output approximates the average of the per query outputs. The approximation error of this step comes only from the nonlinearities of the softmax δ_{attn} and the feed-forward activations δ_{ffn}.

Theorem A.2 (Computation in Superposition)**.** *Let $\{\mathbf{X}^j\}_{j=1}^n$ denote a batch of input queries, where each $\mathbf{X}^j = \{\mathbf{x}_t^j\}_{t=1}^L$ consists of L tokens, and let $\mathrm{F}(\cdot)$ denote the Transformer-based backbone. In the superposition pipeline, we construct the superimposed input $\mathbf{S}^{1:n} = \frac{1}{n}\sum_{j=1}^n \phi^j(\mathbf{X}^j)$, where ϕ^j is applied token-wise, and feed it into $\mathrm{F}(\cdot)$ to perform joint superposition computation. Then the following approximation holds for the superposition pipeline:*

$$\mathrm{F}\big(\mathbf{S}^{1:n}\big) \approx \frac{1}{n}\sum_{j=1}^n \mathrm{F}\big(\phi^j(\mathbf{X}^j)\big). \tag{23}$$

Proof. Let $\mathbf{X}^j = \{\mathbf{x}_t^j\}_{t=1}^L$ denote the j-th query and let $\mathbf{u}_t^j := \phi^j(\mathbf{x}_t^j) = \mathbf{g}^j \odot \mathbf{x}_t^j$ be the transformed token. The superimposed input is the sequence $\mathbf{S}^{1:n} = \{\mathbf{S}_t^{1:n}\}_{t=1}^L$ with $\mathbf{S}_t^{1:n} = \frac{1}{n}\sum_{j=1}^n \mathbf{u}_t^j$. By Definition A.1, the vectors $\mathbf{g}^j$ are independent and zero-mean, so the transformed tokens of different queries are decorrelated in expectation:

$$\mathbb{E}\big[\langle \mathbf{u}_t^j,\, \mathbf{u}_{t'}^{j'} \rangle\big] = \sum_{k=1}^d x_{t,k}^j\, x_{t',k}^{j'}\, \mathbb{E}\big[g_k^j\big]\, \mathbb{E}\big[g_k^{j'}\big] = 0, \qquad \forall\, j \neq j', \tag{24}$$

and Theorem A.1 further shows that the subspaces $\{\mathcal{V}^j\}_{j=1}^n$ are nearly orthogonal with high probability.

Consider a single self-attention head i with parameter matrices $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$. By linearity, the value projection of the superimposed token decomposes exactly as:

$$\mathbf{W}_i^V \mathbf{S}_t^{1:n} = \mathbf{W}_i^V \left(\frac{1}{n}\sum_{j=1}^n \mathbf{u}_t^j\right) = \frac{1}{n}\sum_{j=1}^n \mathbf{W}_i^V \mathbf{u}_t^j. \tag{25}$$

Moreover, the per-query components remain decorrelated after the value projection. Indeed, for any fixed matrix $\mathbf{W}_i^V$, $\langle \mathbf{W}_i^V \mathbf{u}_t^j,\, \mathbf{W}_i^V \mathbf{u}_{t'}^{j'} \rangle = (\mathbf{u}_t^j)^\top (\mathbf{W}_i^V)^\top \mathbf{W}_i^V\, \mathbf{u}_{t'}^{j'}$ is a bilinear form in the independent zero-mean vectors $\mathbf{g}^j$ and $\mathbf{g}^{j'}$, so its expectation vanishes by the same argument as above in the proof of Theorem A.1.

We next turn to the query-key inner products. Expanding the superimposed score at the position pair (t, t') gives self terms and cross terms:

$$\big(\mathbf{W}_i^K \mathbf{S}_{t'}^{1:n}\big)^\top \mathbf{W}_i^Q \mathbf{S}_t^{1:n} = \sum_{j=1}^{n} \tau_{i,(t,t')}^{(j)} + \frac{1}{n^2} \sum_{j \neq j'} \big(\mathbf{W}_i^K \mathbf{u}_{t'}^{j}\big)^\top \mathbf{W}_i^Q \mathbf{u}_t^{j'}, \tag{26}$$

where $\tau_{i,(t,t')}^{(j)} := \frac{1}{n^2}\big(\mathbf{W}_i^K \mathbf{u}_{t'}^{j}\big)^\top \mathbf{W}_i^Q \mathbf{u}_t^{j}$. Each cross term is again a bilinear form in $\mathbf{g}^j$ and $\mathbf{g}^{j'}$ with $j \neq j'$, hence has zero expectation and concentrates at scale $O(1/\sqrt{d})$. Neglecting the cross terms yields:

$$\big(\mathbf{W}_i^K \mathbf{S}_{t'}^{1:n}\big)^\top \mathbf{W}_i^Q \mathbf{S}_t^{1:n} \;\approx\; \sum_{j=1}^{n} \tau_{i,(t,t')}^{(j)}, \tag{27}$$

and their fluctuations are absorbed into the error δ_{attn} in Appendix A.3.

Therefore, the output of the i-th attention head at position t is:

$$\begin{aligned} \mathrm{head}_i\big(\mathbf{S}^{1:n}\big)_t &= \sum_{t'} \alpha_{t,t'}\, \mathbf{W}_i^V \mathbf{S}_{t'}^{1:n} \;\approx\; \frac{1}{n} \sum_{j=1}^{n} \sum_{t'} \alpha_{t,t'}\, \mathbf{W}_i^V \mathbf{u}_{t'}^{j}, \\ \alpha_{t,t'} &= \frac{\exp\Big(\frac{1}{\sqrt{d}} \sum_{j=1}^{n} \tau_{i,(t,t')}^{(j)}\Big)}{\sum_{t''} \exp\Big(\frac{1}{\sqrt{d}} \sum_{j=1}^{n} \tau_{i,(t,t'')}^{(j)}\Big)}, \end{aligned} \tag{28}$$

where t' ranges over the prefix up to position t under causal masking, and the approximation uses the value decomposition together with the approximate score decomposition. We further approximate each superposed softmax weight $\alpha_{t,t'}$ by its per-query counterpart, namely the weight computed from the logits $\big\{\frac{1}{\sqrt{d}}(\mathbf{W}_i^K \mathbf{u}_{t'}^{j})^\top \mathbf{W}_i^Q \mathbf{u}_t^{j}\big\}_{t'}$ for query j alone. Note that the superposed logits carry an additional $1/n^2$ scaling relative to the per-query logits, which distorts the softmax temperature. Together with the neglected cross terms, the error of this step is quantified by δ_{attn} in Appendix A.3. Under this approximation, the i-th head output decomposes across queries as:

$$\mathrm{head}_i\big(\mathbf{S}^{1:n}\big)_t \;\approx\; \frac{1}{n} \sum_{j=1}^{n} \mathrm{head}_i\big(\phi^j(\mathbf{X}^j)\big)_t. \tag{29}$$

The output projection $\mathbf{W}^O$ is linear and thus preserves this decomposition. The same holds for the linear stages of the feed-forward layers and for the residual connections, whereas the element-wise activations and the normalization layers introduce perturbations that break exact superposition. we absorb these perturbations into the approximation error δ_{ffn} in Appendix A.3. Stacking all Transformer layers, where the per-layer errors accumulate and are bounded in Appendix A.3, gives:

$$\mathrm{F}\big(\mathbf{S}^{1:n}\big) = \mathrm{F}\left(\frac{1}{n} \sum_{j=1}^{n} \phi^j(\mathbf{X}^j)\right) \;\approx\; \frac{1}{n} \sum_{j=1}^{n} \mathrm{F}\big(\phi^j(\mathbf{X}^j)\big). \tag{30}$$

□

We then show in Theorem A.3 that each query can retrieve its own reasoning trajectory from the superposed KV cache through attention. The key observation is that the random projections suppress cross query matching while preserving the semantic matching between a query and its own trajectory, and that the values of different queries lie in nearly orthogonal subspaces.

Theorem A.3 (Disentanglement under Superposition)**.** *Let $\{\mathbf{x}_t^j\}_{t=1}^{L}$ denote the token embeddings of query j, and let $\mathbf{u}_t^j := \mathbf{g}^j \odot \mathbf{x}_t^j$ denote the transformed tokens, where $\mathbf{g}^1, \ldots, \mathbf{g}^n$ are independent zero-mean Gaussian vectors. For one attention layer with hidden dimension d, the per query components are:*

$$\mathbf{Q}_t^j = \tfrac{1}{n}\, \mathbf{W}_Q\, \mathbf{u}_t^j, \qquad \mathbf{K}_t^j = \tfrac{1}{n}\, \mathbf{W}_K\, \mathbf{u}_t^j, \qquad \mathbf{V}_t^j = \tfrac{1}{n}\, \mathbf{W}_V\, \mathbf{u}_t^j, \tag{31}$$

and the superposed KV cache is formed by elementwise summation of the position aligned trajectories, $\mathbf{K}^{1:n} = \sum_{j=1}^{n} \mathbf{K}^j$ *and* $\mathbf{V}^{1:n} = \sum_{j=1}^{n} \mathbf{V}^j$*. Let* $\mathcal{V}^j = \mathrm{span}\{\mathbf{V}_t^j\}_{t=1}^{L}$ *denote the value subspace of query* j *with orthogonal projector* $\mathbf{P}^j$*.*

Then, with high probability over the draw of $\{\mathbf{g}^j\}$*, the attention output of each query over the superposed cache recovers its native per query output on its own value subspace:*

$$\left\|\mathbf{P}^j \,\mathrm{Attn}(\mathbf{Q}_t^j, \mathbf{K}^{1:n}, \mathbf{V}^{1:n}) - \mathrm{Attn}(\mathbf{Q}_t^j, \mathbf{K}^j, \mathbf{V}^j)\right\| \;\le\; \varepsilon, \tag{32}$$

for every query j *and every position* t*. Here* $\mathrm{Attn}(\mathbf{Q}_t^j, \mathbf{K}^j, \mathbf{V}^j)$ *is the output computed from the* j*-th components alone, and* ε *denotes the disentanglement error, which vanishes as* $1/\sqrt{d}$ *up to logarithmic factors.*

Proof. We split the proof into two stages. In Stage 1, we establish two properties that make disentanglement possible, both enforced by the random projections with high probability. The first property states that each query matches only the keys of its own trajectory and does not match the keys of the other trajectories. The second property states that the values of different trajectories lie in nearly orthogonal subspaces. In Stage 2, we combine the two properties and show that the attention output of each query over the superposed cache recovers its native per-query output on its own value subspace.

Stage 1. For the first property, fix a query j, a position t, and any other query $j' \neq j$ with position p. Expanding the cross inner product gives:

$$Q_t^j (K_p^{j'})^\top = \frac{1}{n^2} \left(\mathbf{g}^j \odot \mathbf{x}_t^j\right)^\top \mathbf{W}_Q^\top \mathbf{W}_K \left(\mathbf{g}^{j'} \odot \mathbf{x}_p^{j'}\right), \tag{33}$$

which is a bilinear form in the two independent zero-mean Gaussian vectors $\mathbf{g}^j$ and $\mathbf{g}^{j'}$. Standard concentration for Gaussian bilinear forms (Vershynin, 2026) shows that such forms are zero in expectation and concentrate at the scale $\widetilde{O}(1/\sqrt{d})$ under the same coherence condition as Theorem A.1. Hence, with high probability over the draw of $\{\mathbf{g}^j\}$:

$$\left|Q_t^j (K_p^{j'})^\top\right| \;\le\; \varepsilon_{\mathrm{pair}} \;=\; \widetilde{O}(1/\sqrt{d}) \qquad \text{for all } j' \neq j \text{ and all } t, p, \tag{34}$$

where $\varepsilon_{\mathrm{pair}}$ denotes the wish-pair error defined in Theorem A.1. In contrast, the inner product between the query and the keys of its own trajectory involves the same vector $\mathbf{g}^j$ on both sides. Therefore, it is a quadratic, rather than bilinear, form in $\mathbf{g}^j$, whose expectation $\frac{1}{n^2}\sum_{k=1}^{d} x_{t,k}^j \left(\mathbf{W}_Q^\top \mathbf{W}_K\right)_{kk} x_{p,k}^j$ does not vanish as d grows. This nonvanishing self matching is precisely the signal through which query j retrieves its own trajectory from the superposed cache of every preceding agent, while the bound above shows that the random projections suppress all cross query matching.

For the second property, fix $j \neq j'$ and positions t, p. The inner product $(V_t^j)^\top V_p^{j'}$ is again a bilinear form in the independent Gaussian vectors $\mathbf{g}^j$ and $\mathbf{g}^{j'}$, so the same concentration argument gives $|(V_t^j)^\top V_p^{j'}| \le \varepsilon_{\mathrm{pair}} \|V_t^j\| \, \|V_p^{j'}\|$. Assuming the value spans $\{\mathbf{V}_t^j\}_{t=1}^{L}$ are well-conditioned, so that the pairwise control extends to all unit vectors in $\mathcal{V}^j$, lifting to the subspace yields:

$$\|P^j V_p^{j'}\| \;\le\; \varepsilon_v \|V_p^{j'}\|, \qquad \varepsilon_v = O\big(\sqrt{r^j}\,\varepsilon_{\mathrm{pair}}\big), \tag{35}$$

where $r^j = \dim \mathcal{V}^j \le L$. In summary, $\varepsilon_{\mathrm{pair}}$ controls how much the query of one trajectory leaks into the keys of the others, and ε_v controls how much the values of the other trajectories leak into the value subspace of query j. Both quantities vanish as $1/\sqrt{d}$ up to a $\sqrt{r^j}$ factor and logarithmic factors.

Stage 2. We now condition on the two properties above and proceed deterministically. Fix a query j and a position t. The attention logits over the superposed cache decompose as follows:

$$\ell_p \;=\; \frac{Q_t^j (K_p^{1:n})^\top}{\sqrt{d}} \;=\; \underbrace{\frac{Q_t^j (K_p^j)^\top}{\sqrt{d}}}_{s_p} + \underbrace{\sum_{j' \neq j} \frac{Q_t^j (K_p^{j'})^\top}{\sqrt{d}}}_{\delta_p}, \tag{36}$$

where s_p is the native semantic score of query j on its own trajectory and δ_p collects the cross query leakage, which satisfies $|\delta_p| \le (n-1)\,\varepsilon_{\text{pair}}/\sqrt{d}$ by the first property. We denote this leakage scale by ρ. Let α be the native attention weights of query j over its own keys, defined by the logits s_p, and let $\tilde{\alpha}$ be the weights over the superposed cache, defined by the logits ℓ_p. Since the softmax map is 2-Lipschitz from ℓ_∞ to ℓ_1, the mean value theorem gives $\|\tilde{\alpha} - \alpha\|_1 \le 2\rho$.

The attention output over the superposed cache decomposes as follows:

$$\mathrm{Attn}(Q_t^j, K^{1:n}, V^{1:n}) \;=\; \sum_p \tilde{\alpha}_p V_p^{1:n} \;=\; \underbrace{\sum_p \alpha_p V_p^j}_{\text{native output}} \;+\; \underbrace{\sum_p (\tilde{\alpha}_p - \alpha_p)\, V_p^j}_{E_1} \;+\; \underbrace{\sum_p \tilde{\alpha}_p \sum_{j' \neq j} V_p^{j'}}_{E_2}, \tag{37}$$

where E_1 is the weight perturbation on the query's own values and E_2 collects the values of the other trajectories. Writing $V_{\max} = \max_{j,p} \|V_p^j\|$, the weight bound gives $\|E_1\| \le 2\rho\, V_{\max}$, and the second property gives $\|P^j E_2\| \le (n-1)\,\varepsilon_v\, V_{\max}$. Since $\sum_p \alpha_p V_p^j = \mathrm{Attn}(Q_t^j, K^j, V^j)$ already lies in $\mathcal{V}^j$, the triangle inequality yields:

$$\Big\| P^j\, \mathrm{Attn}(Q_t^j, K^{1:n}, V^{1:n}) \;-\; \mathrm{Attn}(Q_t^j, K^j, V^j) \Big\| \;\le\; \varepsilon = (n-1)\, V_{\max}\Big(\frac{2\varepsilon_{\text{pair}}}{\sqrt{d}} + \varepsilon_v\Big). \tag{38}$$

We close the proof with two clarifications. First, the projector P^j is not an explicit module. The downstream computation of query j naturally reads information from the directions spanned by its own trajectory, which is mathematically equivalent to this projection. Second, the bound is meaningful when the gaps between native logits dominate the leakage scale ρ. This is exactly the regime in which the disentangling query retrieves its own trajectory from the superposed cache, which justifies the approximation in Eq. 12. □

A.3 Theoretical Generalization Bound

We now derive the error bound of ReMux. To recap, ReMux performs superposed reasoning for multiple queries in the first $(m-1)$ agents and disentangles valid answers in the final agent. The approximation error for each individual query consists of two components: the superposition error ϵ_{sup} and the disentanglement error ϵ_{dis}.

Superposition error ϵ_{sup}. According to Theorem A.2, a batch of queries can be superimposed with a weight of $1/n$ before feeding into the Transformer-based backbone F. The resulting approximation error is

$$\epsilon_{\text{sup}} = \Big\| \frac{1}{n} \sum_{j=1}^{n} \mathrm{F}\big(\phi^j(\mathbf{X}^j)\big) - \mathrm{F}\big(\mathbf{S}^{1:n}\big) \Big\| \;\lesssim\; \delta_{\text{attn}} + \delta_{\text{ffn}}, \tag{39}$$

where δ_{attn} collects the attention-side errors, namely the residual cross-subspace terms in the query-key scores and the temperature distortion induced by the $1/n^2$ scaling of the superimposed logits, both of which prevent the softmax from factorizing exactly across queries. δ_{ffn} arises from the remaining nonlinearities, namely the feed-forward activations and the normalization layers, which break exact linear superposition when the superimposed input passes through them. Both errors accumulate over the stacked Transformer layers.

Disentanglement error ϵ_{dis}. As established in Theorem A.3, the final agent disentangles the superposed trajectories by matching each disentangling query Q^j against the superposed keys and retrieving the corresponding values on its own value subspace $\mathcal{V}^j$. The error of this readout decomposes into a key-side term and a value-side term:

$$\epsilon_{\text{dis}} = (n-1)\, V_{\max}\Big(\frac{2\varepsilon_k}{\sqrt{d}} + \varepsilon_v\Big) = \widetilde{\mathcal{O}}\Big(\frac{n}{\sqrt{d}}\Big), \tag{40}$$

where $V_{\max} = \max_{j,p} \|V_p^j\|$ bounds the value norms and $\widetilde{\mathcal{O}}$ absorbs the constant $V_{\max}$ and the $\sqrt{r^j}$ factor. The key-side term arises from residual cross-query matching. The per-query random projections suppress the cross inner products to $|Q_t^j (K_p^{j'})^\top| \le \varepsilon_k = \widetilde{\mathcal{O}}(1/\sqrt{d})$ for all $j' \neq j$, which perturbs the softmax weights by at most $2(n-1)\varepsilon_k/\sqrt{d}$ in ℓ_1. The value-side term arises

from subspace leakage. The values of other trajectories lie in nearly orthogonal subspaces, so their components on $\mathcal{V}^j$ satisfy $\|P^j V_p^{j'}\| \le \varepsilon_v \|V_p^{j'}\|$, where ε_v also vanishes as $1/\sqrt{d}$ up to a $\sqrt{r^j}$ factor (with $r^j = \dim \mathcal{V}^j \le L$) and logarithmic factors. In contrast, the native semantic score $Q_t^j (K_p^j)^\top$ does not vanish, since the query and its own keys share the same projection. This nonvanishing self matching is precisely the signal through which query j retrieves its own trajectory from the superposed cache. Consequently, the disentanglement error grows linearly with the number of superposed queries and vanishes as $1/\sqrt{d}$ in the embedding dimension.

Total error bound. For superposing over queries, the total error is bounded as:

$$\epsilon_{\text{total}} = n\epsilon_{\text{query}}, \tag{41}$$

where $\epsilon_{\text{query}} = \epsilon_{\text{sup}} + \epsilon_{\text{dis}}$. The total error accumulates as more queries are superposed. To prevent performance degradation in answer generation, we restrict the superposition count n, as discussed in Appendix C.

B Complexity Analysis

This section presents the complexity analysis of ReMux from three aspects, including token consumption (Appendix B.1), time complexity (Appendix B.2), and space complexity (Appendix B.3). The analysis shows that ReMux consistently achieves lower overhead than the standard one-to-one paradigm across all three aspects.

B.1 Token Consumption Analysis

For the multi-agent system $\mathcal{M}$, standard one-to-one collaborative reasoning pipeline incurs a token overhead of $\mathcal{O}(mnc)$ in the best case, achieved when each agent is called exactly once, where m denotes the number of agents, n denotes the batch size and c denotes the length of reasoning steps per agent.

To achieve a reduction in token overhead, the design of ReMux is decomposed into two fundamental steps:

- In the first step, we form the continuous superposition tokens $\mathbf{S}^{1:n} \in \mathbb{R}^{k\times d}$ as a convex combination of a batch of queries via token-wise superposition. We feed $\mathbf{S}^{1:n}$ into the first $m-1$ agents $\{\mathrm{A}_i\}_{i=1}^{m-1}$. Because the output encapsulates the superimposed reasoning traces for all queries, the token overhead of the first $m-1$ agents reduces from $\mathcal{O}\big((m-1)nc\big)$ to $\mathcal{O}\big((m-1)c\big)$, where $\mathcal{O}\big((m-1)nc\big)$ denotes the token overhead of conventional one-to-one reasoning.
- In the second step, we deploy a disentanglement strategy on the final agent A_m, which generates the explicit answers to each query. The token overhead in the final agent is nc, consistent with standard one-to-one reasoning.

In summary, the total token overhead for ReMux is expressed as:

$$\mathcal{O}\big((m-1)c + nc\big). \tag{42}$$

We further quantify the token efficiency gain by computing the overhead ratio of the conventional one-to-one paradigm to our ReMux. Given baseline cost $\mathcal{O}(mnc)$ and our total cost $\mathcal{O}\big((m-1)c + nc\big)$, the ratio is expressed as:

$$\mathcal{O}\big(\frac{mn}{m-1+n}\big). \tag{43}$$

Eq. 43 shows that the token overhead of ReMux scales with the sum of the number of agents and the number of queries, rather than with their product as in the standard one-to-one paradigm, yielding efficiency gains that grow as more queries are superposed.

B.2 Time Complexity Analysis

In this section, we analyze the computational overhead of ReMux in comparison to the standard one-to-one multi-agent systems (MAS). Due to differences in the implementation efficiency of at-

tention across distinct backbones, we assume that the time complexity scales linearly with the token cost.

Let m denotes the number of agents in MAS, n denote the total number of query and c denotes the length of reasoning steps per agent. Each token is embedded into a d-dimensional representation.

Recall that standard MAS has to process n problems one by one, where the token-level operations involved in each problem is proportional to cd. Therefore, the time complexity of the standard one-to-one MAS is $\mathcal{O}(nmcd)$.

For REMUX, the primary computational overhead arises from three components: (i) latent superposition reasoning, (ii) superposition space alignment, and (iii) answers generation. During latent superposition reasoning, REMUX first employs superposition technology to mapping the c-step reasoning process for queries into a shared space through $m-1$ agents, incurring a computational cost of $\mathcal{O}((m-1)cd)$. During alignment, REMUX employ $\mathbf{W}_a$ to transform the hidden embeddings, incurring a cost of $\mathcal{O}((m-1)cd)$. During answer generation, REMUX generates c-step textual reasoning for each query, incurring a cost of $\mathcal{O}(ncd)$.

Overall, the total computational complexity of REMUX is given by:

$$\mathcal{O}\big(2mcd - 2cd + ncd\big). \tag{44}$$

We further quantify the gain by computing the overhead ratio of the conventional one-to-one paradigm to our REMUX. Given baseline cost $\mathcal{O}(nmcd)$ and our total cost $\mathcal{O}\big(2mcd - 2cd + ncd\big)$, the ratio is expressed as:

$$\mathcal{O}\big(\frac{mn}{2m-2+n}\big). \tag{45}$$

Eq. 45 shows that the time overhead of REMUX scales with the sum of the number of agents and the number of queries, rather than with their product as in the standard one-to-one paradigm, yielding efficiency gains that grow as more queries are superposed.

B.3 SPACE COMPLEXITY ANALYSIS

Following the previous notation, we analyze the space complexity of REMUX compared to the standard one-to-one MAS.

Recall that the standard MAS stores the thoughts for n problems independently in m agents. The total storage overhead is $\mathcal{O}(nmcd)$.

For REMUX, the primary storage overhead arises from two components: (i) superposition reasoning, (ii) answers generation. During superposition reasoning, the first $(m-1)$ agents compress the reasoning processes of n problems into a shared space, leading to a storage overhead of $\mathcal{O}((m-1)cd)$. During answers generation, the m-th agent needs to generate explicit n answers, incurring a cost of $\mathcal{O}(ncd)$. Overall, the total storage complexity of REMUX is given by

$$\mathcal{O}\big((m-1)cd + ncd\big). \tag{46}$$

We further quantify the gain by computing the overhead ratio of the conventional one-to-one paradigm to our REMUX. Given baseline cost $\mathcal{O}(nmcd)$ and our total cost $\mathcal{O}\big((m-1)cd + ncd\big)$, the ratio is expressed as:

$$\mathcal{O}\big(\frac{mn}{m-1+n}\big). \tag{47}$$

Eq. 47 shows that the space overhead of REMUX scales with the sum of the number of agents and the number of queries, rather than with their product as in the standard one-to-one paradigm, yielding efficiency gains that grow as more queries are superposed.

C SELECTION OF SUPERIMPOSED QUERY COUNT n

Let the optimal superposition count $n^\star$ be the maximum n that does not cause a reduction in task accuracy. In this section, we present a data-driven approach from an entropy perspective to determine $n^\star$.

The proposed method is motivated by the observation that interference arises when input embeddings from different queries are superimposed and propagated in the continuous superposition space. Hence, the larger the number of superimposed queries, the more severe the interference. Prior studies on representation entanglement (Menet et al., 2023; Gong et al., 2022; Liu et al., 2025) show that the degree of subspace overlap among multiple embedding sets can be rigorously quantified by the *spectral entropy* of the Gram matrix $\mathbf{G}$. Formally, given a batch of n query embeddings $\{\boldsymbol{z}^j\}_{j=1}^n$, we first construct the Gram matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ with entries $\mathbf{G}_{ij} = (\boldsymbol{z}^i)^\top \boldsymbol{z}^j$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ denote its eigenvalues sorted in descending order. The spectral entropy is defined as:

$$H(\mathbf{G}) = -\sum_{i=1}^{n} p_i \log p_i, \quad p_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \tag{48}$$

where the normalized coefficient p_i represents the fraction of total representational energy carried by the i-th principal component of the embedding batch. Intuitively, $H(\mathbf{G})$ measures how uniformly the embedding energy distributes across orthogonal latent directions, where a larger $H(\mathbf{G})$ implies that query embeddings span more mutually orthogonal subspaces and suffer less cross-query interference. Therefore, spectral entropy effectively quantifies the extent of subspace sharing between different query representations within the superposition space. The effective size of the embedding set can be derived directly from the spectral entropy:

$$r_{\text{eff}} = \exp\left(H(\mathbf{G})\right), \tag{49}$$

where $r_{\text{eff}} \in [1, n]$ denotes the *effective rank* of the Gram matrix, and indicates the number of effectively independent latent dimensions occupied by the batch of query embeddings. To eliminate the influence of batch size, we further perform a normalization operation on the effective rank to derive the normalized effective rank $\tilde{r}_{\text{eff}}$, which is constrained to the interval $(0, 1]$:

$$\tilde{r}_{\text{eff}} = \frac{r_{\text{eff}}}{n}, \tag{50}$$

where $\tilde{r}_{\text{eff}} = 1$ corresponds to the ideal case where all query embeddings are completely orthogonal with zero subspace overlap, and smaller values indicate a higher degree of linear correlation among query representations. This value can be computed efficiently via standard eigenvalue decomposition, which scales favorably with the batch size. A larger $\tilde{r}_{\text{eff}}$ indicates weaker linear dependence among query embeddings and thus less subspace overlap, implying that a larger multiplexing count n can be supported without incurring severe signal interference. Conversely, a smaller $\tilde{r}_{\text{eff}}$ suggests highly correlated representations and calls for a smaller n to mitigate interference accumulation across reasoning steps. Based on this insight, we establish a linear relation between the spectral measure and the optimal multiplexing count $n^\star$:

$$n^\star = \lceil \alpha \cdot \tilde{r}_{\text{eff}} + b \rceil, \tag{51}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation to the nearest integer, α is a scaling factor, and b is a bias term. We empirically obtain pairs of $(\tilde{r}_{\text{eff}}, n^\star)$ by executing REMUX across various benchmark datasets, and estimate the parameters via linear regression. Finally, Algorithm 1 summarizes the above procedure for determining the optimal $n^\star$ given a batch of query embeddings.

Algorithm 1 Selection of Multiplexing Query Count n

1: **Input:** Batch of query embeddings $\{\boldsymbol{z}^j\}_{j=1}^n$; scaling factor α; bias term b estimated from Eq. 51.
2: Construct the Gram matrix from query embeddings: $\mathbf{G}_{ij} = (\boldsymbol{z}^i)^\top \boldsymbol{z}^j$.
3: Compute the eigenvalue spectrum $\boldsymbol{\lambda}$ via eigenvalue decomposition: $\boldsymbol{\lambda} = \texttt{eig}(\mathbf{G})$.
4: Compute normalized energy fractions: $p_i = \lambda_i / \sum_{j=1}^{n} \lambda_j$.
5: Calculate spectral entropy and the rank: $H(\mathbf{G}) = -\sum_{i=1}^{n} p_i \log p_i$, $\tilde{r}_{\text{eff}} = \exp(H(\mathbf{G}))/n$.
6: Compute the optimal multiplexing query count: $n^\star = \lceil \alpha \cdot \tilde{r}_{\text{eff}} + b \rceil$.
7: **Output:** Optimal multiplexing query count $n^\star$.

D EXPERIMENTAL SETUP

This section presents detailed experimental settings for evaluating REMUX, covering dataset descriptions (Appendix D.1), multi-agent architectures (Appendix D.2), baseline comparisons (Appendix D.3), evaluation metrics (Appendix D.4) and implementation configurations (Appendix D.5).

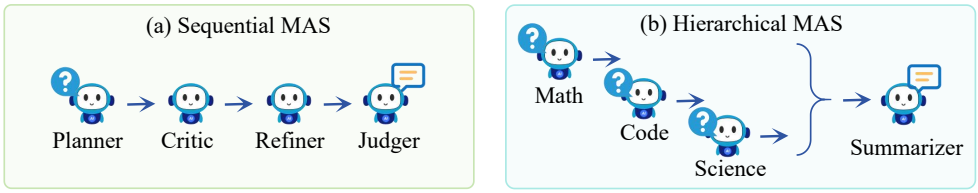


Figure 4: Schematic illustrating the sequential MAS architecture compared with the hierarchical MAS.

D.1 Dataset description

We evaluate ReMux on eight benchmarks spanning math, science, commonsense, and code generation: (i) ARC-E (Clark et al., 2018) is a widely used elementary science multiple-choice benchmark for evaluating commonsense knowledge of large language models, consisting of 2251 training and 2376 test questions. (ii) ARC-C (Clark et al., 2018) is a hard grade-school science multiple-choice benchmark for evaluating scientific reasoning of large language models, with 1119 training and 1172 test questions. (iii) GSM8K (Cobbe et al., 2021) is a commonly used benchmark for evaluating mathematical reasoning of large language models, with 7473 samples in the training split and 1319 samples in the test split. (iv) MedQA (Yang et al., 2025) is a medical multiple-choice benchmark for evaluating domain-specific reasoning of large language models, with 10178 training and 1273 test questions. (v) MBPP+ (Liu et al., 2023) is a strict Python code generation benchmark from EvalPlus with 399 problems, used to evaluate code synthesis of large language models. (vi) HumanEval+ (Liu et al., 2023) is a strict code generation benchmark from EvalPlus with 164 Python problems for evaluating functional correctness of large language models. (vii) GPQA (Rein et al., 2023) is a challenging expert-level STEM multiple-choice benchmark with 198 questions for evaluating advanced scientific reasoning of large language models. (viii) AIME25 (Zhang & Math-AI, 2025) is a hard competition math benchmark with 30 problems from the 2025 American Invitational Mathematics Examination for evaluating advanced mathematical reasoning of large language models.

D.2 Hierarchical and Sequential MAS Settings

In the Sequential MAS setting, agents execute in a strict chained order. Each agent receives outputs from the preceding agent as its input. The planner first devises a problem-solving plan. Subsequently, the critic examines and critiques this plan. Next, the refiner revises the plan incorporating the critic's feedback. Finally, the judger consumes the refined plan and produces the final answer.

In the hierarchical MAS setting, multiple specialized agents (Math Agent, Code Agent, Science Agent) operate in parallel. Each independent agent processes the original question separately and generates its own response. After all these specialist agents finish their reasoning, all their outputs are aggregated and passed to a dedicated summarizer agent. The summarizer analyses information from all parallel agents and derives the final answer.

D.3 Baselines

We compare ReMux against four baselines:

- *Single-Agent*: A single agent that performs text-based reasoning. This baseline verifies whether ReMux can effectively disentangle superimposed reasoning trajectories.
- *Sequential TextMAS* (Zhao et al., 2026): A chain-of-agents design with text-mediated collaboration. It adopts the sequential MAS architecture defined in D.2, serving to compare the superposition-based reasoning paradigm against text-space reasoning.
- *Hierarchical TextMAS* (Wang et al., 2025): Domain-specialized agents collaborate via a text summarizer. It follows the hierarchical MAS architecture defined in D.2, and enables comparison between superposition-based and text-space reasoning paradigms.

- *LatentMAS* (Zou et al., 2026): A pure latent-space MAS that processes each query independently in continuous representations. It has both sequential and hierarchical variants and acts as a strong SOTA baseline to compare superposition-based reasoning against latent one-to-one reasoning.

D.4 Metric Definition

To comprehensively assess both the effectiveness and efficiency of collaborative reasoning, we adopt four metrics: *Accuracy* (Acc.) for task performance, *Token* and *Time* for inference cost, and the superposition count n for multiplexing capacity.

- *Accuracy* (Acc.) measures the task-level performance of a method on each dataset, quantifying the fraction of queries for which the model's prediction matches the ground-truth answer:
$$\text{Acc.} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\big(\hat{y}_i = y_i\big) \times 100\%, \tag{52}$$
where N denotes the total number of test queries, y_i is the ground-truth answer, $\hat{y}_i$ is the model's prediction for the i-th query, and $\mathbf{1}(\cdot)$ is the indicator function that equals 1 when the prediction is correct and 0 otherwise.
- *Token* measures the inference cost of a method in terms of reasoning length, defined as the average number of reasoning steps produced per query across the entire multi-agent system:
$$\text{Token} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m}\sum_{j=1}^{m} L_{i,j}, \tag{53}$$
where N denotes the total number of queries, m denotes the number of agents in the system, and $L_{i,j}$ is the number of reasoning steps taken by the j-th agent for the i-th query. A lower *Token* value indicates higher token efficiency.
- *Time* measures the inference overhead of a method in terms of wall-clock latency, defined as the average time required to process a single query across the entire multi-agent system:
$$\text{Time} = \frac{1}{B}\sum_{i=1}^{B}\frac{T_i}{n}, \tag{54}$$
where B denotes the total number of batches, T_i is the wall-clock time consumed by the i-th batch, and n is the number of queries superimposed within each batch. A lower *Time* value indicates less inference overhead.
- We additionally report the superposition count n, i.e., the maximum number of queries that can be multiplexed within a single forward pass without degrading task accuracy.

For efficiency metrics, improvements are reported as the ratio of the strongest baseline to ours (i.e., $\text{Token}_{\text{baseline}}/\text{Token}_{\text{ours}}$ and $\text{Time}_{\text{baseline}}/\text{Time}_{\text{ours}}$), where higher values indicate greater token savings and speedup, respectively. For task effectiveness, we report the absolute accuracy difference between our method and the baseline in percentage points (i.e., $\text{Acc.}_{\text{ours}} - \text{Acc.}_{\text{baseline}}$), denoted by $\uparrow$/$\downarrow$ for improvements and degradations.

D.5 Implementation Details

We conduct experiments on the cloud server with 22 vCPU Intel(R) Xeon(R) Platinum 8470Q, two RTX PRO 6000(96GB), the Python 3.12 and the PyTorch 2.5.1 platform. For both baselines and our method, we uniformly set temperature $= 0.6$ and top-p $= 0.95$. For the first $m-1$ agents, we set the latent reasoning steps to 120 for MedQA, AIME25, and GPQA, and uniformly to 100 for all remaining datasets. In the final agent, we set the maximum generation length to 2048 tokens for ARC-E, ARC-C, and GSM8K; 4096 tokens for MedQA, MBPP+, and HumanEval+; 8192 tokens for GPQA; and 20000 tokens for AIME25. All agents operate on a shared LLM backbone. We fine-tune this backbone with LoRA to adapt it for superposition computation, with detailed fine-tuning procedures provided in Appendix E. At inference time, we evaluate ReMux following the standard dataset splits described in Appendix D.1.

E Detailed Training Procedure

Our training samples are drawn from the training splits of datasets including ARC-E, ARC-C, GSM8K, and MedQA. Within each dataset's original official training split, we randomly create custom training and validation subsets following an 8:2 split ratio. The original test splits are preserved for final evaluation, and are never visible during model fine-tuning.

We adopt a single LLM backbone shared across all agents. Following Hu et al. (2022), the weights of the original LLM backbone are fully frozen. We implement LoRA fine-tuning via the Hugging Face PEFT library (Mangrulkar et al., 2022), injecting LoRA modules into the *q_proj*, *k_proj*, *v_proj*, and *o_proj* self-attention projection layers. LoRA hyper-parameters are configured as rank $r = 16$, lora_alpha $= 32$, and LoRA dropout 0.1. We use the AdamW optimizer with a peak learning rate of 1×10^{-4}. Besides optimizing LoRA parameters, we also jointly train the space-alignment transformation matrix $\mathbf{W}_a$.

Our training employs a teacher-student paradigm with per-step alignment. The teacher pipeline conducts standard one-to-one inference over individual queries, whose outputs are projected via ϕ^j and superimposed to construct supervision targets. The student pipeline performs superposition-based inference, where the continuous supervised objective $\mathcal{L}_{\text{CSO}}$ (Eq. 9) uses cosine-distance loss to align student latent prefixes against teacher-built supervision signals. This regularization enforces the superposition state to preserve reasoning trajectories from all queries and mitigates representation drift. Finally, we obtain explicit outputs $\{\hat{\mathbf{e}}^j_{m,t}\}_{j=1}^n$ via the disentanglement procedure described in Eq. 13, from which we extract final predictions $\{\hat{y}^j\}_{j=1}^n$ for all n queries in a batch. Combined with ground-truth answers $\{y^j\}_{j=1}^n$ from the dataset, we compute the cross-entropy loss $\mathcal{L}_{\text{CE}}$ to penalize the mismatch between model predictions and ground-truth answers. The overall training objective combines both components as $\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{CSO}}$, as introduced in Section 4.4. Under this total objective, both LoRA parameters and $\mathbf{W}_a$ are updated during back-propagation. We train the model for a fixed budget of 3 epochs.

F Additional Experimental Results

We conduct a set of supplementary experiments to further evaluate REMUX. We first visualize the last-layer hidden representations of superimposed reasoning trajectories and show that the resulting representations occupy a distinct space while the disentangled trajectories recover the corresponding one-to-one reasoning patterns (Appendix F.1). We then conduct detailed ablation studies to evaluate the contributions of each component (Appendix F.2). A further comparison between REMUX and the single-agent baseline is provided in Appendix F.3. Next, we analyze the effect of the latent step size and identify the range that provides an effective balance between reasoning capacity and accumulated approximation error (Appendix F.4). We further investigate how the superposition count affects task accuracy and token consumption, revealing the trade-off between computational efficiency and cross-query interference as the number of superposed queries increases (Appendix F.5). To examine the effectiveness of continuous superposition space alignment, we visualize the distributional and geometric relationships between the raw hidden representations, supervision signals, and aligned representations, demonstrating that alignment mitigates representation drift and preserves superposed reasoning information (Appendix F.6). Finally, we evaluate REMUX on additional backbone models and across both sequential and hierarchical MAS settings, confirming the compatibility of the proposed framework with diverse model families and architectures, including a mixture-of-experts backbone (Appendix F.7).

F.1 Visualization of the Last-layer Hidden Representations of Superimposed Reasoning

In this section, we perform empirical evaluations with Qwen3-14B on ARC-E under both the sequential and hierarchical MAS settings. The superposition count of REMUX is set to 2. In the REMUX pipeline, the first three agents perform superposed reasoning, and the final agent generates the answer for each query separately. We select *TextMAS* as the standard one-to-one baseline, since every reasoning step of *TextMAS* operates within an explicit semantic space. In the *TextMAS* pipeline, each agent reasons over every independent query individually. We extract the last-layer

hidden embeddings at each reasoning step and project them onto a two-dimensional space using the t-SNE algorithm. As shown in Figure 5 and Figure 6, we highlight two key observations. (i) The superimposed reasoning traces are clearly separated from the one-to-one traces in the embedding space, indicating that our superimposed computation lies outside the standard semantic space. (ii) In the judger of the sequential setting and the summarizer of the hierarchical setting, the disentangled reasoning traces for answer generation overlap with their respective one-to-one counterparts, demonstrating that REMUX recovers the underlying reasoning trajectory of each query from the superposition.

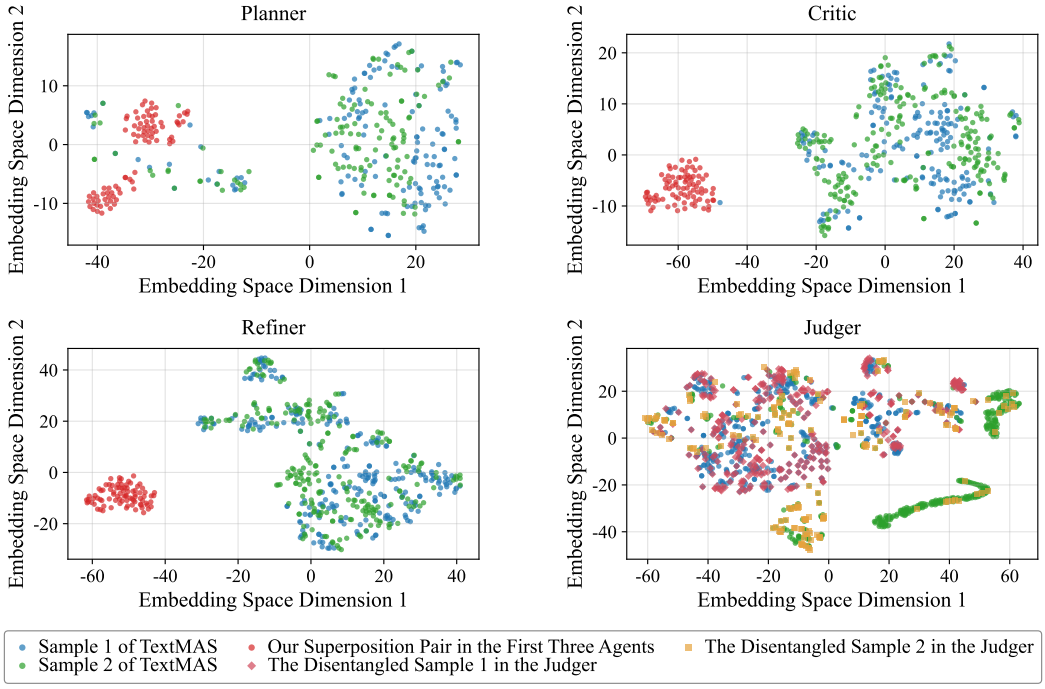


Figure 5: Visualization of the last-layer hidden representations of superimposed reasoning for four roles under the sequential MAS setting.

F.2 ABLATION STUDY

To evaluate the effectiveness of the proposed components in REMUX, we design five ablation variants: (i) *Single-Agent* employs a single agent to evaluate performance; (ii) *w/o* ϕ superimposes multiple queries without the ϕ transformation; (iii) *w/o KV-comm.* communicates directly via the superposition outputs $\hat{s}_{i,c}^{1:n}$; (iv) *w/o* $\mathbf{W}_a$&$\mathcal{L}_{CSO}$ denotes the setting without the interference mitigation strategy based on space alignment; (v) *w/o* $\mathcal{L}_{Total}$ runs inference without any fine-tuning. We conduct evaluations on sequential MAS with the Qwen3-4B model, and the results are presented in Table 3 and Table 4.

We highlight the following observations: (i) Compared with the *Single-Agent* baseline, REMUX achieves substantial accuracy gains on all benchmarks. This shows the advantage of our superimposed reasoning paradigm, as it retains multi-agent collaborative intelligence. (ii) Removing the ϕ transformation also leads to measurable performance losses, as without transformation superposition cannot effectively preserve the information of multiple concurrent queries. (iii) When disabling KV-communication (*w/o KV-comm.*), we observe both an accuracy decline and increased token consumption and inference time, confirming that our superimposed KV-Cache communication reduces computational overhead while retaining reasoning signals. (iv) Without the space-alignment interference mitigation ($\mathbf{W}_a$&$\mathcal{L}_{CSO}$), the model suffers moderate accuracy degradation, showing that interference between superimposed queries is a critical bottleneck. (v) The variant *w/o* $\mathcal{L}_{Total}$ exhibits the most severe performance drop, demonstrating the necessity of the fine-tuning objective for

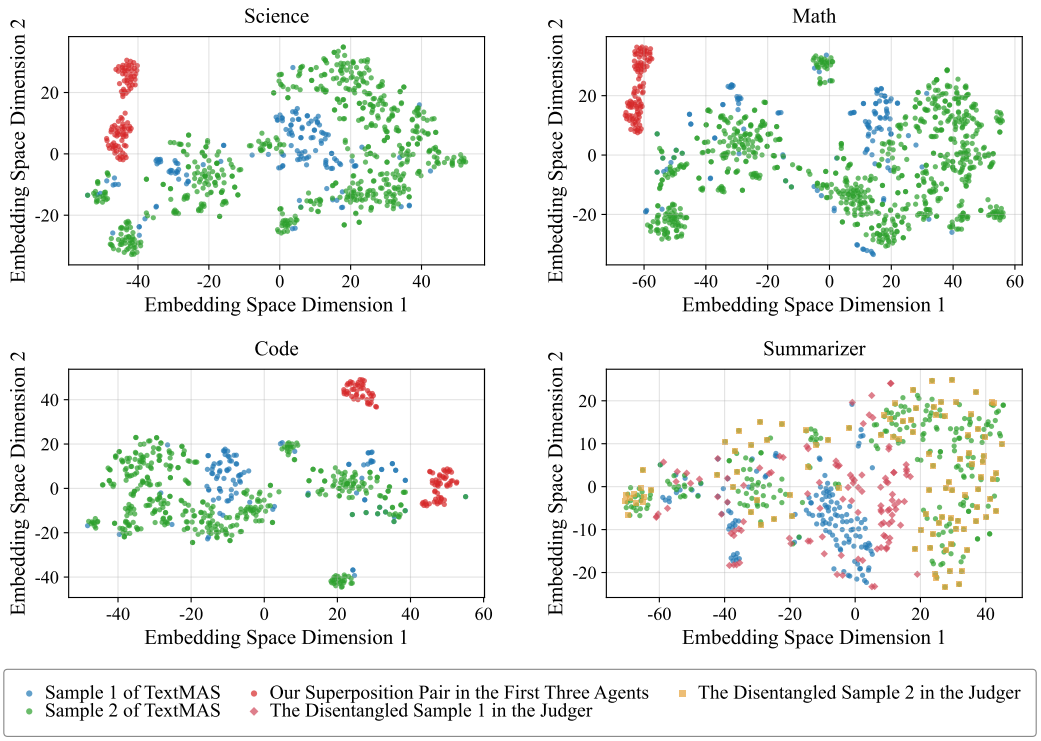


Figure 6: Visualization of the last-layer hidden representations of superimposed reasoning for four roles under the hierarchical MAS setting.

superposition-based computation. In summary, our proposed mechanisms, including superposition transformation, superimposed KV-Cache communication, continuous supervised loss and model fine-tuning, all contribute to the overall model performance.

Variants	Datasets											
	ARC-E			ARC-C			GSM8K			MedQA		
	Acc.	Token	Time	Acc.	Token	Time	Acc.	Token	Time	Acc.	Token	Time
REMUX	92.1	302	0.7	90.3	365	0.7	87.5	461	1.1	65.2	607	1.5
Single-Agent	83.5	156	0.2	78.0	289	0.2	84.9	276	0.5	51.0	498	0.6
w/o ϕ	86.6	302	0.7	82.4	365	0.7	85.3	461	1.1	55.7	607	1.5
w/o KV-comm.	84.0	481	0.9	78.3	394	0.8	85.0	535	1.3	51.4	694	1.8
w/o $\mathbf{W}_a$&$\mathcal{L}_{CSO}$	88.9	302	0.7	85.6	365	0.7	86.3	461	1.1	55.2	607	1.5
w/o $\mathcal{L}_{Total}$	74.8	302	0.7	62.0	365	0.7	75.3	461	1.1	45.2	607	1.5

Table 3: Ablation studies on REMUX evaluated on ARC-E, ARC-C, GSM8K and MedQA.

F.3 COMPARISON WITH SINGLE-AGENT

In this section, we further supplement the comparison results between the single-agent baseline (Single-Agent) and REMUX under two MAS settings, i.e., sequential MAS and hierarchical MAS. We conduct evaluations on eight datasets using both Qwen3-8B and Qwen3-14B backbones. The results under sequential MAS and hierarchical MAS are reported in Fig. 7 and Fig. 8, respectively. Under the sequential MAS setting, REMUX obtains accuracy gains ranging from 1.2% to 7.7% across eight datasets. For the hierarchical MAS setting, REMUX achieves accuracy improvements ranging from 2.6% to 10.3% across eight datasets. These performance gains demonstrate that RE-

Variants	Datasets											
	HumanEval+			GPQA			AIME25			MBPP+		
	Acc.	Token	Time	Acc.	Token	Time	Acc.	Token	Time	Acc.	Token	Time
REMUX	77.5	437	2.9	41.3	1623	6.5	50.0	1734	7.1	71.0	419	0.7
Single-Agent	75.0	386	0.7	35.0	1375	9.4	37.2	19467	263.1	65.3	392	0.4
w/o ϕ	76.8	437	2.9	40.2	1625	6.5	48.9	1732	7.1	68.1	419	0.7
w/o KV-comm.	75.1	492	3.2	36.0	1701	6.8	41.5	1852	7.6	66.0	496	1.0
w/o $\mathbf{W}_a$&$\mathcal{L}_{CSO}$	75.0	432	2.9	35.0	1643	6.5	46.3	1780	7.1	69.4	496	0.7
w/o $\mathcal{L}_{Total}$	72.5	430	2.9	32.7	1620	6.5	35.8	1741	7.1	60.8	419	0.7

Table 4: Ablation studies on REMUX evaluated on HumanEval+, GPQA, AIME25 and MBPP+.

MUX can effectively reuse superimposed historical reasoning trajectories to produce higher-quality responses compared with a single agent.

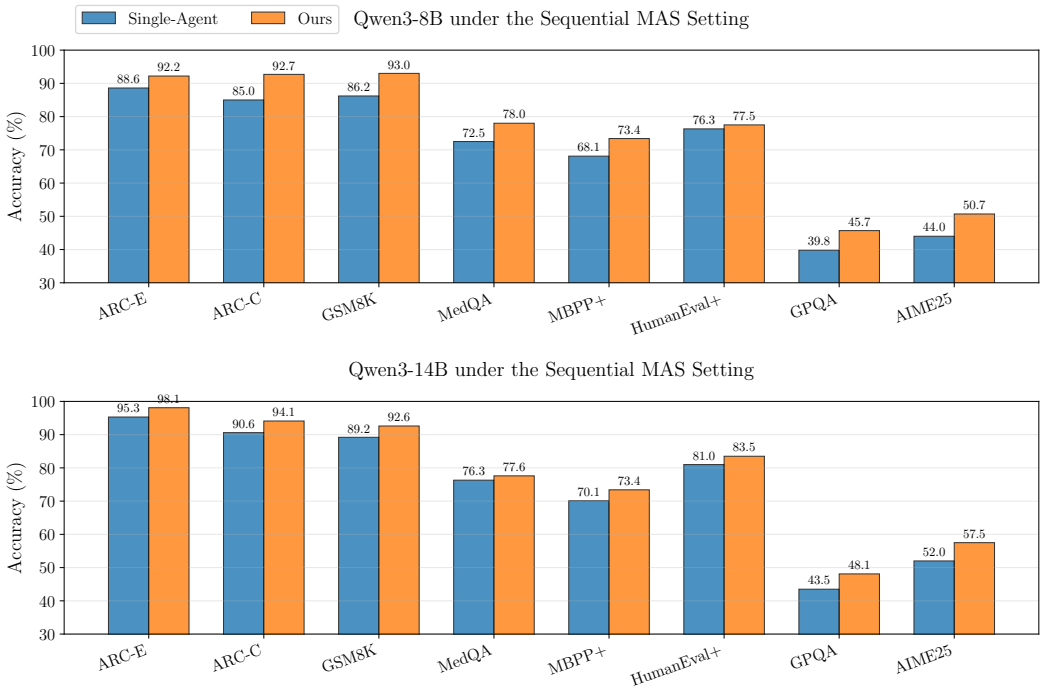


Figure 7: Comparison between REMUX and Single-Agent under the sequential MAS setting.

F.4 PARAMETER ANALYSIS ON LATENT STEP

In this section, we investigate the impact of the latent step size on task accuracy. We conduct experiments with Qwen3-4B under the sequential MAS setting, varying the latent step size and observing the resulting accuracy across eight datasets. We analyze the scenarios where the superposition number n is 1 and 15, respectively. The results are presented in Figure 9.

We observe a clear rise-and-fall trend. As the latent step size increases, accuracy first improves, since longer latent trajectories provide richer reasoning signals for disentanglement and answer generation. Beyond a certain range, however, accuracy begins to degrade, as excessively long latent trajectories accumulate step-wise approximation errors and dilute the critical information for answer generation. This indicates the existence of a sweet spot for the latent step size. Empirically, the best performance is achieved when the latent step size falls between 60 and 120 steps.

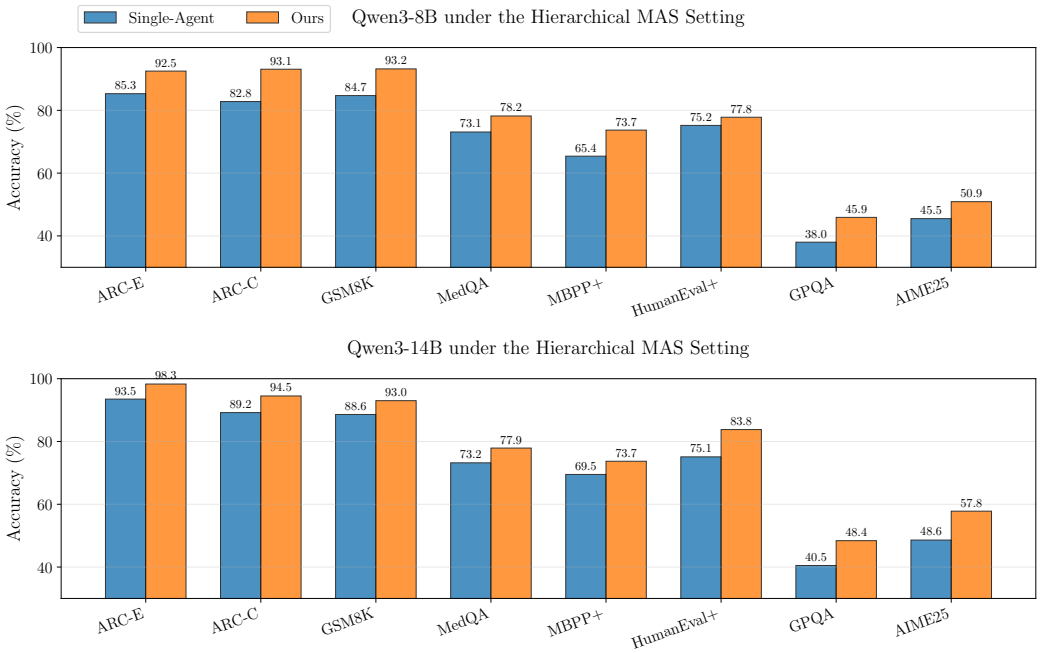


Figure 8: Comparison between REMUX and Single-Agent under the hierarchical MAS setting.

We further compare the non-superposed setting ($n = 1$) with the superposed setting ($n = 15$). Both settings exhibit the same rise-and-fall pattern across datasets, which indicates that this phenomenon stems from the latent step size itself rather than from superposition.

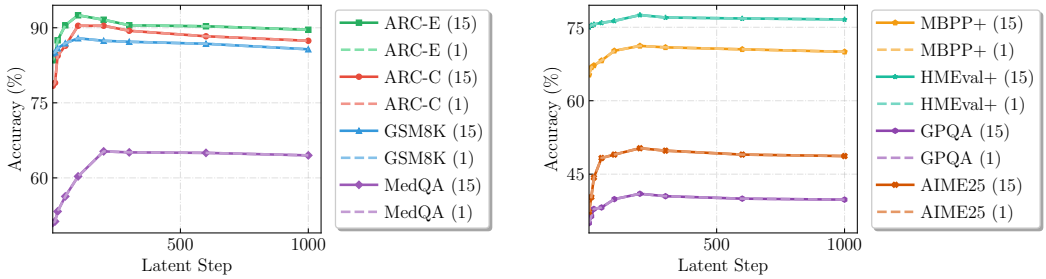


Figure 9: Parameter Analysis on Latent Step.

F.5 IMPACT OF SUPERPOSITION COUNT

In this section, we analyze the impact of the superposition count n on both task accuracy and token consumption. We conduct experiments with Qwen3-4B under the sequential MAS setting and evaluate on all eight datasets. The results are shown in Figure 10 and Figure 11.

We obtain the following two observations. On the one hand, task accuracy remains stable when n is small but degrades as n grows excessively. This is because more superposed queries induce stronger cross-query interference in the superposition space, which accumulates along the reasoning trajectories and eventually harms disentanglement at the final agent. This observation is consistent with the theoretical error bound in Appendix A.3, where the total error grows linearly with the number of superposed queries. On the other hand, token consumption decreases monotonically as n increases, since the reasoning trajectories of all queries within a batch share the same latent sequence and the reasoning cost is amortized across queries.

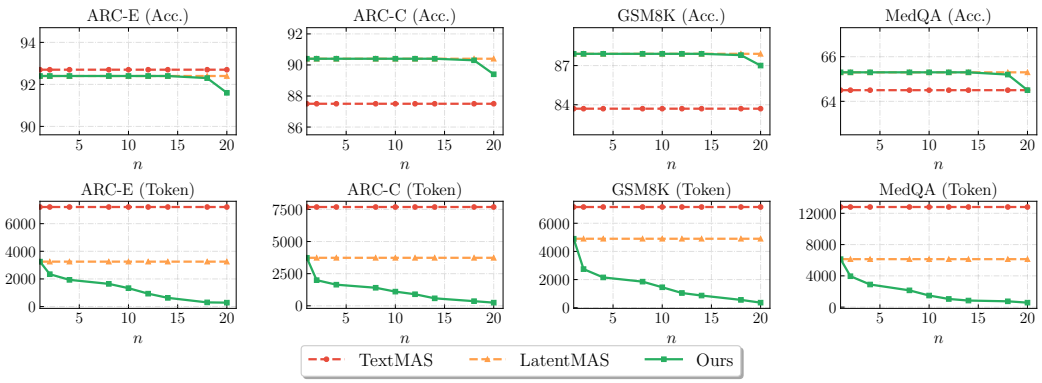


Figure 10: Analysis of accuracy and token consumption with increasing superposition count (n), evaluated on ARC-E, ARC-C, GSM8K and MedQA.

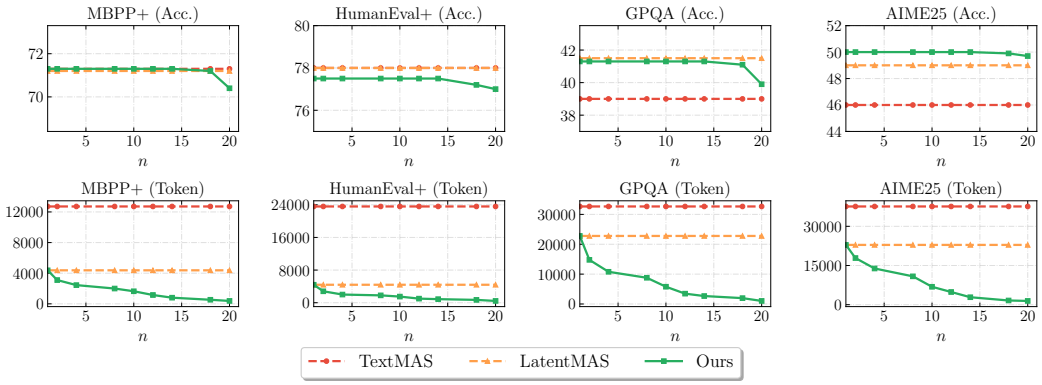


Figure 11: Analysis of accuracy and token consumption with increasing superposition count (n), evaluated on MBPP+, HumanEval+, GPQA and AIME25.

In summary, the superposition count n controls a trade-off between task accuracy and token efficiency. A moderate n preserves accuracy while delivering substantial token savings, whereas an excessively large n amplifies interference and degrades performance. We further provide an algorithm for selecting the value of n, as presented in Appendix C.

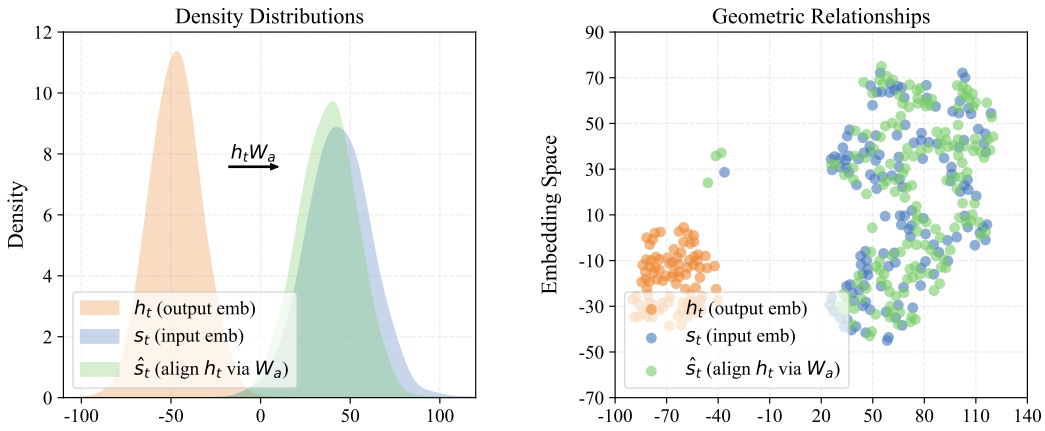


Figure 12: Effectiveness of continuous superposition space alignment in the hierarchical MAS.

F.6 EFFECTIVENESS OF THE CONTINUOUS SUPERPOSITION SPACE ALIGNMENT

To verify the effectiveness of space alignment in REMUX, we visualize the density distributions and the geometric relationships of the superimposed representations in the embedding space. The experiments are conducted with Qwen3-14B on ARC-E under both the sequential and hierarchical MAS settings. The results under the sequential setting and the hierarchical setting are presented in Figure 3(b) and Figure 12, respectively.

We obtain the following two observations. First, in the density distributions, the raw last-layer output $\mathbf{h}_t$ deviates from the supervision signal $\mathbf{s}_t$, whereas the aligned representation $\hat{\mathbf{s}}_t$ after the $\mathbf{W}_a$ transformation closely matches the distribution of $\mathbf{s}_t$. Second, in the geometric relationships, $\hat{\mathbf{s}}_t$ and $\mathbf{s}_t$ cluster together in the embedding space, while $\mathbf{h}_t$ lies apart from them. The deviation of $\mathbf{h}_t$ arises because superimposed hidden representations drift over successive reasoning steps, so the input prefix gradually leaves the continuous superposition space. The $\mathbf{W}_a$ transformation, trained with the continuous supervised objective $\mathcal{L}_{CSO}$, anchors the input prefix back to the superposition space at every reasoning step, which corrects the accumulated drift and preserves the reasoning information of all n queries in the shared representations. In summary, the space alignment strategy effectively mitigates representation drift and cross-query interference, ensuring that the superimposed reasoning trajectories capture the reasoning of all queries. This validates the design of the interference mitigation module in REMUX.

F.7 TEST REMUX IN ADDITIONAL BACKBONES

To further verify the model-agnostic design of REMUX, we additionally evaluate it on six backbones from three model families under both the sequential and hierarchical MAS settings, and further report the hierarchical MAS results of the Qwen3 series. Specifically, the results on Qwen3-4B/8B/14B under the hierarchical setting are reported in Table 6, the results on DeepSeek-R1-Distill-Qwen-8B/14B/32B are reported in Table 7 for the sequential setting and Table 8 for the hierarchical setting, and the results on Ministral-3-8B, Mistral-Nemo-12B, and Moonlight-16B-A3B are reported in Table 9 for the sequential setting and Table 10 for the hierarchical setting.

On the Qwen3 series under the hierarchical setting, REMUX reduces token consumption by $13.1\times$ and accelerates inference by $6.3\times$–$8.2\times$ across the three model scales. On the DeepSeek-R1-Distill-Qwen series, REMUX reduces token consumption by $10.7\times$–$12.0\times$ and accelerates inference by $3.6\times$–$7.0\times$ across the three model scales and both MAS settings, while the average accuracy drop remains within 0.5. On Ministral-3-8B and Mistral-Nemo-12B, REMUX achieves $11.5\times$–$14.2\times$ token reduction and $4.1\times$–$6.3\times$ speedup with at most 0.6 accuracy degradation. Notably, REMUX also remains effective on the MoE backbone Moonlight-16B-A3B, yielding $12.3\times$–$12.6\times$ token reduction and $4.1\times$–$4.5\times$ speedup, since expert activation only replaces the feed-forward modules while the attention mechanism underpinning our superposition is untouched. These results demonstrate that REMUX is compatible with diverse backbones across model families and architectures, consistently delivering substantial efficiency gains without compromising task accuracy.

Tasks	Metrics	Qwen3-4B				Qwen3-8B				Qwen3-14B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	92.7	92.4	92.1	0.6 ↓	92.8	92.1	92.2	0.1 ↓	98.2	98.0	98.1	0.1 ↑
	Token	7205	3251	302	**10.8**×	7326	3146	295	**10.7**×	6981	3139	257	**12.2**×
	Time	20.5	3.0	0.7	**4.3**×	21.4	4.1	0.9	**4.5**×	27.5	6.2	1.2	**5.1**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
ARC-C	Acc.	87.5	90.4	90.3	0.1 ↓	93.2	93.5	92.7	0.8 ↓	95.2	94.7	94.1	0.6 ↓
	Token	7685	3734	365	**10.2**×	7623	3695	361	**10.2**×	7459	3698	337	**11.0**×
	Time	15.7	3.1	0.7	**4.4**×	21.8	6.2	0.9	**6.8**×	26.6	9.3	1.1	**8.5**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
GSM8K	Acc.	83.7	87.9	87.5	0.4 ↓	91.4	92.8	93.0	0.2 ↑	92.5	93.1	92.6	0.5 ↓
	Token	7155	4900	461	**10.6**×	7083	4875	458	**10.6**×	6981	4881	420	**11.6**×
	Time	18.4	6.8	1.1	**6.2**×	20.6	6.9	1.1	**6.3**×	25.7	9.0	1.3	**6.9**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
MedQA	Acc.	64.5	65.3	65.2	0.1 ↓	77.1	77.9	78.0	0.1 ↑	78.0	77.2	77.6	0.4 ↑
	Token	12806	6135	607	**10.1**×	12806	6083	585	**10.4**×	11572	6059	560	**10.8**×
	Time	61.7	9.8	1.5	**6.5**×	46.6	10.1	1.8	**5.6**×	56.3	10.0	2.0	**5.0**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
MBPP+	Acc.	71.3	71.3	71.0	0.3 ↓	72.0	73.0	73.4	0.4 ↑	73.5	73.8	73.4	0.4 ↓
	Token	12714	4377	419	**10.4**×	12174	4392	410	**10.7**×	11769	4306	407	**10.6**×
	Time	26.5	4.4	0.7	**6.3**×	40.1	7.2	1.3	**5.5**×	64.6	8.6	1.5	**5.7**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
HumanEval+	Acc.	78.0	78.0	77.5	0.5 ↓	76.8	76.8	77.5	0.7 ↑	84.8	83.1	83.5	0.4 ↑
	Token	23592	4402	437	**10.1**×	22347	4395	420	**10.5**×	20584	4356	419	**10.4**×
	Time	25.1	10.7	2.9	**3.7**×	33.1	12.8	3.0	**4.3**×	47.0	15.6	3.5	**4.5**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
GPQA	Acc.	39.0	41.5	41.3	0.2 ↓	43.0	45.6	45.7	0.1 ↑	47.8	47.8	48.1	0.3 ↑
	Token	32657	22782	1623	**14.0**×	31429	23165	1590	**14.5**×	30840	21968	1584	**13.9**×
	Time	136.4	28.6	6.5	**4.4**×	157.0	40.1	6.8	**5.9**×	189.3	58.7	7.0	**8.4**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
AIME25	Acc.	46.0	49.0	50.0	1.0 ↑	46.0	50.0	50.7	0.7 ↑	50.0	57.0	57.5	0.5 ↑
	Token	37601	22883	1734	**13.1**×	36486	22176	1659	**13.4**×	34065	22047	1683	**13.1**×
	Time	431.8	56.8	7.1	**8.0**×	447.0	69.7	7.6	**9.2**×	558.3	73.0	8.0	**9.1**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
Average	Acc.	70.3	72.0	71.9	0.1 ↓	74.0	75.2	75.4	0.2 ↑	77.5	78.1	78.1	**0.0**
	Token	17677	9058	744	**12.2**×	17159	8991	722	**12.4**×	16281	8807	708	**12.4**×
	Time	92.0	15.4	2.7	**5.8**×	98.5	19.6	2.9	**6.7**×	124.4	23.8	3.2	**7.4**×
	n	1	1	15	-	1	1	20	-	1	1	20	-

Table 5: Main results of our method on eight datasets under the sequential MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), average time consumption per query ("Time"), and the superposition count ("n"). Improvements over SOTA baselines use ratios of (SOTA baselines/Ours).

Tasks	Metrics	Qwen3-4B				Qwen3-8B				Qwen3-14B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	92.6	92.6	92.4	0.3 ↓	92.9	92.4	92.5	0.1 ↑	98.1	98.1	98.3	0.2 ↑
	Token	6820	3251	281	**11.6**×	6795	3146	270	**11.7**×	6510	3139	232	**13.5**×
	Time	19.4	3.0	0.6	**5.0**×	20.1	4.1	0.8	**5.1**×	25.8	6.2	1.0	**6.2**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
ARC-C	Acc.	87.7	90.6	90.5	0.1 ↓	93.1	93.7	93.1	0.6 ↓	95.3	94.9	94.5	0.4 ↓
	Token	7140	3734	334	**11.2**×	7085	3695	326	**11.3**×	6935	3698	301	**12.3**×
	Time	14.6	3.1	0.6	**5.2**×	20.3	6.2	0.8	**7.8**×	24.9	9.3	1.0	**9.3**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
GSM8K	Acc.	83.9	88.1	87.8	0.3 ↓	91.3	93.0	93.2	0.2 ↑	92.4	93.3	93.0	0.3 ↓
	Token	6690	4900	426	**11.5**×	6635	4875	421	**11.6**×	6490	4881	384	**12.7**×
	Time	17.2	6.8	1.0	**6.8**×	19.1	6.9	1.0	**7.0**×	23.9	9.0	1.2	**7.5**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
MedQA	Acc.	64.4	65.5	65.4	0.1 ↓	77.3	78.1	78.2	0.1 ↑	78.1	77.4	77.9	0.5 ↑
	Token	11940	6135	562	**11.0**×	11860	6083	541	**11.2**×	10780	6059	512	**11.8**×
	Time	57.2	9.8	1.4	**7.0**×	43.5	10.1	1.6	**6.3**×	52.4	10.0	1.8	**5.6**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
MBPP+	Acc.	71.4	71.5	71.3	0.2 ↓	71.9	73.2	73.7	0.5 ↑	73.4	74.0	73.7	0.3 ↓
	Token	11820	4377	382	**11.5**×	11380	4392	370	**11.9**×	10970	4306	365	**11.8**×
	Time	24.7	4.4	0.6	**7.3**×	37.4	7.2	1.2	**6.0**×	60.2	8.6	1.4	**6.1**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
HumanEval+	Acc.	77.9	78.2	77.8	0.4 ↓	76.9	77.0	77.8	0.8 ↑	84.7	83.3	83.8	0.5 ↑
	Token	21780	4402	401	**11.0**×	20640	4395	380	**11.6**×	19050	4356	376	**11.6**×
	Time	23.4	10.7	2.6	**4.1**×	30.8	12.8	2.7	**4.7**×	43.9	15.6	3.2	**4.9**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
GPQA	Acc.	39.1	41.7	41.6	0.1 ↓	42.9	45.8	45.9	0.1 ↑	47.7	48.0	48.4	0.4 ↑
	Token	30140	22782	1468	**15.5**×	28970	23165	1424	**16.3**×	28390	21968	1407	**15.6**×
	Time	126.8	28.6	6.0	**4.8**×	145.2	40.1	6.2	**6.5**×	174.1	58.7	6.4	**9.2**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
AIME25	Acc.	45.9	49.2	50.2	1.0 ↑	46.1	50.2	50.9	0.7 ↑	50.1	57.2	57.8	0.6 ↑
	Token	34730	22883	1562	**14.7**×	33560	22176	1482	**15.0**×	31250	22047	1490	**14.8**×
	Time	400.2	56.8	6.6	**8.6**×	414.3	69.7	7.0	**9.9**×	516.4	73.0	7.4	**10.0**×
	n	1	1	15	-	1	1	20	-	1	1	20	-
Average	Acc.	70.4	72.4	72.1	0.3 ↓	73.9	75.6	75.7	0.1 ↑	77.5	78.3	78.4	0.1 ↑
	Token	16390.0	8906.8	677.0	**13.1**×	15865.0	8591.0	654.3	**13.1**×	14922.5	8306.8	634.4	**13.1**×
	Time	85.4	15.1	2.4	**6.3**×	88.8	18.4	2.8	**6.6**×	101.4	23.8	2.9	**8.2**×
	n	1	1	15	-	1	1	20	-	1	1	20	-

Table 6: Main results of REMUX on eight datasets under the hierarchical MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), average time consumption per query ("Time"), and the superposition count ("n"). For Token and Time, improvements over SOTA baselines are given by SOTA baselines/Ours and marked with ×.

Tasks	Metrics	DeepSeek-R1-Distill-Qwen-8B				DeepSeek-R1-Distill-Qwen-14B				DeepSeek-R1-Distill-Qwen-32B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	92.0	92.0	91.7	0.3↓	98.0	97.6	97.5	0.5↓	98.0	98.0	97.8	0.2↓
	Token	7045	3162	293	**10.8**×	5796	3105	268	**11.6**×	6284	4083	404	**10.1**×
	Time	18.5	4.0	0.7	**5.7**×	35.0	5.7	1.0	**5.7**×	70.8	13.1	2.6	**5.0**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
ARC-C	Acc.	92.7	92.3	92.0	0.7↓	95.8	95.8	95.6	0.2↓	94.5	92.0	91.3	0.7↓
	Token	7283	3536	352	**10.0**×	6841	3704	331	**11.2**×	6670	5381	526	**10.2**×
	Time	19.6	5.9	0.8	**7.4**×	42.7	7.2	1.1	**6.5**×	78.6	17.5	2.9	**6.0**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
GSM8K	Acc.	91.5	92.3	91.5	0.8↓	92.4	92.4	92.3	0.1↓	93.0	93.0	92.7	0.3↓
	Token	6894	4823	449	**10.7**×	6842	4295	423	**10.2**×	6433	4180	384	**10.9**×
	Time	18.8	6.5	1.0	**6.5**×	26.1	6.7	1.2	**5.6**×	70.6	10.5	2.0	**5.3**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
MedQA	Acc.	77.0	77.5	77.0	0.5↓	78.0	78.5	78.2	0.3↓	80.4	80.0	79.5	0.9↓
	Token	11850	5922	590	**10.0**×	11985	5945	568	**10.5**×	8055	7352	729	**10.1**×
	Time	37.5	9.5	1.6	**5.9**×	56.8	8.5	1.7	**5.0**×	94.5	22.5	4.4	**5.1**×
	n	1	1	15	-	1	1	15	-	1	1	20	
MBPP+	Acc.	70.4	71.0	70.5	0.5↓	73.5	73.7	73.5	0.2↓	73.6	73.0	72.9	0.7↓
	Token	11593	4251	402	**10.6**×	10845	4254	411	**10.4**×	10202	6937	645	**10.8**×
	Time	37.5	6.8	1.1	**6.2**×	55.1	7.6	1.2	**6.3**×	138.8	20.1	3.1	**6.5**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
HumanEval+	Acc.	76.5	76.5	76.3	0.2↓	84.5	84.2	83.5	1.0↓	78.4	78.2	78.0	0.4↓
	Token	21535	4215	403	**10.5**×	18584	4356	419	**10.4**×	14999	6832	574	**11.9**×
	Time	30.6	11.5	2.8	**4.1**×	46.5	15.6	3.5	**4.5**×	163.6	19.4	4.5	**4.3**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
GPQA	Acc.	43.0	44.5	44.0	0.5↓	47.8	47.8	47.7	0.1↓	43.5	43.5	43.0	0.5↓
	Token	29854	20439	1550	**13.2**×	27965	19032	1481	**12.9**×	27625	18956	1524	**12.4**×
	Time	147	35.5	5.2	**6.8**×	183.5	49.1	6.2	**7.9**×	344.8	47.0	18.8	**2.5**×
	n	1	1	15	-	1	1	15	-	1	1	20	
AIME25	Acc.	46.0	48.3	48.2	0.1↓	50.0	52.4	52.7	0.3↑	52.6	52.3	52.0	0.6↓
	Token	33957	20411	1652	**12.4**×	32054	20468	1590	**12.9**×	32575	21435	1874	**11.4**×
	Time	405	60.2	7.1	**8.5**×	405.1	59.4	8.1	**7.3**×	473.3	79.4	10.3	**7.7**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
Average	Acc.	73.6	74.2	73.9	0.3↓	77.5	77.8	77.6	0.2↓	76.8	76.3	75.9	0.4↓
	Token	16251	8345	711	**11.7**×	15114	8145	686	**11.9**×	14105	9395	833	**11.3**×
	Time	89.3	17.5	2.5	**7.0**×	106.4	20.0	3.0	**6.7**×	179.4	28.7	6.1	**4.7**×
	n	1	1	15	-	1	1	15	-	1	1	20	-

Table 7: Main evaluation results of REMUX on three DeepSeek backbones under the sequential MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), average speed per query ("Time"), and the superposition count ("n"). For Token and Time, improvements over SOTA baselines are given by SOTA baselines/Ours and marked with $\times$.

Tasks	Metrics	DeepSeek-R1-Distill-Qwen-8B				DeepSeek-R1-Distill-Qwen-14B				DeepSeek-R1-Distill-Qwen-32B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	90.6	90.2	90.0	0.6↓	98.0	98.0	97.5	0.5↓	96.5	96.3	96.0	0.5↓
	Token	8241	3218	305	**10.6**×	7942	3197	270	**11.8**×	7941	4376	412	**10.6**×
	Time	22.6	4.3	0.7	**6.1**×	43.3	6.5	10	**6.5**×	89.4	16.5	2.7	**6.1**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
ARC-C	Acc.	92.0	91.5	91.4	0.6↓	93.5	93.5	92.8	0.7↓	93.0	93.0	92.1	0.9↓
	Token	8146	3602	358	**10.1**×	8390	3844	347	**11.1**×	8035	5531	539	**10.3**×
	Time	23.0	6.1	0.7	**8.7**×	46.5	7.8	1.1	**7.1**×	91.7	20.6	3.0	**6.9**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
GSM8K	Acc.	88.3	88.8	88.5	0.3↓	92.6	92.6	92.3	0.3↓	92.5	92.0	91.8	0.7↓
	Token	8152	497	452	**1.1**×	8346	4301	428	**10.0**×	7850	4207	392	**10.7**×
	Time	22.3	7.0	1.0	**7.0**×	46.0	7.0	1.2	**4.7**×	89.0	15.7	2.1	**7.5**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
MedQA	Acc.	75.0	75.0	74.5	0.5↓	75.2	75.0	74.8	0.4↓	79.4	79.0	78.8	0.6↓
	Token	13482	6004	595	**10.1**×	13547	6032	580	**10.4**×	9784	7150	663	**10.8**×
	Time	43.2	10.5	1.7	**6.2**×	62.5	9.3	2.1	**4.4**×	104.1	29.4	3.2	**9.2**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
MBPP+	Acc.	70.0	71.0	70.2	0.8↓	70.6	69.8	69.8	0.8↓	72.0	72.0	71.8	0.2↓
	Token	13274	4425	415	**10.7**×	14243	4394	417	**10.5**×	11846	7452	659	**11.3**×
	Time	39.6	7.7	1.4	**5.5**×	67.8	8.3	1.2	**6.9**×	148.0	31.5	3.2	**9.8**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
HumanEval+	Acc.	75.4	75.2	75.0	0.4↓	81.6	82.5	81.6	0.9↓	78.4	78.0	78.1	0.3↓
	Token	23464	4394	416	**10.6**×	21564	4543	439	**10.3**×	16845	7591	594	**12.8**×
	Time	34.2	12.1	3.0	**4.0**×	52.5	18.4	3.7	**5.0**×	176.4	32.0	4.7	**6.8**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
GPQA	Acc.	40.0	43.0	42.2	0.8↓	45.6	45.6	45.0	0.6↓	43.0	43.0	42.5	0.5↓
	Token	31865	20975	1627	**12.9**×	32564	20454	1524	**13.4**×	29076	19780	1683	**11.8**×
	Time	152	36.6	5.8	**6.3**×	234.7	53.8	7.0	**7.7**×	397.8	49.6	19.5	**2.5**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
AIME25	Acc.	44.0	45.2	44.8	0.4↓	48.5	48.5	48.0	0.5↓	52.0	51.8	51.7	0.3↓
	Token	33852	21094	1670	**12.6**×	33865	21057	1649	**12.8**×	33151	22631	1886	**12.0**×
	Time	410	62.9	7.8	**8.1**×	412.5	61.2	8.3	**7.4**×	479.5	80.5	11.2	**7.2**×
	n	1	1	15	-	1	1	15	-	1	1	20	-
Average	Acc.	71.9	72.5	72.1	0.4↓	76.0	75.9	75.5	0.5↓	75.9	75.8	75.4	0.5↓
	Token	17685	7901	737	**10.7**×	17784	8472	704	**12.0**×	15566	9815	853	**11.5**×
	Time	93.3	18.4	2.8	**6.6**×	118.5	20.3	5.6	**3.6**×	197.0	34.5	5.1	**6.8**×
	n	1	1	15	-	1	1	15	-	1	1	20	-

Table 8: Main evaluation results of REMUX on three DeepSeek backbones under the hierarchical MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), average speed per query ("Time"), and the superposition count ("n"). For Token and Time, improvements over SOTA baselines are given by SOTA baselines/Ours and marked with ×.

Tasks	Metrics	Ministral-3-8B				Mistral-Nemo-12B				Moonlight-16B-A3B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	95.3	95.5	95.0	0.5↓	88.3	88.2	88.2	0.1↓	97.2	96.8	96.8	0.4↓
	Token	4875	4382	427	**10.3**×	4302	2947	285	**10.3**×	4250	3085	293	**10.5**×
	Time	15.9	5.5	1.3	**4.2**×	4.7	2.1	0.9	**2.3**×	28.0	2.7	0.8	**3.4**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
ARC-C	Acc.	92.0	92.6	91.2	0.8↓	71.6	71.6	71.4	0.2↓	95.3	94.6	94.5	0.8↓
	Token	4872	4008	353	**11.4**×	5798	3092	296	**10.4**×	5541	3654	310	**11.8**×
	Time	15.3	4.9	2.1	**2.3**×	8.9	2.0	0.9	**2.2**×	31.0	3.2	1.0	**3.2**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
GSM8K	Acc.	89.0	89.0	88.6	0.4↓	70.5	70.5	70.0	0.5↓	93.8	93.3	93.2	0.6↓
	Token	3771	3510	326	**10.8**×	6470	3463	325	**10.7**×	6027	4183	401	**10.4**×
	Time	20.3	4.3	1.2	**3.6**×	8.5	2.2	0.9	**2.4**×	19.4	3.6	1.0	**3.6**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
MedQA	Acc.	80.0	80.2	79.6	0.6↓	65.0	64.8	64.7	0.3↓	77.9	77.6	77.4	0.5↓
	Token	10824	4930	384	**12.8**×	13888	4950	378	**13.1**×	13364	5592	543	**10.3**×
	Time	21.0	6.3	1.8	**3.5**×	59.3	2.5	1.0	**2.5**×	48.3	4.4	1.5	**2.9**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
MBPP+	Acc.	63.3	63.3	62.8	0.5↓	39.0	39.0	38.5	0.5↓	71.6	71.2	71.0	0.6↓
	Token	5617	4862	419	**11.6**×	13475	5804	438	**13.3**×	13047	4586	426	**10.8**×
	Time	28.3	6.0	2.1	**2.9**×	27.2	3.3	1.7	**1.9**×	36.8	5.2	1.1	**4.7**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
HumanEval+	Acc.	78.1	78.0	77.5	0.6↓	61.5	61.5	61.2	0.3↓	80.6	80.6	80.3	0.3↓
	Token	4880	3508	331	**10.6**×	13614	3495	328	**10.7**×	13096	3977	363	**11.0**×
	Time	46.0	4.3	2.7	**1.6**×	34.7	2.4	0.6	**4.0**×	39.5	4.9	2.5	**2.0**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
GPQA	Acc.	43.5	43.5	43.0	0.5↓	29.5	29.5	29.2	0.3↓	45.6	45.5	45.4	0.2↓
	Token	23469	8549	698	**12.2**×	26138	10495	674	**15.6**×	26841	16640	1250	**13.3**×
	Time	233.4	27.6	2.3	**12.0**×	154.2	11.8	1.9	**6.2**×	167.1	26.7	5.6	**4.8**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
AIME25	Acc.	45.0	45.0	44.7	0.3↓	37.8	37.8	37.6	0.2↓	48.5	48.6	48.4	0.2↓
	Token	34598	10754	714	**15.1**×	36507	11642	729	**16.0**×	30564	16980	1204	**14.1**×
	Time	206.2	36.8	2.0	**18.4**×	217.5	16.3	2.6	**6.3**×	358	29.4	6.2	**4.7**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
Average	Acc.	73.3	73.4	72.8	0.6↓	57.9	57.9	57.6	0.3↓	76.3	76.0	75.9	0.4↓
	Token	11613	5563	457	**12.2**×	15024	5736	432	**13.3**×	14091	7337	599	**12.3**×
	Time	73.3	12.0	1.9	**6.3**×	64.4	5.3	1.3	**4.1**×	91.0	10.0	2.5	**4.1**×
	n	1	1	15	-	1	1	15	-	1	1	15	-

Table 9: Main evaluation results of REMUX with Ministral-3-8B, Mistral-Nemo-12B, and Moonlight-16B-A3B under the sequential MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average time consumption per query ("Time"), and the superposition count ("n"). For Token and Time, improvements over SOTA baselines are given by SOTA baselines/Ours and marked with ×.

Tasks	Metrics	Ministral-3-8B				Mistral-Nemo-12B				Moonlight-16B-A3B			
		TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**	TextMAS	LatentMAS	Ours	**Improve**
ARC-E	Acc.	93.6	93.6	93.0	0.6↓	88.3	88.3	88.2	0.1↓	97.2	97.2	97.8	0.4↓
	Token	6281	4975	432	**11.5**×	5476	3162	294	**10.8**×	4705	3178	298	**10.7**×
	Time	21.5	6.8	2.9	**2.3**×	5.0	2.3	0.9	**2.6**×	34.6	3.0	0.8	**3.8**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
ARC-C	Acc.	90.7	91.0	90.7	0.3↓	71.8	71.8	71.5	0.3↓	96.0	95.8	95.5	0.5↓
	Token	6025	4759	368	**12.9**×	5944	3140	306	**10.2**×	6235	3796	324	**11.7**×
	Time	20.3	5.8	2.2	**2.6**×	9.4	2.2	0.9	**2.4**×	39.1	3.8	1.0	**3.8**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
GSM8K	Acc.	87.3	87.5	87.1	0.4↓	70.5	70.5	70.2	0.3↓	93.0	93.0	92.6	0.4↓
	Token	6584	4316	403	**10.7**×	6250	3394	319	**10.6**×	6936	4235	410	**10.3**×
	Time	35.6	5.7	2.7	**2.1**×	8.3	2.1	0.8	**2.6**×	25.1	4.0	1.0	**4.0**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
MedQA	Acc.	77.2	77.2	76.5	0.7↓	66.0	65.7	65.6	0.4↓	76.5	77.2	77.2	0.3↓
	Token	13595	5254	410	**12.8**×	15423	5068	331	**15.3**×	15965	5871	550	**10.8**×
	Time	34.2	7.2	2.7	**2.7**×	70.4	2.8	1.0	**2.8**×	62.7	5.2	1.5	**3.5**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
MBPP+	Acc.	63.0	63.0	62.6	0.4↓	39.0	39.0	38.7	0.3↓	70.5	70.5	70.2	0.3↓
	Token	6451	4988	413	**12.1**×	14158	6027	442	**13.6**×	16493	4975	440	**11.3**×
	Time	22.6	6.9	2.7	**2.6**×	28.8	4.5	1.8	**2.5**×	42.4	7.0	1.2	**5.8**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
HumanEval+	Acc.	77.5	77.0	76.8	0.7↓	61.7	61.7	61.2	0.5↓	79.5	79.8	79.5	0.3↓
	Token	6042	3745	349	**10.7**×	14753	3654	330	**11.1**×	15854	4976	372	**13.4**×
	Time	20.4	4.3	2.0	**2.2**×	38.6	3.0	0.6	**5.0**×	45.8	6.9	2.5	**2.8**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
GPQA	Acc.	42.0	42.0	41.8	0.2↓	29.0	29.0	28.5	0.5↓	42.3	42.0	42.0	0.3↓
	Token	29754	9018	792	**11.4**×	28746	11962	684	**17.5**×	31558	18075	1283	**14.1**×
	Time	264.1	31.3	4.6	**6.8**×	165.3	12.9	2.0	**6.5**×	195.0	31.7	5.9	**5.4**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
AIME25	Acc.	43.6	44.0	43.6	0.4↓	37.8	37.8	37.5	0.3↓	45.2	45.5	45.0	0.5↓
	Token	38745	11845	729	**16.2**×	38904	12586	730	**17.2**×	34595	17458	1292	**13.5**×
	Time	239.0	42.6	4.0	**10.7**×	223.7	18.1	2.6	**7.0**×	405.2	31.6	6.5	**4.9**×
	n	1	1	15	-	1	1	15	-	1	1	15	-
Average	Acc.	71.9	71.9	71.5	0.4↓	58.0	58.0	57.7	0.3↓	75.0	75.1	75.0	0.1↓
	Token	13311	5611	487	**11.5**×	16207	6124	430	**14.2**×	16543	7821	621	**12.6**×
	Time	54.8	13.8	3.0	**4.6**×	68.7	6.0	1.3	**4.6**×	106.2	11.7	2.6	**4.5**×
	n	1	1	15	-	1	1	15	-	1	1	15	-

Table 10: Main evaluation results of REMUX with Ministral-3-8B, Mistral-Nemo-12B, and Moonlight-16B-A3B under the hierarchical MAS setting. We report four metrics in total: task accuracy (%, "Acc."), average token consumption per query ("Token"), average speed per query ("Time"), and the superposition count ("n"). For Token and Time, improvements over SOTA baselines are given by SOTA baselines/Ours and marked with $\times$.

G PROMPT TEMPLATES FOR BASELINES AND REMUX

This section provides the complete prompt templates used in our experiments. The templates are organized by collaboration architecture (sequential and hierarchical MAS). REMUX shares the same prompts with the *LatentMAS* baseline under both architectures. The *TextMAS* baseline uses its own text-mediated prompts under both architectures, and the *Single-Agent* baseline uses a single unified prompt. The answer format is customized per dataset group, covering numerical answers, A-to-D option selection, and Python code blocks.

Sequential Architecture Prompts Shared by Baseline LatentMAS and Our Method

System prompt: Under no circumstances should you output any tag starting with to wrap your reasoning. All logical calculations and steps need to be written within the main response content directly. You are prohibited from splitting thinking content into an independent tagged block. Only provide brief steps and final computed answer.

Planner: Given an input question, design a clear, step-by-step plan for how to solve the 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: 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:(i) original plan contents. (ii) 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: You are a Refiner Agent to provide a refined step-by-step plan for solving the given question. You are provided with: (i) latent-format information: a previous plan with feedback. (ii) 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:

Judger:

- Group 1 (GSM8K, AIME25). Target Question: *question*. You are the final solver agent in a sequential multi-agent system. You are provided with the Refiner Agent's plan as reference. The plan might contain irrelevant or incorrect contents. Ignore them if they are 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*.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). 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. Your final answer must be selected from A,B,C,D. Now, reason step by step and output the final answer inside *YOUR_FINAL_ANSWER*.
- Group3 (MBPP+, HumanEval+). 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. You must put all python code as self-contained Python function in markdown code blocks. For example python import math def add(a, b): return a + b. Do not add any other contents inside the markdown code block. Now, reason step by step and output the final answer inside python *YOUR_PYTHON_CODE*.

Hierarchical Architecture Prompts Shared by Baseline LatentMAS and Our Method

System prompt: You are an analytical problem-solving assistant.

Math Agent: You are a math agent.

- Group 1 (GSM8K, AIME24, AIME25). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*. Your final answer must be selected from A,B,C,D.
- Group 3 (MBPP+, HumanEval+). Given the input question, reason step by step: please provide an efficient and self-contained Python function that solves the following problem in a markdown code block. You must put all python code as self-contained Python function in markdown code blocks.

Science Agent: You are a science agent.

- Group 1 (GSM8K, AIME24, AIME25). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*. Your final answer must be selected from A,B,C,D.
- Group 3 (MBPP+, HumanEval+). Given the input question, reason step by step: please provide an efficient and self-contained Python function that solves the following problem in a markdown code block. You must put all python code as self-contained Python function in markdown code blocks.

Code Agent: You are a code agent.

- Group 1 (GSM8K, AIME24, AIME25). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Given the input question, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*. Your final answer must be selected from A,B,C,D.
- Group 3 (MBPP+, HumanEval+). Given the input question, reason step by step: please provide an efficient and self-contained Python function that solves the following problem in a markdown code block. You must put all python code as self-contained Python function in markdown code blocks.

Task Summarizer Agent: You are a task summarizer.

- Group 1 (GSM8K, AIME24, AIME25). Given the input question and responses from previous agents as reference, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Given the input question and responses from previous agents as reference, reason step-by-step and put the final answer inside *YOUR_FINAL_ANSWER*. Your final answer must be selected from A,B,C,D. Do not add any other contents inside the box.
- Group 3 (MBPP+, HumanEval+). Given the input question and responses from previous agents as reference, reason step by step: please provide an efficient and self-contained Python function that solves the following problem in a markdown code block. You must put all python code as self-contained Python function in markdown code blocks. For example python import needed_library def FUNC_NAME(a, b): return a + b. Do not add any other contents inside the markdown code block.

Sequential TextMAS Prompts

System prompt: Under no circumstances should you output any tag starting with to wrap your reasoning. All logical calculations and steps need to be written within the main response content directly. You are prohibited from splitting thinking content into an independent tagged block.

Planner: You are a Planner Agent. Given an input question, design a clear plan for how to solve the question. Important requirement: Produce a very detailed, fully-expanded plan. Do NOT produce the final answer.

Critic: You are a Critic Agent. You are provided with: (i) the original question, and (ii) the Planner Agent's plan in text format. Your job is to carefully evaluate the correctness and completeness of the plan and provide helpful feedback.

Refiner: You are a Refiner Agent. You are provided with: (i) the original question, and (ii) the Planner Agent's plan together with Critic Agent's feedback in text format. Your job is to incorporate the feedback and produce an improved, refined plan.

Judger:

- Group 1 (GSM8K, AIME25). Target Question: *question*. You are the final solver agent in a sequential multi-agent system (planner → critic → refiner → solver). You are provided with the Refiner Agent's plan as reference. Refined Plan from Previous Agents: *context*. The plan might contain irrelevant or incorrect contents. Ignore them if they are 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.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Target Question: *question*. You are the final solver agent in a sequential multi-agent system (planner → critic → refiner → solver). You are provided with the Refiner Agent's plan as reference. Refined Plan from Previous Agents: *context*. The plan might contain irrelevant or incorrect contents. Ignore them if they are not helpful for solving the target question. 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. Do not add any other contents inside the box. Now, reason step by step and output the final answer inside YOUR_FINAL_ANSWER.
- Group 3 (MBPP+, HumanEval+). Target Question: *question*. You are the final solver agent in a sequential multi-agent system (planner → critic → refiner → solver). You are provided with the Refiner Agent's plan as reference. Refined Plan from Previous Agents: *context*. The plan might contain irrelevant or incorrect contents. Ignore them if they are not helpful for solving the target question. You must reason step-by-step to solve the *provided Target Question* without outputting other irrelevant information. You must put all python code as self-contained Python function(s) in markdown code blocks. Do not add any other contents inside the markdown code block.

Hierarchical TextMAS Prompts

System prompt: You are an analytical problem-solving assistant.

Math Agent:You are a math agent.

- Group 1 (GSM8K, AIME25, ARC-E, ARC-C, GPQA, MedQA). Input Question: *question*. Reason and put the final answer inside YOUR_FINAL_ANSWER.
- Group 2 (MBPP+, HumanEval+). Input Question: *question*. You must put all python code as self-contained Python function in markdown code blocks. Do not add any other contents inside the markdown code block.

Science Agent:You are a science agent.

- Group 1 (GSM8K, AIME25, ARC-E, ARC-C, GPQA, MedQA). Input Question: *question*. Reason and put the final answer inside YOUR_FINAL_ANSWER.
- Group 2 (MBPP+, HumanEval+). Input Question: *question*. You must put all python code as self-contained Python function in markdown code blocks. Do not add any other contents inside the markdown code block.

Code Agent:You are a code agent.

- Group 1 (GSM8K, AIME25, ARC-E, ARC-C, GPQA, MedQA). Input Question: *question*. Reason and put the final answer inside YOUR_FINAL_ANSWER.
- Group 2 (MBPP+, HumanEval+). Input Question: *question*. You must put all python code as self-contained Python function in markdown code blocks. Do not add any other contents inside the markdown code block.

Task Summarizer Agent:You are a task summarizer.

- Group 1 (GSM8K, AIME25, ARC-E, ARC-C, GPQA, MedQA). Input Question: *question*. Content from Previous Agent: *context*. Reason and put the final answer inside YOUR_FINAL_ANSWER.
- Group 2 (MBPP+, HumanEval+). Input Question: *question*. Content from Previous Agent: *context*. Given the final answer in markdown python code block.

Single Agent Baseline Prompts

System prompt: You are an analytical problem-solving assistant.

Agent:

- Group 1 (GSM8K, AIME2025). Target Question: *question*. You must reason step-by-step to solve the *provided Target Question* without outputting other irrelevant information. Do not output any irrelevant information. Output the final answer inside YOUR_FINAL_ANSWER. Now, reason step by step and output the final answer inside YOUR_FINAL_ANSWER.
- Group 2 (ARC-E, ARC-C, GPQA, MedQA). Target Question: *question*. Solve the *provided Target Question* without outputting other irrelevant information. Your final answer must be selected from A,B,C,D. Do not add any other contents inside the box. Output the final answer inside YOUR_FINAL_ANSWER.
- Group 3 (MBPP+, HumanEval+). Target Question: *question*. You must put all python code as self-contained Python function(s) in markdown code blocks. For example
  ```python
  import math
  def add(a, b):
  return a + b```. Do not add any other contents inside the markdown code block. Now, reason step by step and output the final answer.
  ```