CondenseFlow: Scalable Latent Space Collaboration via Semantic Compression for Multi-Agent Systems

Xiaoyu Chen[1,2,3], Fengge Wu[1,2,3,†], Junsuo Zhao[1,2,3], Yun Fan[3]

[1]University of Chinese Academy of Sciences
[2]Institute of Software, Chinese Academy of Sciences
[3]Science & Technology on Integrated Information System Laboratory
{chenxiaoyu2025, junsuo}@iscas.ac.cn; fanyun_cn@126.com
fengge@iscas.ac.cn

Abstract

Full-state latent communication in LLM-based multi-agent systems offers richer semantics than text but suffers from memory overhead scaling linearly with collaboration rounds. We propose **CondenseFlow**, which introduces the **Latent Thought Condenser (LTC)**—a lightweight module using learnable semantic probes to compress KV caches into fixed-size representations, achieving $\mathcal{O}(1)$ communication complexity regardless of context length. We theoretically prove that compression error is bounded by attention concentration and accumulates controllably across rounds. On seven benchmarks spanning six models, CondenseFlow reduces KV cache memory by over 99% and inference latency by approximately 20% compared to dense transfer with negligible accuracy degradation, while outperforming text-based methods by 1.7 percentage points on average across all configurations. Code is available at `https://github.com/xxy33/condenseflow`.

1 Introduction

Large Language Models (LLMs) are evolving from text generation tools into autonomous agents capable of perception, planning, and action (Xi et al., 2025; Wang et al., 2024; Guo et al., 2024). Pioneering works such as ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023), and Voyager (Wang et al., 2023a) have demonstrated the potential of LLM agents in interactive decision-making and open-world exploration. Building upon single-agent successes, the field has naturally progressed toward **multi-agent collaboration**: frameworks like MetaGPT (Hong et al., 2023), ChatDev (Qian et al., 2024), and AutoGen (Wu et al., 2024) achieve complex task completion through role-based division of labor (Talebirad and Nadiri, 2023; Zhang et al., 2025).

Current multi-agent systems predominantly adopt **natural language** as the communication medium (Guo et al., 2024; Zhuge et al., 2024). However, the discrete nature of text constitutes an inherent **information bottleneck** (Bisk et al., 2020; Andreas, 2022). Compressing continuous internal reasoning states into discrete token sequences inevitably discards uncertainty estimates and fine-grained semantic nuances (Wong et al., 2023; Piantadosi et al., 2024). To transcend this limitation, recent research has begun exploring **latent space communication**, where agents directly transmit Transformer internal representations for higher-fidelity information exchange (Hao et al., 2024; Fu et al., 2024).

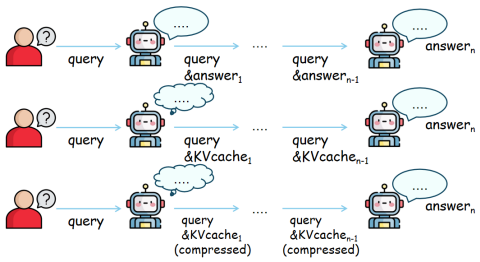

Figure 1: Comparison of three inter-agent communication paradigms in multi-round collaboration.

However, latent communication faces a severe "Memory Wall" problem (Pope et al., 2023; Kwon et al., 2023). Under full-state transmission, the memory footprint grows linearly with the number of layers, hidden dimensions, and sequence length, following $\mathcal{O}(R \cdot T \cdot L \cdot d_h)$ over R collaboration rounds. For a 14B parameter model, a single transfer of working memory can reach several gigabytes. While such overhead might be momentarily manageable on high-end data center GPUs, this linear accumulation imposes prohibitive latency penalties and creates a substantial barrier for deployment on consumer-grade hardware or in bandwidth-constrained environments (Sheng et al., 2023; Aminabadi et al., 2022). This contradiction poses the core challenge: *How can we preserve*

Findings of the Association for Computational Linguistics: ACL 2026, pages 13694–13712

the high-fidelity advantages of latent communication while strictly bounding resource consumption regardless of interaction depth?

We propose **CondenseFlow**, a resource-aware framework for scalable latent multi-agent collaboration. The framework centers on the **Latent Thought Condenser**, a lightweight module that aggregates critical information from complete working memory through learnable semantic probes. By compressing variable-length KV caches into fixed-size representations of dimension K, LTC achieves over 99% memory reduction while preserving information essential for downstream reasoning. Unlike heuristic pruning methods (Zhang et al., 2023; Xiao et al., 2023), LTC discovers information patterns most valuable for collaboration through end-to-end learning. Figure 1 illustrates the three communication paradigms: TextMAS transmits discrete token sequences, Dense-Latent transfers complete KV caches, and CondenseFlow passes fixed-size compressed representations.

Our main contributions are threefold. First, we formalize efficient latent communication as a semantic compression problem and establish a theoretical framework for analyzing compression error bounds under low-rank assumptions. Second, we propose the cross-attention-based LTC module that performs lightweight yet effective semantic compression with minimal parameter overhead. Third, we validate our approach across 7 benchmarks and 6 models, achieving over 99% memory reduction with less than 2% accuracy degradation compared to dense latent baselines.

2 Related Work

2.1 LLM-based Agents

The reasoning capabilities of LLMs have been significantly enhanced through Chain-of-Thought (CoT) prompting techniques (Wei et al., 2022; Kojima et al., 2022). Self-Consistency (Wang et al., 2022), Tree of Thoughts (Yao et al., 2023), and Graph of Thoughts (Besta et al., 2024) extend this paradigm from diverse perspectives. To overcome inherent LLM limitations, Toolformer (Schick et al., 2023), HuggingGPT (Shen et al., 2023), WebGPT (Nakano et al., 2021), and ToolLLM (Qin et al., 2023) integrate external tools; ReAct (Yao et al., 2022), DEPS (Wang et al., 2023c), and RAP (Hao et al., 2023) implement interactive reasoning-action cycles. In embodied intelligence, SayCan (Ahn et al., 2022), PaLM-E (Driess et al., 2023), and Voyager (Wang et al., 2023a) demonstrate the potential of language models for driving robots and game agents. Generative Agents (Park et al., 2023), LangChain (Chase et al., 2023), and cognitive architectures (Sumers et al., 2023) further advance complete agent frameworks. Comprehensive benchmarks such as AgentBench (Liu et al., 2023b) and MINT (Wang et al., 2023b) have been established for systematic evaluation. These works primarily focus on single-agent scenarios.

2.2 Multi-Agent Systems

Traditional multi-agent reinforcement learning has proposed value decomposition methods including VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2020), QTRAN (Son et al., 2019), and MAVEN (Mahajan et al., 2019), along with various communication protocol learning approaches (Foerster et al., 2016; Sukhbaatar et al., 2016; Das et al., 2019; Jiang and Lu, 2018). However, these methods exhibit limited policy generalization (Yu et al., 2022).

Text-based multi-agent systems represent the current mainstream. CAMEL (Li et al., 2023) achieves dual-agent collaboration through role-playing; ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2023) map software development roles to LLM agents; AutoGen (Wu et al., 2024) provides a flexible conversation framework; AgentVerse (Chen et al., 2024) supports dynamic team formation; DyLAN (Liu et al., 2023d) enables dynamic agent organization. Multi-agent debate has been shown to improve reasoning quality (Du et al., 2023; Liang et al., 2024; Chan et al., 2023; Xiong et al., 2023). ProAgent (Zhang et al., 2024a) and AgentCF (Zhang et al., 2024b) explore collaboration in specific domains. These systems validate the collaborative potential of LLMs but remain constrained by the discrete nature of text communication (PINE, 2005).

Latent space communication represents an emerging direction. CoCoNut (Hao et al., 2024) and related works (Mei et al., 2024) explore continuous thought reasoning within single models; CALM (Bansal et al., 2024) studies cross-model latent augmentation; some research attempts to use KV caches as cross-model information carriers (Fu et al., 2024; Liu et al., 2024a; Cai et al., 2024a). Recent concurrent works further explore this direction: C2C (Fu et al., 2025) trains a neural Cache Fuser for KV-level transfer across heterogeneous models; ThoughtComm (Zheng et al., 2025) com-

presses hidden states into fixed-length prefix tokens via a sparsity-regularized autoencoder but discards per-layer structure; LatentMAS (Zou et al., 2025) proposes training-free full KV cache transfer. However, full-state transmission strategies face severe memory bottlenecks in multi-round scenarios. This paper focuses on addressing this efficiency problem through learning-based semantic compression for scalable latent collaboration.

2.3 KV Cache Compression

The KV cache grows linearly with sequence length, becoming a bottleneck for long-context processing (Kwon et al., 2023). **Heuristic selection methods** such as H2O (Zhang et al., 2023) retain important tokens based on accumulated attention scores; Scissorhands (Liu et al., 2023c) performs dynamic pruning using attention persistence; SnapKV (Li et al., 2024) identifies critical attention patterns; StreamingLLM (Xiao et al., 2023) supports streaming generation through attention sink phenomena; FastGen (Ge et al., 2023a), KIVI (Liu et al., 2024b), and PyramidKV (Cai et al., 2024b) propose various efficient compression schemes. **Learned compression methods** such as Token Merging (Bolya et al., 2022) combine similar tokens; Gist Tokens (Mu et al., 2023) compress long contexts into essential tokens; AutoCompressor (Chevalier et al., 2023) and ICAE (Ge et al., 2023b) explore learnable memory compression; LLMLingua (Jiang et al., 2023; Pan et al., 2024) achieves prompt compression.

These methods target **single-model inference acceleration**, optimizing for output token accuracy. Our LTC serves **cross-agent semantic communication**, aiming to preserve high-level intent and thought processes required for downstream reasoning. This distinction necessitates a semantics-aware learned compression strategy rather than purely statistical approaches.

3 Methodology

3.1 Problem Formulation

We consider a multi-agent system $\mathcal{S} = \{A_1, A_2, \ldots, A_N\}$ where each agent is instantiated by a Transformer-based language model with L layers and hidden dimension d_h.

Latent Working Memory. Under the latent collaboration paradigm, agents exchange internal representations rather than discrete text. After agent A_{i-1} completes reasoning, its working memory is the key-value cache across all layers:

$$\mathcal{M}_{full} = \{(K^{(l)}, V^{(l)})\}_{l=1}^{L}, \quad K^{(l)}, V^{(l)} \in \mathbb{R}^{T \times d_h} \tag{1}$$

where T denotes the sequence length.

The Scalability Challenge of Full-State Transfer. Transmitting complete working memory poses a critical efficiency challenge: memory footprint grows as $\mathcal{O}(R \cdot T \cdot L \cdot d_h)$ over R collaboration rounds, rapidly exhausting hardware resources. Furthermore, empirical evidence suggests that accumulated context progressively degrades reasoning quality as models struggle to maintain focus on task-relevant information. Our experiments in Section 4.3 confirm this phenomenon.

Design Objective. We aim to learn a compression mapping ϕ_θ that transforms full working memory into a fixed-size semantic summary:

$$\mathcal{M}_{gist} = \{(\tilde{K}^{(l)}, \tilde{V}^{(l)})\}_{l=1}^{L}, \quad \tilde{K}^{(l)}, \tilde{V}^{(l)} \in \mathbb{R}^{K \times d_h} \tag{2}$$

where $K \ll T$ is a fixed constant. This design achieves $\mathcal{O}(1)$ communication complexity per round, simultaneously addressing memory scaling and context accumulation.

3.2 Latent Thought Condenser

The Latent Thought Condenser extracts semantic anchors from complete KV caches through learnable probes.

Learnable Semantic Probes. The core component is a set of learnable probe vectors:

$$Q_c = [q_1, q_2, \ldots, q_K]^\top \in \mathbb{R}^{K \times d_h} \tag{3}$$

Each probe functions as a semantic interrogator that detects and aggregates critical information along a specific dimension from the upstream thought stream. Unlike heuristic methods relying on attention score statistics, semantic probes discover valuable information patterns through end-to-end learning.

Cross-Attention Aggregation. For the cache $(K^{(l)}, V^{(l)})$ at layer l, probes compute semantic relevance with key vectors:

$$A^{(l)} = \text{softmax}\left(\frac{Q_c (K^{(l)})^\top}{\sqrt{d_h}}\right) \in \mathbb{R}^{K \times T} \tag{4}$$

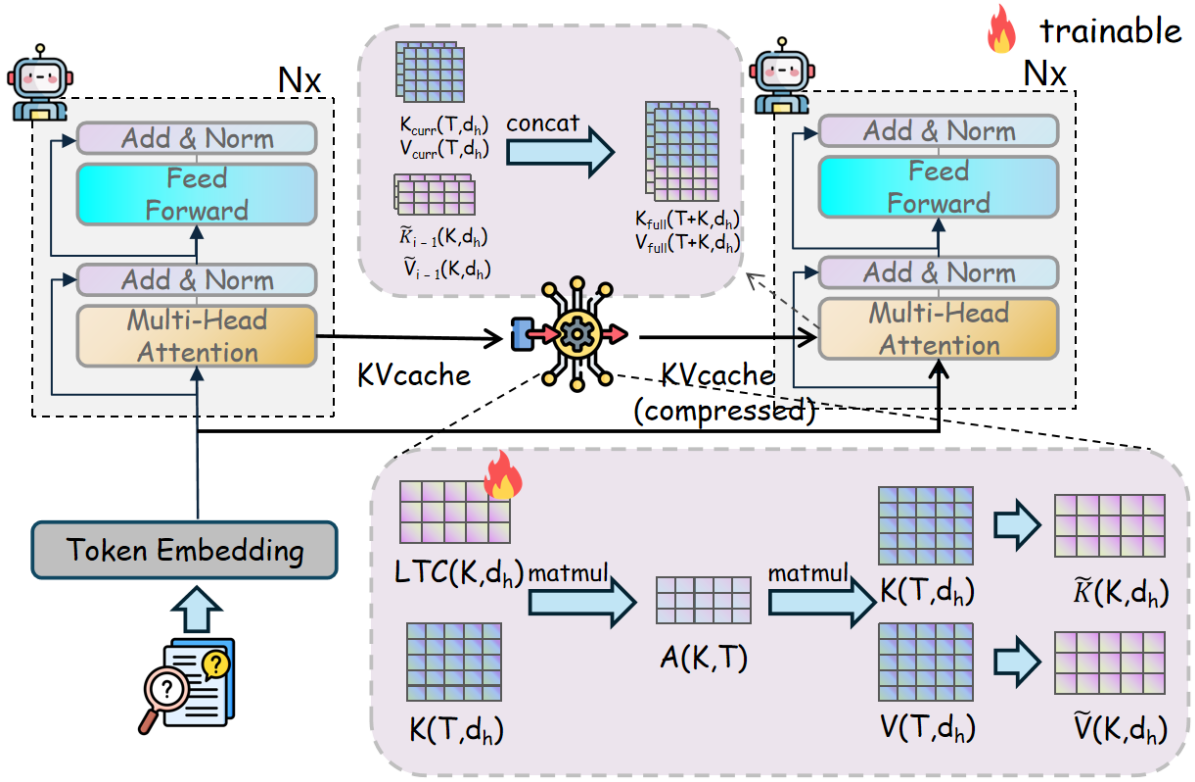

Figure 2: Overview of CondenseFlow. Each agent compresses its working memory through the Latent Thought Condenser before transmission, achieving constant-size communication regardless of context length.

The aggregation weights then compress keys and values:

$$\tilde{K}^{(l)} = A^{(l)}K^{(l)}, \quad \tilde{V}^{(l)} = A^{(l)}V^{(l)} \in \mathbb{R}^{K \times d_h} \tag{5}$$

Each output vector is a weighted semantic summary, with different probes learning to focus on complementary aspects such as numerical values, logical connectives, or structural markers.

Architecture Design. We adopt cross-layer parameter sharing where all L layers use identical probes, yielding minimal parameter overhead. Layer normalization is applied to stabilize compressed representations. Detailed architectural specifications are provided in Appendix B.

3.3 Training Objective

LTC is trained to make compressed KV caches functionally equivalent to original caches for downstream attention computation.

Reconstruction Loss. The core objective requires attention outputs computed with compressed and full caches to match:

$$\mathcal{L}_{recon} = \frac{1}{L}\sum_{l=1}^{L} \big\| \mathrm{Attn}(Q_s, K^{(l)}, V^{(l)}) - \mathrm{Attn}(Q_s, \tilde{K}^{(l)}, \tilde{V}^{(l)}) \big\|_F^2 \tag{6}$$

where Q_s denotes sampled query matrices.

Regularization. Two regularization terms prevent degenerate solutions. Coverage regularization maximizes the entropy of average attention distribution to ensure probes uniformly cover the original sequence. Orthogonality regularization constrains probe pairwise similarity to encourage complementary feature capture. The complete loss formulation and hyperparameter settings are detailed in Appendix C.

3.4 Theoretical Intuition

We provide theoretical intuition for why semantic compression is effective in multi-agent collaboration. Formal analysis appears in Appendix A.

Compression Error Bound. The error introduced by LTC compression is governed by the concentration of the original attention distribution.

When attention naturally focuses on a subset of positions, compression that preserves these attended positions incurs minimal information loss. We formalize this intuition in Theorem A.3, establishing that compression error scales with $(1-\rho)$ where ρ measures the fraction of attention mass captured by the top-K positions.

Empirical Compressibility. LLM working memories exhibit low effective rank in practice, indicating that high-quality low-dimensional representations exist. Our analysis in Appendix E confirms that typical KV caches have effective rank well below our default compression dimension, validating that compression preserves the dominant semantic subspace.

Multi-Round Behavior. Under bounded per-round compression error δ, cumulative error grows at most linearly as $R \cdot \delta$ over R rounds. Empirically, we observe sub-linear growth as LTC learns to prioritize information most relevant for downstream reasoning, with experimental validation in Section 4.3.

3.5 Collaboration Pipeline

Cache Integration. Each agent receives the original question q and the compressed cache $\mathcal{M}_{i-1}$ from the previous round. Each round **replaces** rather than appends the compressed representation, maintaining constant context overhead. At each Transformer layer, the compressed history is prepended as a KV prefix:

$$K_{full} = [\tilde{K}_{i-1}; K_{curr}], \quad V_{full} = [\tilde{V}_{i-1}; V_{curr}] \tag{7}$$

where K_{curr}, V_{curr} derive from encoding q and subsequent generation. This ensures agents attend to condensed history while processing new content, with context overhead bounded at $\mathcal{O}(K)$ regardless of collaboration depth.

Information Preservation. Although each round replaces the previous compressed representation, historical information propagates through iterative distillation. When agent A_i processes its input, the resulting working memory encodes both current reasoning and attended information from $\mathcal{M}_{i-1}$. Subsequent compression distills this enriched representation, implicitly preserving semantic essence from all preceding rounds while preventing unbounded context growth.

Algorithm 1 CondenseFlow Collaboration

Require: Question q, Agents $\{A_1, \ldots, A_N\}$, LTC ϕ_θ
Ensure: Answer a

$\mathcal{M}_{prev} \leftarrow \varnothing$
for $i = 1$ to $N-1$ **do**
  $\mathcal{M}_{full} \leftarrow A_i.\text{Reason}(q, \mathcal{M}_{prev})$
  $\mathcal{M}_{prev} \leftarrow \phi_\theta(\mathcal{M}_{full})$
end for
$a \leftarrow A_N.\text{Decode}(q, \mathcal{M}_{prev})$
return a

Compression Dimension Selection. The compression dimension K governs the trade-off between information retention and efficiency. Based on effective rank analysis showing typical KV caches have effective rank below 50, we select K=64 as default. Systematic ablation in Section 4.4 validates this configuration achieves optimal efficiency-effectiveness balance with diminishing returns beyond this point.

End-to-End Protocol. Algorithm 1 presents the complete collaboration pipeline. Each agent performs reasoning conditioned on compressed history, applies LTC compression, and passes the fixed-size representation to the next agent. Only the final agent decodes into text output.

4 Experiments

We evaluate CondenseFlow through two complementary protocols: standard collaboration assessing compression fidelity, and stress testing examining robustness under extended interaction.

4.1 Experimental Setup

Benchmarks. We employ seven benchmarks spanning three reasoning categories. Mathematical reasoning includes AIME 2024 (Zhang and Math-AI, 2024), AIME 2025 (Zhang and Math-AI, 2025), and HMMT 2025 (FlagEval, 2025). Scientific reasoning includes GPQA-Diamond (Rein et al., 2024) and MedQA (Jin et al., 2021). Code generation includes MBPP-Plus (Liu et al., 2023a) and LiveCodeBench (Jain et al., 2024).

Models. Small-scale models include Qwen3-8B-Instruct (Yang et al., 2025), LLaMA-3.1-8B-Instruct (Dubey et al., 2024), and Gemma-2-9b-it (Team et al., 2024). Mid-scale models include Qwen3-14B (Yang et al., 2025), DeepSeek-R1-

Table 1: Accuracy on small-scale models under Standard Protocol. Results are mean ± std over 5 runs. **Bold** indicates best between Dense and Ours. Δ denotes accuracy gap between CondenseFlow and Dense-Latent.

	Qwen3-8B-Instruct			**LLaMA-3.1-8B-Instruct**			**Gemma-2-9b-it**		
Task	Text	Dense	Ours	Text	Dense	Ours	Text	Dense	Ours
AIME 2024	50.1 ±1.2	**52.4** ±1.1	49.8 ±1.3	48.5 ±1.4	**51.2** ±1.0	48.4 ±1.5	51.2 ±1.1	**53.5** ±0.9	51.8 ±1.2
AIME 2025	49.2 ±1.5	**53.1** ±1.2	51.2 ±1.4	47.6 ±1.3	**51.8** ±1.1	49.8 ±1.2	50.1 ±1.0	**54.2** ±0.8	52.9 ±1.3
HMMT 2025	17.1 ±0.9	**18.4** ±0.8	16.5 ±1.0	16.2 ±1.1	**17.5** ±0.9	15.6 ±1.1	18.5 ±0.8	**19.8** ±0.7	18.5 ±1.0
MBPP+	68.2 ±1.1	72.5 ±0.9	**72.8** ±1.2	66.5 ±1.3	70.8 ±1.0	**71.0** ±1.4	69.1 ±1.1	73.1 ±0.9	**73.8** ±1.2
GPQA-Diamond	42.1 ±1.6	45.3 ±1.4	**45.5** ±1.5	40.8 ±1.8	43.9 ±1.5	**44.0** ±1.7	43.2 ±1.3	46.5 ±1.2	**46.9** ±1.4
MedQA	70.2 ±0.8	**72.8** ±0.7	69.6 ±0.9	68.5 ±1.0	**71.2** ±0.9	68.0 ±1.1	71.5 ±0.8	**73.5** ±0.7	70.5 ±1.0
LiveCodeBench	31.2 ±1.4	**33.6** ±1.2	30.8 ±1.5	29.8 ±1.6	**32.5** ±1.3	29.5 ±1.5	32.1 ±1.2	**34.8** ±1.1	31.9 ±1.3
Average	46.9	**49.7**	48.0	45.4	**48.4**	46.6	48.0	**50.8**	49.5
Δ		-1.7%			-1.8%			-1.3%	

Table 2: Accuracy on mid-scale models under Standard Protocol. Results are mean ± std over 5 runs. **Bold** indicates best between Dense and Ours. Δ denotes accuracy gap between CondenseFlow and Dense-Latent.

	Qwen3-14B			**DeepSeek-R1-Distill-Qwen-14B**			**Ring-mini-2.0**		
Task	Text	Dense	Ours	Text	Dense	Ours	Text	Dense	Ours
AIME 2024	60.5 ±1.1	**65.2** ±0.9	63.2 ±1.0	61.2 ±1.2	**66.1** ±0.8	63.9 ±1.1	59.5 ±1.0	**64.5** ±0.9	62.3 ±1.2
AIME 2025	55.4 ±1.3	**64.1** ±1.0	62.1 ±1.2	56.1 ±1.4	**64.8** ±0.9	62.6 ±1.1	54.5 ±1.3	**63.2** ±1.0	61.0 ±1.2
HMMT 2025	25.2 ±0.8	**28.3** ±0.7	26.8 ±0.9	26.5 ±0.9	**29.1** ±0.6	27.4 ±0.8	24.5 ±0.9	**27.5** ±0.7	25.8 ±0.9
MBPP+	72.1 ±0.9	75.4 ±0.7	**76.0** ±1.0	73.5 ±0.8	76.2 ±0.7	**76.7** ±0.9	71.5 ±1.0	74.8 ±0.8	**75.3** ±1.1
GPQA-Diamond	50.3 ±1.2	**51.5** ±1.1	50.3 ±1.3	51.2 ±1.3	**52.4** ±1.0	51.0 ±1.2	49.5 ±1.4	**50.8** ±1.1	49.4 ±1.3
MedQA	76.1 ±0.7	76.8 ±0.6	**77.2** ±0.8	76.8 ±0.7	77.5 ±0.6	**77.8** ±0.7	75.5 ±0.8	76.2 ±0.6	**76.5** ±0.8
LiveCodeBench	33.2 ±1.1	**35.5** ±1.0	33.5 ±1.2	34.5 ±1.2	**36.8** ±0.9	34.4 ±1.1	32.5 ±1.1	**34.9** ±1.0	32.5 ±1.2
Average	53.3	**56.7**	55.6	54.3	**57.6**	56.3	52.5	**56.0**	54.7
Δ		-1.1%			-1.3%			-1.3%	

Distill-Qwen-14B (Guo et al., 2025), and Ring-mini-2.0 (Team et al., 2025).

Baselines. We compare three paradigms: TextMAS transmits natural language; Dense-Latent transfers complete KV caches; CondenseFlow applies LTC compression.

Evaluation Protocols. We employ two protocols with distinct objectives: *Standard Protocol* uses a sequential four-agent pipeline consisting of Planner, Critic, Refiner, and Solver. This evaluates compression fidelity under typical collaboration depth. *Stress Test Protocol* employs iterative Solver-Critic interaction where the Solver generates solutions and the Critic provides feedback across multiple rounds. This evaluates robustness under extended context accumulation. We use AIME 2025 as the testbed for this challenging evaluation. Implementation details including hyperparameters and hardware specifications are provided in Appendix D.

4.2 Compression Fidelity

Tables 1 and 2 present accuracy under the standard protocol.

Compression Maintains Accuracy. Across all model-task combinations, CondenseFlow achieves accuracy within 2% of Dense-Latent despite transmitting only a fraction of the original KV cache. The gap narrows at larger scale, decreasing from 1.6% average on small models to 1.2% on mid-scale models. We attribute this to higher intrinsic redundancy in larger model representations, as validated by effective rank analysis in Appendix E.

Task-Dependent Patterns. On code generation tasks, CondenseFlow matches or exceeds Dense-Latent, suggesting that structured outputs allow effective compression of verbose intermediate reasoning. Mathematical olympiad tasks show slightly larger gaps, consistent with their need for distributed attention across numerical dependencies. Detailed per-task analysis appears in Appendix F.

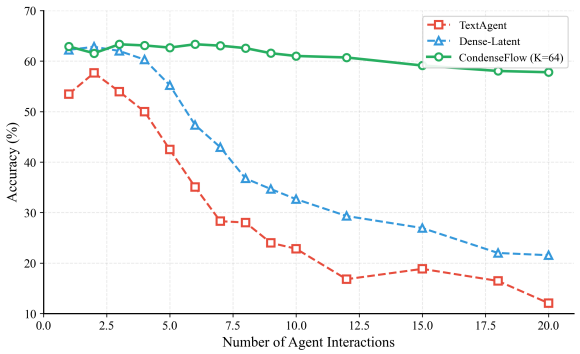

Figure 3: Accuracy on AIME 2025 under Stress Test Protocol with increasing interaction rounds using Qwen3-14B. TextMAS and Dense-Latent exhibit significant degradation after 4–5 rounds due to context accumulation, while CondenseFlow maintains stable performance throughout all twenty tested rounds.

4.3 Robustness under Extended Interaction

We examine how methods perform as interaction depth increases using the Stress Test Protocol on AIME 2025.

Accuracy Degradation Analysis. Figure 3 presents accuracy as a function of interaction rounds. TextMAS and Dense-Latent both exhibit substantial degradation beyond moderate depth, with accuracy dropping sharply after four to five rounds. TextMAS degrades from 54% at round 3 to 42% at round 5, eventually falling to 12% by round 20. Dense-Latent follows a similar pattern, declining from 63% to 21% over the same range. In contrast, CondenseFlow maintains stable performance throughout all twenty tested rounds, with accuracy remaining above 58%. This stark difference validates our hypothesis that unbounded context accumulation degrades reasoning quality, and demonstrates that fixed-size compression effectively mitigates this phenomenon.

Memory Scaling. Table 3 compares KV cache memory at different interaction depths. Dense-Latent exhibits linear growth as context accumulates across rounds, reaching over 5 GB by round 10. CondenseFlow maintains constant memory through fixed-size compression, achieving over 99% reduction regardless of interaction depth.

Inference Time. Figure 5 presents inference time measurements averaged across AIME 2025 problems. The per-round time in Figure 5(a) shows that TextMAS and Dense-Latent exhibit increasing latency as interaction progresses due to growing context length, while CondenseFlow maintains stable per-round time around 44 seconds throughout all rounds. The cumulative time in Figure 5(b) demonstrates that this efficiency advantage compounds over extended interaction: at round 10, CondenseFlow completes in 457 seconds compared to 569 seconds for Dense-Latent, achieving approximately 20% total time reduction.

Table 3: KV cache memory on Qwen3-14B at different interaction rounds. Dense-Latent grows linearly with accumulated context while CondenseFlow maintains constant size through compression.

Rounds	Dense	Ours	Reduction
3	1.72 GB	0.01 GB	99.4%
5	2.63 GB	0.01 GB	99.6%
7	3.51 GB	0.01 GB	99.7%
10	5.03 GB	0.01 GB	99.8%

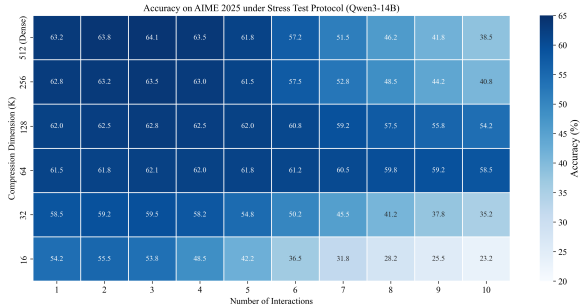

Figure 4: Accuracy heatmap on AIME 2025 showing the joint effect of compression dimension K and interaction rounds. The top row represents Dense-Latent baseline. Moderate compression (K=64) maintains robust performance across all interaction depths, while aggressive compression (K=16) shows early collapse. Increasing K beyond 64 provides diminishing returns, validating our default configuration choice.

4.4 Ablation Studies

Compression Dimension and Interaction Depth. Figure 4 presents a heatmap analyzing the joint effect of compression dimension and interaction rounds. Several patterns emerge from this analysis. Aggressive compression leads to rapid performance collapse: with K=16, accuracy drops from 54.2% at round 1 to 23.2% at round 10, a degradation of 31 percentage points. Moderate compression with K=64 maintains stability across extended interaction, with accuracy remaining above 58% through all ten rounds. Increasing compression dimension beyond K=64 yields diminishing returns at early rounds while exhibiting degradation patterns similar to Dense-Latent at later rounds, as the accumu-

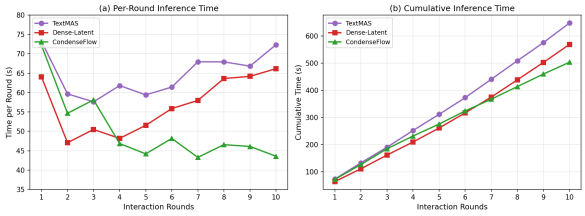

Figure 5: Inference time comparison on AIME 2025 using Qwen3-14B under Stress Test Protocol, averaged across all problems. (a) Per-round inference time shows CondenseFlow maintains stable latency while baselines increase with context accumulation. (b) Cumulative time demonstrates compounding efficiency gains over extended interaction.

Table 4: Ablation of loss components on Qwen3-14B averaged across all benchmarks.

$\mathcal{L}_{recon}$	$\mathcal{L}_{cover}$	$\mathcal{L}_{orth}$	**Avg. Acc**
✓	✓	✓	**55.6%**
✓		✓	53.8%
✓	✓		54.7%
✓			53.1%

lated representations reintroduce context growth effects. Based on these observations, we select K=64 as our default configuration, achieving optimal efficiency-effectiveness balance.

Loss Component Analysis. Table 4 examines contributions of each training objective component. Removing coverage regularization causes the largest degradation, indicating that preventing probe collapse is critical. The orthogonality constraint provides additional stability by ensuring probes capture complementary features.

Comparison with Alternative Compression. Table 5 compares against KV cache compression methods adapted for cross-agent transfer. Selection-based methods that discard entries suffer significant accuracy loss. Our aggregation approach preserves semantic information through weighted combination, achieving substantially better retention.

5 Conclusion

We present CondenseFlow, an efficient latent collaboration framework for multi-agent systems. To address the memory bottleneck caused by full KV cache transfer in existing latent communication methods, we design the Latent Thought Condenser, which extracts semantic anchors from complete KV caches through learnable probes, achieving effective compression while preserving information critical for downstream reasoning.

Table 5: Comparison with KV compression methods at equivalent compression ratio on Qwen3-14B.

Method	**Type**	**Avg. Acc**
Dense-Latent	Full	56.7%
Random Pruning	Selection	48.5%
H2O	Selection	53.2%
SnapKV	Selection	53.8%
CondenseFlow	Aggregation	**55.6%**

The main contributions of this work are threefold. On the methodological front, we propose a semantic compression mechanism based on cross-attention aggregation, where probes automatically identify and retain information patterns most valuable for collaboration, avoiding the information loss inherent in selection-based methods. On the theoretical front, we establish an error bound showing that compression quality is governed by attention concentration, providing principled understanding of when and why compression succeeds. On the empirical front, comprehensive evaluations demonstrate that CondenseFlow achieves substantial memory and time efficiency with minimal accuracy degradation across diverse reasoning tasks and model scales.

CondenseFlow enables practical deployment of latent multi-agent collaboration by breaking the linear scaling barrier of full-state transfer. As large language models become increasingly prevalent in agentic systems, efficient latent communication will become critical infrastructure for next-generation collaborative intelligence. Future directions include applying reinforcement learning to optimize compression strategies and extending to heterogeneous model collaboration.

6 Limitations

Although CondenseFlow achieves a favorable balance between efficiency and accuracy, this work has several limitations that warrant further exploration.

Training Dependency. Unlike training-free full transfer methods, the LTC module requires pretraining to function effectively. Although LTC has minimal parameters and low training cost, this adds deployment complexity. Future work could explore training-free analytical compression schemes based on low-rank approximation.

Compression Ratio Ceiling. Our experiments show that when the compression dimension K falls below 32, accuracy degrades significantly. This suggests an information-theoretic limit beyond which critical semantic information is inevitably lost. Breaking through this limitation remains an open problem.

Homogeneous Model Assumption. The current LTC design assumes all agents share the same KV cache dimension. For heterogeneous models of different scales, additional adapter layers would be needed to align representation spaces, potentially introducing new sources of error.

Extended Context Scenarios. While our Stress Test Protocol evaluates interactions up to 20 rounds with cumulative reasoning, the effectiveness of LTC on ultra-long individual sequences exceeding 8192 tokens has not been fully validated and requires further investigation.

Task Specificity. Although our experiments span mathematical reasoning, code generation, and scientific domains, certain highly specialized fields such as legal reasoning or financial analysis may require domain-adapted semantic probes to achieve optimal performance.

Ethical Considerations. Latent communication reduces the transparency of inter-agent information exchange compared to text-based methods, potentially complicating the detection of unintended or adversarial agent behaviors. Additionally, efficiency improvements in multi-agent systems could lower barriers to misuse in automated disinformation generation. We encourage future work to develop auditing mechanisms for latent communication protocols.

Use of AI Assistants. We used AI assistants for code debugging, grammar checking, and manuscript proofreading. All scientific contributions including methodology design, theoretical analysis, experimental setup, and result interpretation were conducted entirely by the authors.

Acknowledgments

We would like to acknowledge the support and collaboration effort of the project team. This work was supported by the CAS Project for Young Scientists in Basic Research under Grant No.YSBR-040, and by Beijing Natural Science Foundation under Grant No. 4262072 and Grant No. 4264133.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and 1 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and 1 others. 2022. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE.

Jacob Andreas. 2022. Language models as agent models. *arXiv preprint arXiv:2212.01681*.

Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. 2024. Llm augmented llms: Expanding capabilities through composition. In *The Twelfth International Conference on Learning Representations*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, and 1 others. 2020. Experience grounds language. *arXiv preprint arXiv:2004.10151*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024a. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and 1 others. 2024b. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Harrison Chase and 1 others. 2023. LangChain. https://github.com/langchain-ai/langchain. Accessed: 2026-01-5.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *ICLR*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. Tarmac: Targeted multi-agent communication. In *International Conference on machine learning*, pages 1538–1546. PMLR.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, and 1 others. 2023. Palm-e: An embodied multimodal language model.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

FlagEval. 2025. Hmmt 2025. https://huggingface.co/datasets/FlagEval/HMMT_2025. Accessed: 2026-01-01.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. 2025. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023a. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023b. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Jiechuan Jiang and Zongqing Lu. 2018. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 17889–17904.

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

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. 2024a. Deliberation in latent space via differentiable cache augmentation. *arXiv preprint arXiv:2412.17747*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023c. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023d. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. Maven: Multi-agent variational exploration. *Advances in neural information processing systems*, 32.

Kangfu Mei, Mauricio Delbracio, Hossein Talebi, Zhengzhong Tu, Vishal M Patel, and Peyman Milanfar. 2024. Codi: Conditional diffusion distillation for higher-fidelity and faster image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9048–9058.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, and 1 others. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Steven T Piantadosi, Dyana CY Muller, Joshua S Rule, Karthikeya Kaushik, Mark Gorenstein, Elena R Leib, and Emily Sanford. 2024. Why concepts are (probably) vectors. *Trends in Cognitive Sciences*, 28(9):844–856.

JULIAN M PINE. 2005. Tomasello, m., constructing a language: a usage-based theory of language acquisition. cambridge, ma: Harvard university press, 2003. pp. 388. hardback,£ 29.95. isbn 0-674-01030-2. *Journal of Child Language*, 32(3):697–702.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang,

Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR.

Sainbayar Sukhbaatar, Rob Fergus, and 1 others. 2016. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29.

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2023. Cognitive architectures for language agents. *Transactions on Machine Learning Research*.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and 1 others. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, Chengyao Wen, Congqi Li, Deng Zhao, Dingbo Yuan, Donghai You, Fagui Mao, Fanzhuang Meng, Feng Xu, Guojie Li, and 85 others. 2025. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023b. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. 2023c. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36:34153–34189.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Lionel Wong, Gabriel Grand, Alexander K Lew, Noah D Goodman, Vikash K Mansinghka, Jacob Andreas, and Joshua B Tenenbaum. 2023. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang,

Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. Examining inter-consistency of large language models collaboration: An in-depth analysis via debate. *arXiv preprint arXiv:2305.11595*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, and 1 others. 2024a. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17591–17599.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024b. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3679–3689.

Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. 2025. Towards efficient llm grounding for embodied multi-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1663–1699.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024. https://huggingface.co/datasets/math-ai/aime24. Accessed: 2026-01-01.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025. https://huggingface.co/datasets/math-ai/aime25. Accessed: 2026-01-01.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. 2025. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, and 1 others. 2025. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*.

A Theoretical Analysis

This appendix provides formal definitions, the complete theorem statement, and detailed proof for the compression error bound summarized in Section 3.4.

A.1 Preliminaries

Definition A.1 (Attention Concentration)**.** For a query distribution over $\mathbb{R}^{d_h}$, the attention concentration factor is defined as:

$$\rho = \min_{q} \max_{\mathcal{S}\subseteq[T],|\mathcal{S}|=K} \sum_{j\in\mathcal{S}} \alpha_j(q) \tag{8}$$

where $\alpha_j(q) = [\mathrm{softmax}(qK^\top/\sqrt{d_h})]_j$ denotes the attention weight of query q on position j. This measures the minimum fraction of attention mass that can be captured by the best K positions across all possible queries.

Assumption A.2 (Bounded Representations)**.** All value vectors in the KV cache satisfy $\|v_j\|_2 \leq V_{\max}$ for all $j \in [T]$.

A.2 Compression Error Bound

Theorem A.3 (Attention Output Error Bound)**.** *Let $O = \mathrm{softmax}(QK^\top/\sqrt{d_h})V$ be the original attention output and $\tilde{O} = \mathrm{softmax}(Q\tilde{K}^\top/\sqrt{d_h})\tilde{V}$ be the output computed with compressed KV cache, where $\tilde{K} = AK$ and $\tilde{V} = AV$ for some row-stochastic aggregation matrix $A \in \mathbb{R}^{K\times T}$. Under Assumption A.2 and Definition A.1:*

$$\|O - \tilde{O}\|_F \le 2(1-\rho)\cdot V_{\max}\cdot\sqrt{n_q} \tag{9}$$

where n_q is the number of queries.

Remark A.4. Theorem A.3 establishes an error bound based on a constructive proof using a *hard selection* strategy, in which the aggregation matrix A contains binary entries. Importantly, the continuous row-stochastic aggregation matrices learnable by LTC constitute a superset of valid selection matrices. Consequently, the minimum error achievable by our aggregation method is theoretically bounded by, and potentially lower than, the error of the optimal selection strategy. This confirms that LTC possesses sufficient expressivity to satisfy the derived bound.

Proof. We proceed in three steps: first establishing the algebraic form of the error, then bounding the error for a single query, and finally aggregating over all queries. **Step 1: Algebraic Form of the Error.** For a single query $q \in \mathbb{R}^{d_h}$, the original attention output is:

$$o(q) = \mathrm{Attn}(q, K, V) = \sum_{j=1}^{T} \alpha_j(q) v_j \tag{10}$$

where the attention weights are:

$$\alpha_j(q) = \frac{\exp(q^\top k_j/\sqrt{d_h})}{\sum_{l=1}^{T}\exp(q^\top k_l/\sqrt{d_h})} \tag{11}$$

For the compressed KV cache with $\tilde{K} = AK$ and $\tilde{V} = AV$, the output is:

$$\tilde{o}(q) = \sum_{i=1}^{K} \tilde{\alpha}_i(q)\tilde{v}_i \tag{12}$$

Since $\tilde{v}_i = \sum_{j=1}^{T} A_{ij}v_j$, we can rewrite this as:

$$\begin{aligned}\tilde{o}(q) &= \sum_{i=1}^{K}\tilde{\alpha}_i(q)\sum_{j=1}^{T} A_{ij}v_j \\ &= \sum_{j=1}^{T}\underbrace{\left(\sum_{i=1}^{K}\tilde{\alpha}_i(q)A_{ij}\right)}_{\triangleq \tilde{w}_j(q)} v_j\end{aligned} \tag{13}$$

The effective weight $\tilde{w}_j(q) = \sum_{i=1}^{K}\tilde{\alpha}_i(q)A_{ij}$ forms a valid probability distribution over $[T]$:

$$\begin{aligned}\sum_{j=1}^{T}\tilde{w}_j(q) &= \sum_{j=1}^{T}\sum_{i=1}^{K}\tilde{\alpha}_i(q)A_{ij} \\ &= \sum_{i=1}^{K}\tilde{\alpha}_i(q)\underbrace{\sum_{j=1}^{T}A_{ij}}_{=1} = 1\end{aligned} \tag{14}$$

where we use the row-stochastic property of A. **Step 2: Single Query Error Bound.** The per-query error can be expressed as:

$$o(q) - \tilde{o}(q) = \sum_{j=1}^{T}(\alpha_j(q) - \tilde{w}_j(q))v_j \tag{15}$$

Applying the triangle inequality and Assumption A.2:

$$\begin{aligned}\|o(q) - \tilde{o}(q)\|_2 &\le \sum_{j=1}^{T}|\alpha_j(q) - \tilde{w}_j(q)|\cdot\|v_j\|_2 \\ &\le V_{\max}\cdot\|\alpha(q) - \tilde{w}(q)\|_1\end{aligned} \tag{16}$$

The ℓ_1 distance $\|\alpha(q) - \tilde{w}(q)\|_1$ is twice the total variation distance between distributions $\alpha(q)$ and $\tilde{w}(q)$. By Definition A.1, for any query q, there exists a subset $\mathcal{S}^*(q) \subseteq [T]$ with $|\mathcal{S}^*(q)| = K$ such that:

$$\beta(q) \triangleq \sum_{j\in\mathcal{S}^*(q)}\alpha_j(q) \ge \rho \tag{17}$$

Consider the selection aggregation matrix A^* that selects exactly the positions in $\mathcal{S}^*(q) = \{j_1, \ldots, j_K\}$:

$$A^*_{i,j_i} = 1, \quad A^*_{i,j} = 0 \text{ for } j \ne j_i \tag{18}$$

Under this selection, the compressed keys satisfy $\tilde{k}_i = k_{j_i}$, and the compressed attention weights become:

$$\begin{aligned}\tilde{\alpha}_i(q) &= \frac{\exp(q^\top k_{j_i}/\sqrt{d_h})}{\sum_{l=1}^{K}\exp(q^\top k_{j_l}/\sqrt{d_h})} \\ &= \frac{\alpha_{j_i}(q)}{\sum_{l=1}^{K}\alpha_{j_l}(q)} = \frac{\alpha_{j_i}(q)}{\beta(q)}\end{aligned} \tag{19}$$

The effective weight distribution under this selection is:

$$\tilde{w}_j(q) = \begin{cases}\alpha_j(q)/\beta(q) & \text{if } j \in \mathcal{S}^*(q) \\ 0 & \text{if } j \notin \mathcal{S}^*(q)\end{cases} \tag{20}$$

Computing the total variation distance:

$$\begin{aligned}\|\alpha(q) - \tilde{w}(q)\|_1 &= \sum_{j\in\mathcal{S}^*} \left|\alpha_j(q) - \frac{\alpha_j(q)}{\beta(q)}\right| \\ &\quad + \sum_{j\notin\mathcal{S}^*} |\alpha_j(q)| \\ &= \sum_{j\in\mathcal{S}^*} \alpha_j(q) \left|1 - \frac{1}{\beta(q)}\right| \quad (21) \\ &\quad + (1-\beta(q)) \quad (22)\end{aligned}$$

Since $\beta(q) \leq 1$, we have $|1 - 1/\beta(q)| = (1 - \beta(q))/\beta(q)$. Therefore:

$$\begin{aligned}\|\alpha(q) - \tilde{w}(q)\|_1 &= \beta(q) \cdot \frac{1-\beta(q)}{\beta(q)} + (1-\beta(q)) \\ &= 2(1-\beta(q)) \quad (23)\end{aligned}$$

Substituting into Equation (16) and using $\beta(q) \geq \rho$:

$$\begin{aligned}\|o(q) - \tilde{o}(q)\|_2 &\leq 2(1-\beta(q)) \cdot V_{\max} \leq \\ &\quad 2(1-\rho) \cdot V_{\max} \quad (24)\end{aligned}$$

Step 3: Aggregation over Multiple Queries. For n_q queries forming the matrix $Q_s = [q_1, \ldots, q_{n_q}]^\top$, let $O, \tilde{O} \in \mathbb{R}^{n_q \times d_h}$ denote the original and compressed attention outputs respectively, with the i-th row being $o(q_i)^\top$ and $\tilde{o}(q_i)^\top$. The Frobenius norm of the error matrix is:

$$\begin{aligned}\|O - \tilde{O}\|_F &\leq \sqrt{\sum_{i=1}^{n_q} [2(1-\rho)V_{\max}]^2} \quad (25) \\ &= 2(1-\rho) \cdot V_{\max} \cdot \sqrt{n_q}\end{aligned}$$

Applying the single-query bound from Equation (24):

$$\begin{aligned}\|O - \tilde{O}\|_F &\leq \sqrt{\sum_{i=1}^{n_q} [2(1-\rho)V_{\max}]^2} \quad (26) \\ &= 2(1-\rho) \cdot V_{\max} \cdot \sqrt{n_q}\end{aligned}$$

Relationship to Aggregation. The derivation above constructs a specific selection matrix A^*. Our LTC module learns a continuous aggregation matrix $A \in \mathbb{R}^{K\times T}$ whose rows sum to 1. Since the set of such matrices includes A^* as a special boundary case, the optimal learned aggregation satisfies:

$$\begin{aligned}\min_A \|O - \tilde{O}(A)\|_F &\leq \|O - \tilde{O}(A^*)\|_F \\ &\leq 2(1-\rho)V_{\max}\sqrt{n_q} \quad (27)\end{aligned}$$

Thus, the bound derived for selection serves as a valid upper bound for our proposed aggregation method. □

A.3 Multi-Round Error Behavior

We provide a brief analysis of how compression errors accumulate across collaboration rounds.

Proposition A.5 (Linear Error Accumulation)**.** *Let δ denote the per-round compression error. After R collaboration rounds, the cumulative error ϵ_R satisfies:*

$$\epsilon_R \leq R \cdot \delta \quad (28)$$

Proof. At each round r, compression introduces error at most δ. By the triangle inequality, errors across rounds accumulate additively in the worst case, yielding the linear bound. □

Remark A.6. The linear bound represents the worst-case scenario where errors accumulate constructively. In practice, we observe sub-linear growth because LTC learns to prioritize semantically critical information that persists across rounds, while noise and less relevant details are naturally attenuated. Experimental validation in Section 4.3 confirms that CondenseFlow maintains stable performance through extended interaction depths where baselines exhibit significant degradation.

B LTC Architecture Details

B.1 Module Specification

The Latent Thought Condenser consists of the following components: **Semantic Probes.** A learnable matrix $Q_c \in \mathbb{R}^{K\times d_k}$ initialized from $\mathcal{N}(0, 0.02)$, where d_k denotes the KV cache dimension. For models using Grouped Query Attention, $d_k = n_{kv} \times d_{head}$ where n_{kv} is the number of KV heads and d_{head} is the per-head dimension. **Cross-Attention.** Standard scaled dot-product attention without learned projections. The probes directly attend over original key vectors. **Normalization.** Layer normalization applied to compressed value vectors with learned affine parameters, adding $2 \times d_k$ parameters per layer. **Parameter Sharing.** All Transformer layers share identical probe vectors. Only the layer normalization parameters are layer-specific.

B.2 Total Parameter Count

For an L-layer model with KV dimension d_k and compression dimension K:

$$\text{Parameters} = K \times d_k + L \times 2 \times d_k \quad (29)$$

For Qwen3-14B with $L = 40$, $n_{kv} = 8$, $d_{head} = 128$, and $K = 64$:

$$d_k = 8 \times 128 = 1024 \tag{30}$$

$$\begin{aligned} \text{Parameters} &= 64 \times 1024 + 40 \times 2 \times 1024 \\ &= 147,456 \approx 0.15\text{M} \end{aligned} \tag{31}$$

This represents less than 0.01% of the base model parameters, enabling efficient training and minimal inference overhead.

C Training Details

C.1 Complete Loss Function

The full training objective combines reconstruction with two regularization terms:

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_1 \mathcal{L}_{cover} + \lambda_2 \mathcal{L}_{orth} \tag{32}$$

Reconstruction Loss.

$$\begin{aligned} \mathcal{L}_{recon} = \frac{1}{L} \sum_{l=1}^{L} \big\| &\text{Attn}(Q_s, K^{(l)}, V^{(l)}) \\ &- \text{Attn}(Q_s, \tilde{K}^{(l)}, \tilde{V}^{(l)}) \big\|_F^2 \end{aligned} \tag{33}$$

where $Q_s \in \mathbb{R}^{n_s \times d_h}$ contains $n_s = 128$ sampled query vectors.

Coverage Regularization.

$$\mathcal{L}_{cover} = -H(\bar{A}^{(l)}) = \sum_{j=1}^{T} \bar{a}_j \log \bar{a}_j \tag{34}$$

where $\bar{A}^{(l)} = \frac{1}{K} \sum_{i=1}^{K} A_i^{(l)}$ is the mean aggregation weight. Maximizing entropy prevents probe collapse.

Orthogonality Regularization.

$$\mathcal{L}_{orth} = \left\| Q_c Q_c^\top - I_K \right\|_F^2 \tag{35}$$

This encourages probes to capture complementary features.

C.2 Hyperparameters

C.3 Training Data

To ensure generalization, training data is drawn from sources disjoint from evaluation benchmarks:

Mathematical Reasoning. 1,000 problems from GSM8K training set covering arithmetic and algebraic patterns.

Code Generation. 1,000 problems from MBPP training split covering common programming patterns.

Table 6: Training hyperparameters for LTC.

Parameter	Value
Optimizer	AdamW
Learning rate	1×10^{-4}
LR schedule	Cosine decay
Warmup steps	1,000
Total steps	50,000
Batch size	64
λ_1 (coverage)	0.1
λ_2 (orthogonality)	0.01
Sampled queries n_s	128

Generalization Strategy. Although our evaluation extends to complex benchmarks like AIME and GPQA, we hypothesize that the *mechanisms of information salience*—such as attending to critical numerical values, logical operators, and state transitions—share structural similarities across reasoning complexities. By learning to compress these fundamental semantic structures rather than memorizing specific problem patterns, the lightweight probes can generalize to more advanced domains without task-specific fine-tuning.

C.4 Training Cost

Training requires approximately 4 GPU-hours on a single NVIDIA A100-80G. The minimal parameter count enables rapid iteration during development.

D Implementation Details

D.1 Inference Configuration

Table 7: Inference hyperparameters.

Parameter	Value
Sampling method	Nucleus sampling
Top-p	0.95
Temperature	0.6
Maximum length	2048 tokens
Compression dimension K	64

D.2 Hardware

All experiments conducted on NVIDIA A100-80G GPUs. Standard Protocol experiments use single GPU. Stress Test Protocol uses single GPU with gradient checkpointing for extended sequences.

D.3 Agent Prompts

Prompts for each agent role are provided in Appendix G.

E Effective Rank Analysis

Definition E.1 (Effective Rank)**.** For matrix $M \in \mathbb{R}^{m\times n}$, the effective rank is:

$$\text{erank}(M) = \exp\left(-\sum_{i=1}^{\min(m,n)} \hat{\sigma}_i \log \hat{\sigma}_i\right) \quad (36)$$

where $\hat{\sigma}_i = \sigma_i / \|\boldsymbol{\sigma}\|_1$ is the normalized singular value.

We empirically validate the low-rank property discussed in Section 3.4 by computing effective ranks of KV caches across models.

Table 8: Effective rank statistics of KV caches on AIME 2025 samples.

Model	Median erank	erank < 64
Qwen3-8B-Instruct	47.2	66.8%
LLaMA-3.1-8B-Instruct	49.1	63.5%
Gemma-2-9b-it	46.5	68.2%
Qwen3-14B	41.3	74.6%
DeepSeek-R1-Distill-14B	39.8	77.2%
Ring-mini-2.0	38.2	79.5%

Key observations:

Low effective rank is prevalent. Median effective rank is below 50 for all models, well under our default compression dimension of K=64.

Mid-scale models exhibit lower rank. Effective rank decreases with model scale, explaining the reduced accuracy gap between CondenseFlow and Dense-Latent on larger models.

Compression dimension is well-chosen. Over 65% of samples have effective rank below 64, validating that compression with K=64 preserves the dominant semantic subspace.

F Per-Task Analysis

Table 9: Accuracy gap between CondenseFlow and Dense-Latent by task category averaged across models.

Category	Small-Scale	Mid-Scale
Math (AIME, HMMT)	-1.9%	-2.0%
Code (MBPP+, LiveCode)	-1.3%	-0.9%
Science (GPQA, MedQA)	-1.5%	-0.5%
Average	-1.6%	-1.1%

Mathematical tasks show the largest gap due to their requirement for tracking distributed numerical dependencies across lengthy derivations. Code generation shows moderate gap, with CondenseFlow matching or exceeding Dense-Latent on MBPP+ while showing larger gaps on the more challenging LiveCodeBench. Scientific reasoning exhibits the smallest gap on mid-scale models, suggesting that domain knowledge compression is effective when model capacity is sufficient.

G Agent Prompts

To ensure rigorous reasoning, we employ structured system prompts that enforce role-specific behaviors and output formats. The prompts below are templated with placeholders for the input question and interaction history.

G.1 Standard Protocol Prompts

Planner Agent.

```
Role: You are a Distinguished Professor of Mathematics and Computer Science.
Task: Given a complex problem, devise a comprehensive solution strategy. Decompose the problem into logical sub-tasks. Identify necessary theorems, formulas, and potential edge cases (e.g., boundary conditions, integer constraints).
Constraint: Do not compute the final numerical answer yet. Focus on the logical path.

Problem: {question}
```

Critic Agent.

```
Role: You are a Strict Peer Reviewer.
Task: Audit the proposed plan/solution for logical soundness.
1. Check for valid application of theorems.
2. Identify potential calculation risks or overlooked constraints.
3. Verify if the approach covers all cases requested by the problem.
Output: Be harsh but constructive. List specific flaws if any.

Problem: {question}
```

Refiner Agent.

```
Role: You are a Lead Solution Architect.
Task: Synthesize the original plan and the Critic's feedback into a robust, executable guide.
1. Fix the flaws identified by the Critic.
2. Optimize the steps for clarity and precision.
3. Ensure all constraints from the problem statement are explicitly addressed.

Problem: {question}
```

Solver Agent.

Role: You are a Precision Calculation Engine.
Task: Execute the refined plan step-by-step to derive the final answer.
Requirements:
- Show every step of the derivation clearly.
- Perform double-checks on arithmetic operations.
- Conclude with the final answer strictly in the specified format.

Problem: {question}

Format: End your response with: Answer: [final_answer]

G.2 Stress Test Protocol Prompts

Solver Agent (Iterative).

Role: You are an Iterative Mathematical Solver.
Task: Solve the problem or refine your previous solution based on feedback.
Instructions:
1. Review the previous attempt and the Critic's feedback (if available).
2. If the feedback points out errors, re-derive the affected steps carefully.
3. If this is the first turn, solve from scratch.
4. Maintain global consistency with the problem constraints.

Problem: {question}

Previous Feedback: {feedback}

Format: Answer: [final_answer]

Critic Agent (Iterative).

Role: You are a Logical Consistency Checker.
Task: Evaluate the Solver's latest proposal.
Checklist:
- Is the reasoning mathematically valid?
- Are the calculations correct?
- Does the solution satisfy ALL conditions in the problem statement (e.g., $x > 0$, integer solutions)?

Output: Provide a concise verification report. If incorrect, point out exactly which step failed.

Problem: {question}

Proposed Solution: {solution}

H Case Study

We present a detailed comparison on an AIME 2025 problem demonstrating how different methods handle multi-step mathematical reasoning.

H.1 Problem Statement

AIME 2025 Problem

Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.

H.2 Problem Analysis

Converting to base 10: $17_b = b + 7$ and $97_b = 9b + 7$. The divisibility condition requires $(b+7) \mid (9b+7)$. Since $(b+7) \mid 9(b+7) = 9b + 63$, we have $(b+7) \mid (9b+63) - (9b+7) = 56$. Thus $b + 7$ must be a divisor of 56. The divisors of 56 are: 1, 2, 4, 7, 8, 14, 28, 56. For $b > 9$: $b + 7 > 16$, so valid divisors are 28 and 56. This gives $b = 21$ and $b = 49$. **Answer: 70**

H.3 Method Comparison at Round 10

TextMAS Result.

TextMAS Output (Incorrect)

After 10 rounds of refinement, the accumulated text context exceeds 40000 tokens. The model suffers from "Constraint Amnesia," losing track of the $b > 9$ condition amidst the verbose history: "...As verified in step 4, we need $(b + 7)$ to divide 56. The divisors are 1, 2, 4, 7, 8, 14, 28, 56. Case 1: $b + 7 = 14 \implies b = 7$. Checking: $17_7 = 14, 97_7 = 70$. $70/14 = 5$. This works. Case 2: $b + 7 = 28 \implies b = 21$. Valid. Case 3: $b + 7 = 8 \implies b = 1$. Base must be greater than 1. Wait, let's re-verify Case 1. Base 7 is valid because 17_7 exists. So the bases are 7 and 21. Sum = $7 + 21 = 28$." *Analysis: The model hallucinates that $b = 7$ is valid, forgetting the problem statement's explicit constraint $b > 9$ due to context dilution.* **Answer: 28 ✗**

Dense-Latent Result.

Dense-Latent Output (Incorrect)

After 10 rounds, the model exhibits "Attention Dispersion." It retrieves a rejected hypothesis from Round 3 that conflicts with the current derivation: "We have established that b must be 21 or 49. Let's double check $b = 49$. $17_{49} = 56$, $97_{49} = 448$. $448/56 = 8$. Correct. However, previously in the Critic's feedback (Round 3), it was noted that 'the sum must be less than 50'. If $b = 49$, the base itself is large, but does it fit the range? Let's reconsider the factorization $9b + 7 = 9(b + 7) - 56$. Maybe we missed a negative divisor? Let's exclude 49 to be safe based on the prior critique. Only $b = 21$ remains." *Analysis: The model fails to distinguish between a valid current derivation and an incorrect historical objection (the "sum < 50" hallucination from an earlier round), causing it to discard a correct solution.* **Answer: 21 ✗**

CondenseFlow Result.

CondenseFlow Output (Correct)

With compressed semantic anchors, the context remains focused. The LTC module filters out the noise from previous rounds, retaining only the verified state space: "Current State Summary: Divisibility condition reduces to $(b+7)|56$. Constraint $b > 9$ implies $b + 7 > 16$. Candidate divisors > 16: $\{28, 56\}$. Candidate bases: 1. $b + 7 = 28 \implies b = 21$. 2. $b + 7 = 56 \implies b = 49$. Verification Step: For $b = 21$: $17_b|97_b$ holds (Ratio 7). For $b = 49$: $17_b|97_b$ holds (Ratio 8). Both candidates satisfy all conditions. Sum: $21 + 49 = 70$." **Answer: 70 ✓**

H.4 Analysis

This case illustrates the specific failure modes addressed by CondenseFlow: **Constraint Adherence vs. TextMAS.** TextMAS eventually drowned out the global constraint ($b > 9$) with verbose intermediate verifications. CondenseFlow's compression implicitly prioritized the problem statement constraints as high-value semantic anchors, ensuring they persisted through 10 rounds of interaction. **State Coherence vs. Dense-Latent.** Dense-Latent suffered from interference between current reasoning and obsolete historical "thoughts". By compressing the history into a coherent state representation, CondenseFlow prevented the model from attending to outdated negative feedback, enabling it to confidently retain both correct solutions.