SEE WHAT I SEE, KNOW WHAT I THINK: DENSE LATENT COMMUNICATION ACROSS HETEROGENEOUS AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents build internal representations of the information they observe and the reasoning they perform. Sharing these representations offers a way to communicate both source information and reasoning across agents. For agents built from different models, this requires aligning their representations while preserving information useful to the receiver. We study this problem through KV-cache communication, examining how an agent uses internal states shared by other agents, with or without direct access to the information that other agents observed. A controlled self-communication study shows that cache pruning causes substantially greater degradation when the receiving agent lacks access to that information. We use this finding to guide dense cross-model cache alignment, combining positional disentanglement and KV-group transformations with reconstruction followed by generation training. Across six directed Qwen3 pairs, aligned caches improve in-domain accuracy over text communication when both agents observe the same input, with fewer estimated inference FLOPs. Experiments with three-agent document sharing and Mistral-to-Qwen transfer further demonstrate that aligned caches can carry information across both multiple separate observations and different model families.

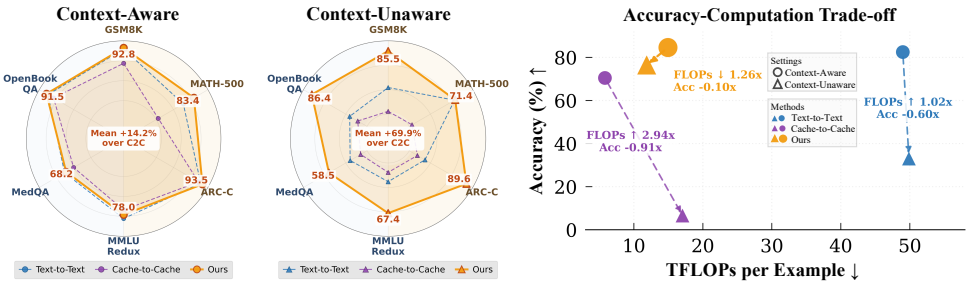

Figure 1: **Dense latent communication across heterogeneous agents.** Aligned sender KV caches support receiver generation with or without direct access to the source input. We evaluate task accuracy and estimated inference FLOPs.

1 INTRODUCTION

Language-model agents can contribute both what they observe and what they infer: a retriever may supply evidence, while a planner develops intermediate conclusions (Tran et al., 2025; Wang et al., 2026). Their internal representations offer a channel for sharing source information and the computations built on it (Zheng et al., 2025; Du et al., 2025; Zou et al., 2025). Text provides a flexible, model-agnostic interface, but expressing these contributions as messages requires generation and may omit useful details. Latent communication (Figure 2) instead aims to make the sender's internal states directly usable by another agent.

KV caches retain contextual states computed by the sender and provide an interface that the receiver's attention layers can use directly. Work on cache sharing, selective transmission, and learned projection provides ways to reuse these states across agents (Shi et al., 2025; Fu et al., 2025; Li et al., 2026b; Liu et al., 2024). For example, Cache-to-Cache learns projection and fusion between models that process a shared input (Fu et al., 2025). We study how to make these states useful across models,

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.

including when the receiver has not seen the sender's input. This requires handling differences in cache dimensions, layer organization, positional conventions, and learned representations (Fu et al., 2025; Heo et al., 2026), while preserving task-relevant content. To examine how transfer depends on the receiver's available context, we compare context-aware communication, where the receiver sees the source input, with context-unaware communication, where it relies on the transferred message. We also consider agents with complementary private evidence (Section 2.2). Appendix A discusses related interfaces in more detail.

Our starting point is a simple question: how much of a cache must be retained when the receiver has not seen the source input? In a controlled Qwen3-4B self-communication study, context-aware performance can be largely preserved with a small subset of selected KV groups, whereas context-unaware performance requires substantially denser retention. This difference motivates an alignment objective that preserves the contents of the cache, alongside an objective for downstream generation. Our approach combines positional disentanglement, per-KV-group transformations and gates, and two-phase training: receiver-cache reconstruction followed by generation supervision.

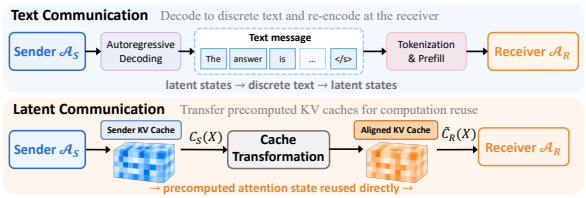

Figure 2: Text communication generates and re-encodes messages; latent communication transfers aligned KV caches.

Across six directed pairs of Qwen3 models, the learned interface supports communication with and without receiver access to the source input. In the context-aware setting, it improves accuracy over text communication on all three in-domain benchmarks across all six pairs, while using fewer estimated inference FLOPs. In the context-unaware setting, it outperforms both text communication and Cache-to-Cache in 33 of the 36 Qwen3 task–pair combinations in our evaluation. In the three-agent HotpotQA task, the receiver combines two independently encoded document memories, improving answer exact match and F1 over both text communication and a receiver given all documents directly in all three evaluated configurations. Mistral-to-Qwen transfer also improves over text communication on the three in-domain benchmarks in both context-aware and context-unaware settings. Together, these results demonstrate that aligned caches can support collaboration across model sizes, model families, and distributed observations. Our main contributions are summarized as follows.

- **Observation-dependent analysis:** a controlled KV-group retention study comparing communication with and without receiver access to the source input.
- **Cross-model cache alignment:** a dense KV-cache interface trained to reconstruct receiver caches and support downstream generation.
- **Empirical evaluation:** transfer across six directed Qwen3 pairs and six benchmarks, comparing task accuracy and inference FLOPs, with additional tests of distributed document memories and cross-family transfer.

2 Communication Setup

2.1 KV-Cache Interface

We consider language-model agents whose local states reflect both their observations and model-specific computations. Each directed interaction transfers a locally constructed cache into a receiving agent's representation space; a receiver may consume several independently constructed memories. We first define the interface from a sender language model $\mathcal{A}_S$ to a receiver $\mathcal{A}_R$. Given source input $\boldsymbol{X}_S$, the sender constructs a KV cache

$$\mathcal{C}_S(\boldsymbol{X}_S) = \{(\boldsymbol{K}_S^{(l,g)}(\boldsymbol{X}_S), \boldsymbol{V}_S^{(l,g)}(\boldsymbol{X}_S))\}_{l,g}, \tag{1}$$

where l indexes layers and g indexes KV groups. A learned transformation produces a receiver-compatible cache $\tilde{\mathcal{C}}_R = \mathcal{T}_{\boldsymbol{\theta}}(\mathcal{C}_S(\boldsymbol{X}_S))$. The receiver generates an output using this cache and its locally available input $\boldsymbol{X}_R$:

$$p_{\mathcal{A}_R}(\boldsymbol{y} \mid \boldsymbol{X}_R, \tilde{\mathcal{C}}_R). \tag{2}$$

Copies of the same model can share compatible cache representations directly, whereas different models generally require alignment of cache structure and learned representations. We review standard attention operations, tensor dimensions, and KV-cache construction in Appendix B.

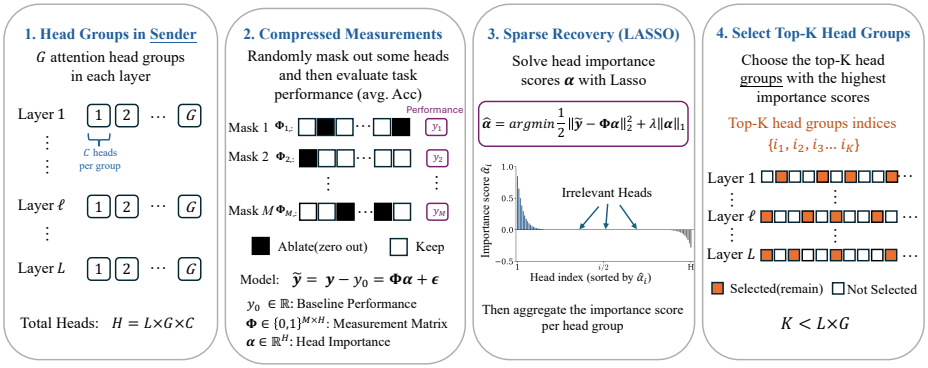

Figure 3: Post-hoc ranking and retention procedure. Random sender query-head ablations supply performance measurements for a Lasso surrogate. Aggregated coefficients rank KV groups for a separate sweep that zeros non-selected cache groups.

2.2 Communication Regimes

We distinguish three regimes according to receiver access to task information:

- **Context-aware:** both agents observe the source input, $\boldsymbol{X}_S = \boldsymbol{X}_R = \boldsymbol{X}$. The transferred cache supplies the sender's representation of an input also available to the receiver.
- **Context-unaware:** only the sender observes $\boldsymbol{X}$; $\boldsymbol{X}_R = \emptyset$ denotes the absence of source input, rather than the absence of a generation prompt. The message is the receiver's only access to source-side task information, including the question and any answer options.
- **Partial-aware:** agents have complementary local observations, and communication conveys evidence unavailable to the receiver. Question visibility depends on the task: in our three-agent evaluation (Section 5.3), two senders encode private supporting documents before the question arrives, while the receiver later observes the question and its own distractor documents.

The aware and unaware regimes provide controlled comparisons of source-input availability. The partial-aware setting evaluates how a receiver uses evidence held by other agents, including memories constructed before its question arrives.

For m incoming memories, the receiver conditions on

$$p_{\mathcal{A}_R}\left(\boldsymbol{y} \mid \boldsymbol{X}_R, \mathrm{Merge}(\widetilde{\mathcal{C}}_{1\to R}, \ldots, \widetilde{\mathcal{C}}_{m\to R})\right). \tag{3}$$

Here each cache is transformed independently. Our evaluated extension uses two senders and one receiver; Merge concatenates transformed content caches along the sequence dimension and applies receiver positional encoding over the combined sequence (Section 4).

3 How Receiver Context Changes Cache Requirements

Controlled setup. We ask whether receiver access to the source input changes how much of a transmitted KV cache is needed for successful communication. To isolate this question from cross-model alignment, we use Qwen3-4B as both sender and receiver on GSM8K, MATH-algebra, and ARC-Challenge. The sender encodes the source input without autoregressive generation. We compare the context-aware and context-unaware regimes defined in Section 2.2. This self-communication study probes cache-retention sensitivity; heterogeneous alignment is evaluated separately.

3.1 Retention Experiment

Figure 3 outlines our adaptation of the compressed-sensing head-ablation method extended from Bair et al. (2026). Let $\Phi \in \{0,1\}^{M\times N}$ encode M random ablations over the sender's N query heads across all layers: $\Phi_{mi} = 1$ means head i is ablated in measurement m. Given the resulting task accuracies $\mathbf{y}$ and unablated accuracy y_0, we fit a Lasso surrogate to the centered measurements $\tilde{\mathbf{y}} = \mathbf{y} - y_0\mathbf{1}$:

$$(\hat{\mathbf{x}}, \hat{b}) = \arg\min_{\mathbf{x},b} \frac{1}{2M}\|\tilde{\mathbf{y}} - \Phi\mathbf{x} - b\mathbf{1}\|_2^2 + \alpha\|\mathbf{x}\|_1, \tag{4}$$

where b is an intercept and α controls sparsity. More-negative coefficients indicate heads whose ablation predicts a larger accuracy decrease. For KV group g in layer l, we aggregate its associated

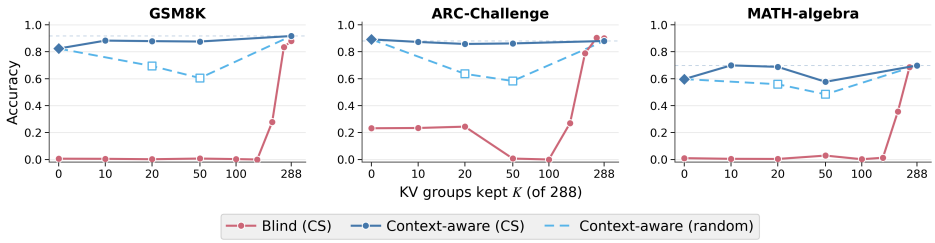

Figure 4: **Receiver context changes cache-retention sensitivity.** Qwen3-4B self-communication accuracy versus retained KV groups (K of 288). Solid blue: context-aware CS selection; open blue squares: context-aware random selection; solid red: context-unaware CS selection. The context-aware $K = 0$ point is the receiver-only baseline. Non-selected cache groups are zeroed.

query-head coefficients as

$$s_{l,g} = \sum_{h \in \text{group}(g)} \hat{x}_{l,h}, \tag{5}$$

where $\hat{x}_{l,h}$ indexes the fitted vector by layer and head. We retain the K groups with the smallest scores out of 288 and compare against random group selection under both receiver regimes. The additive surrogate ranks groups by estimated ablation sensitivity. Non-selected key and value entries are zeroed without changing tensor shapes, so the sweep measures retention sensitivity rather than transmitted-byte savings. Appendix E gives the ablation implementation, fitting settings, and evaluation details.

3.2 Findings and Implications

Figure 4 distinguishes supplementing an available input from conveying missing source information. With access to the source input, the receiver maintains high accuracy with only 10 of 288 KV groups: 88.3%, 87.3%, and 69.9% on GSM8K, ARC-Challenge, and MATH-algebra, versus 91.7%, 88.0%, and 69.8% with all groups. Without that input, accuracy remains low through $K = 150$ and recovers at higher retention. The contrast shows that access to the source input reduces dependence on the transmitted cache. When that input is unavailable, successful generation depends on retaining a broader set of cache groups. Receiver-only references use separate evaluation runs, so we base this interpretation on retention sensitivity within each regime (Appendix E).

Performance also varies nonlinearly with retention, and adding groups does not consistently help in the context-aware regime. Group selection matters: at $K = 50$, selected groups achieve 87.6% on GSM8K, compared with 60.5% for random selection, although the two perform similarly at $K = 10$. These patterns suggest unequal, context-dependent group contributions and possible interactions. Dense coverage need not imply uniform weighting.

The contrast in Figure 5 motivates three complementary choices in Section 4. *Dense cache reconstruction* encourages preservation of broad source content. *Separate KV-group transformations and learnable gates* accommodate differences in group relevance within that coverage. *Generation supervision* refines these transformations and weights for the receiver's task, beyond matching cache values alone. This combines broad alignment with fine-grained, task-directed refinement.

4 Design of Dense Latent Communication

The retention study in Section 3 motivates broad cache coverage with group-specific adaptation: missing source information calls for denser retention, while group selection and receiver context affect its usefulness. We use *dense alignment* to mean learning receiver-compatible key and value states across receiver layers and KV groups, with cache reconstruction as well as downstream generation supervision. Learned gates weight individual groups within this interface; they do not impose a fixed sparse transmission budget. We learn a directed cache transformation $\mathcal{T}_{\boldsymbol{\theta}}$ from a sender's representation space to a receiver's, using positional disentanglement, layer alignment, KV-group transformations, and gates (Figure 6). The same interface supports a single sender or several independently encoded memories. We first describe training for the two-agent benchmarks, then explain the extension to two senders.

4.1 Two-Phase Training

Generation loss supervises the receiver's output but does not directly constrain how closely the transferred cache represents the source input. We therefore first train the adapter to reconstruct

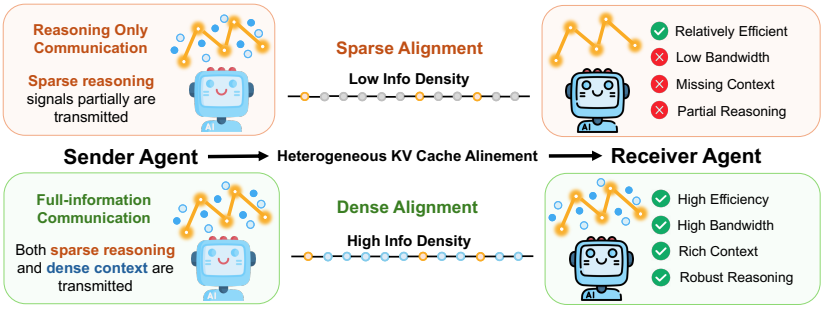

Figure 5: Conceptual contrast between selective cache transfer and dense alignment. Our retention study measures sensitivity to KV-group pruning under different receiver observation regimes; it motivates preserving broad source information when the receiver lacks the source input.

the receiver's cache, then tune it for downstream generation. This design gives the adapter both a representation target and a task objective.

Phase I: receiver-cache reconstruction. We run both agents on the same source context $\boldsymbol{X}$ and obtain the sender and receiver caches

$$\mathcal{C}_m(\boldsymbol{X}) = \left\{ \left(\boldsymbol{K}_m^{(l,g)}(\boldsymbol{X}), \boldsymbol{V}_m^{(l,g)}(\boldsymbol{X}) \right) \right\}_{l,g}, \qquad m \in \{S, R\}, \tag{6}$$

where l indexes transformer layers and g indexes KV groups. When tokenizers differ, we compare positions ending at shared character boundaries in the source text (Appendix F). The adapter produces $\widetilde{\mathcal{C}}_R(\boldsymbol{X}) = \mathcal{T}_{\boldsymbol{\theta}}(\mathcal{C}_S(\boldsymbol{X}))$ and is optimized to reconstruct the receiver's own cache:

$$\mathcal{L}_{\text{rec}} = \sum_{l,g} \left(\left\| \widetilde{\boldsymbol{K}}_R^{(l,g)} - \boldsymbol{K}_R^{(l,g)} \right\|_2^2 + \left\| \widetilde{\boldsymbol{V}}_R^{(l,g)} - \boldsymbol{V}_R^{(l,g)} \right\|_2^2 \right). \tag{7}$$

The reconstruction target encourages the adapter to retain information the receiver would encode from $\boldsymbol{X}$. This is motivated by the context-unaware setting, where the receiver must use the transformed cache in place of direct access to that input.

Phase II: generation-oriented communication. After reconstruction pretraining, we optimize the transformed cache for downstream generation under both communication regimes. For each training example, the receiver-side context $\boldsymbol{X}_R$ is drawn from a mixture of context-aware and context-unaware prompts,

$$\boldsymbol{X}_R = \begin{cases} \boldsymbol{X}, & \text{context-aware}, \\ \emptyset, & \text{context-unaware}, \end{cases} \tag{8}$$

and the same adapter $\mathcal{T}_{\boldsymbol{\theta}}$ is updated across both cases:

$$\mathcal{L}_{\text{gen}} = -\sum_t \log p_{\mathcal{A}_R} \left(y_t \mid y_{<t}, \widetilde{\mathcal{C}}_R(\boldsymbol{X}), \boldsymbol{X}_R \right). \tag{9}$$

The mixture exposes the same adapter to both uses of the cache: supplementing an observed input and providing source information that is otherwise unavailable. Reconstruction encourages agreement with receiver-native states, while generation training optimizes their usefulness for prediction.

Multiple independently encoded memories. For the three-agent task, two senders encode their documents separately without seeing the later question. Since they use the same sender model, the same adapter transforms both caches. We remove sender key rotations, transform each content cache, concatenate along the sequence dimension, and apply receiver key rotations at consecutive positions over the combined memory. Values are concatenated without rotary encoding. Phase II trains this two-memory interface conditioned on the receiver's question and local documents, using task-specific supervision rather than the original aware/unaware mixture.

4.2 Cache Transformation Architecture

Sender and receiver caches may differ in layer count, head dimension, KV-group organization, and positional encoding. Matching tensor shapes alone is insufficient because the models also learn different representations. The adapter handles these differences through positional disentanglement, fixed layer and group routing, and learned transformations.

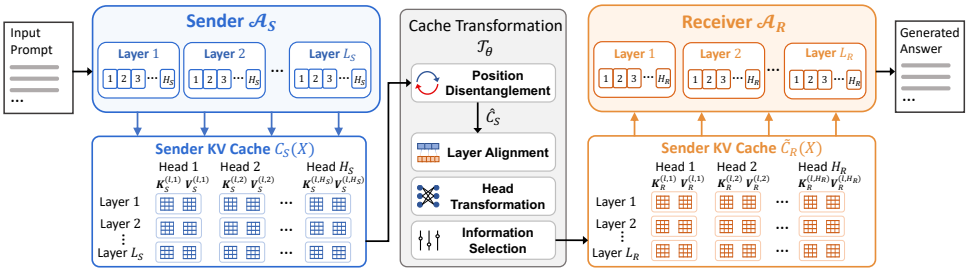

Figure 6: Overview of dense alignment: transform sender KV caches into receiver-compatible caches with positional disentanglement, per-KV-group transformations, and two-phase training.

Position-disentangled cache transformation. Rotary positional embeddings entangle keys with model-specific phase rotations. Removal and restoration below apply to keys; values are transformed without rotary encoding. Since dense communication should align the information stored in the cache rather than copy the sender's positional convention, $\mathcal{T}_{\boldsymbol{\theta}}$ first maps caches into a position-disentangled space:

$$\widehat{\mathcal{C}}_m = \mathrm{RemoveRoPE}_m\left(\mathcal{C}_m\right), \qquad m \in \{S, R\}. \tag{10}$$

The cross-model transformation is applied to $\widehat{\mathcal{C}}_S$ to produce a receiver-side content cache $\mathcal{C}'_R$. The final communicated cache restores the receiver's positional convention:

$$\widetilde{\mathcal{C}}_R = \mathrm{AddRoPE}_R\left(\mathcal{C}'_R\right). \tag{11}$$

The adapter thus transforms content without sender key rotations and places the result at the receiver's memory positions.

Layer alignment across different depths. Let L_S and L_R denote the sender and receiver depths. For each receiver layer l, we pair it with a sender layer through a monotonic depth-preserving map

$$a(l) = \min\left(\mathrm{round}\left(\frac{l L_S}{L_R}\right), L_S - 1\right), \qquad l = 0, \ldots, L_R - 1. \tag{12}$$

This fixed map pairs layers at similar relative depths.

KV-group transformation with structured gates. The adapter transforms KV heads; query heads are not stored in the cache. For receiver group g in layer l, it routes one sender group with keys and values $\widehat{\boldsymbol{K}}_S^{(a(l),\pi_l(g))}, \widehat{\boldsymbol{V}}_S^{(a(l),\pi_l(g))} \in \mathbb{R}^{B\times S\times d_S}$, where B is batch size and S is sequence length. Each token vector is mapped to $\mathbb{R}^{d_R}$ and placed in receiver slot (l, g), which the receiver's native query heads attend to.

Let G_S and G_R denote the number of sender and receiver KV groups per layer. For receiver group $g \in \{0, \ldots, G_R - 1\}$, the fixed head routing is

$$\pi_l(g) = g \bmod G_S, \tag{13}$$

so $\pi_l(g)$ maps a receiver KV group index to a sender KV group index in the aligned sender layer $a(l)$. If the layouts match, $\pi_l(g) = g$. If the receiver has more KV groups, sender groups are reused cyclically; if the sender has more KV groups, only the sender groups addressed by the receiver indices are injected. This routing is fixed, while the transformation parameters and gates below are learned. Each receiver KV group is produced from its routed sender-side content cache:

$$\left(\boldsymbol{K}_R'^{(l,g)}, \boldsymbol{V}_R'^{(l,g)}\right) = \gamma^{(l,g)} \cdot \left(T_{K,\boldsymbol{\theta}}^{(l,g)}\left(\widehat{\boldsymbol{K}}_S^{(a(l),\pi_l(g))}\right), T_{V,\boldsymbol{\theta}}^{(l,g)}\left(\widehat{\boldsymbol{V}}_S^{(a(l),\pi_l(g))}\right)\right), \tag{14}$$

where $\gamma^{(l,g)} \in [0, 1]$ is a learnable gate for the receiver-side group. In this architecture, $T_{K,\boldsymbol{\theta}}^{(l,g)}$ and $T_{V,\boldsymbol{\theta}}^{(l,g)}$ are separate MLPs for each receiver KV group, applied to each token's key and value vectors:

$$T_{\star,\boldsymbol{\theta}}^{(l,g)}(\boldsymbol{z}) = \boldsymbol{W}_{\star,2}^{(l,g)}\, \sigma\left(\boldsymbol{W}_{\star,1}^{(l,g)} \boldsymbol{z} + \boldsymbol{b}_{\star,1}^{(l,g)}\right) + \boldsymbol{b}_{\star,2}^{(l,g)}, \tag{15}$$

Here $\star \in \{K, V\}$, σ is GELU, and $\boldsymbol{z} \in \mathbb{R}^{d_S}$ is a sender key or value vector. The projections $\boldsymbol{W}_{\star,1}^{(l,g)} \in \mathbb{R}^{16d_S \times d_S}$ and $\boldsymbol{W}_{\star,2}^{(l,g)} \in \mathbb{R}^{d_R \times 16d_S}$ expand the head dimension by 16 and map to the receiver dimension. Parameters are shared across tokens but separate for each $(l, g, \star)$, including groups routed from the same sender group.

The retention study suggests that cache groups differ in their task relevance under the tested interventions. Separate transformations and gates let the adapter adjust each receiver group while reconstructing states across the cache.

Pair	Method	In-domain			OOD			TFLOPs ↓
		GSM8K	MATH-500	ARC-C	MMLU-Redux	MedQA	OpenBookQA	
Within-family: Qwen3 sender → Qwen3 receiver								
4B → 8B	Receiver-only	81.10	49.20	91.00	72.10	53.00	91.20	19.85
	T2T	88.10	76.00	91.74	**80.75**	**67.40**	90.40	37.73
	C2C	77.86	44.20	86.09	75.87	56.87	85.60	**4.50**
	Ours	**92.95**	**82.00**	**93.69**	78.52	67.24	**91.20**	12.56
4B → 14B	Receiver-only	83.70	46.40	92.60	71.80	64.70	91.80	31.16
	T2T	92.34	77.80	92.00	**82.87**	71.17	92.00	56.24
	C2C	82.34	44.20	92.43	72.76	63.00	87.00	**6.64**
	Ours	**93.86**	**86.00**	**94.20**	78.57	**71.96**	**93.60**	21.54
8B → 4B	Receiver-only	82.40	44.20	89.20	65.00	47.70	88.00	9.18
	T2T	89.39	63.20	90.96	**79.39**	**66.46**	87.00	33.91
	C2C	72.48	37.40	86.78	70.19	55.07	77.20	**3.43**
	Ours	**91.81**	**83.40**	**93.00**	77.86	66.30	**89.60**	7.95
8B → 14B	Receiver-only	83.70	46.40	92.60	71.80	64.70	91.80	31.16
	T2T	93.93	75.00	91.74	**84.13**	**72.66**	92.80	67.08
	C2C	82.26	43.20	92.43	73.81	64.65	87.00	**7.42**
	Ours	**94.09**	**85.00**	**94.37**	77.38	70.46	**93.40**	21.79
14B → 4B	Receiver-only	82.40	44.20	89.20	65.00	47.70	88.00	9.18
	T2T	90.60	60.20	89.83	**80.61**	**69.91**	85.00	43.48
	C2C	70.58	36.60	85.83	69.73	53.42	76.40	**5.01**
	Ours	**91.13**	**82.60**	**91.89**	77.66	63.00	**88.80**	10.18
14B → 8B	Receiver-only	81.10	49.20	91.00	72.10	53.00	91.20	19.85
	T2T	91.58	73.00	92.35	**83.43**	**72.90**	90.20	55.50
	C2C	76.35	43.00	89.39	75.99	62.69	85.60	**7.83**
	Ours	**92.95**	**81.40**	**93.77**	78.04	70.38	**92.60**	15.64
Cross-family: Mistral-7B-Instruct-v0.3 → Qwen3-4B								
	Receiver-only	82.40	44.20	89.20	65.00	47.70	**88.00**	9.18
	T2T	72.25	46.20	78.33	66.03	54.20	75.20	15.17
	C2C	51.78	26.60	83.11	66.55	58.21	73.20	14.17
	Ours	**90.37**	**79.60**	**91.30**	**74.54**	**61.98**	83.80	**7.62**

Table 1: **Context-aware accuracy (%)** for six Qwen3 pairs and Mistral-to-Qwen transfer. Ours uses final Phase-II checkpoints. Estimated inference TFLOPs summarize model computation. The cross-family receiver-only row uses the same Qwen3-4B baseline as the within-family evaluation. Additional training protocols are detailed in Appendix F; compute accounting is given in Appendix D.1.

5 EXPERIMENTS

We first evaluate individual cache transfers with and without receiver access to the source input, including a cross-family pair. We then test whether a receiver can combine memories independently constructed by two agents before its question arrives.

Training Data. For the six-pair Qwen3 study, Phase I uses 30,000 reconstruction steps on OpenHermes. Phase II uses 2,000 steps on GSM8K (Cobbe et al., 2021), MATH-algebra (Hendrycks et al., 2021), and ARC-Challenge (Clark et al., 2018), supervised by traces generated by the receiver with gold-answer guidance (Appendix C). The three-agent and cross-family training protocols are detailed in Appendix F.

Experimental Setup. We evaluate the final Phase-II checkpoint for each trained adapter. We evaluate our learned cache transformation across all six directed pairs among Qwen3-4B, 8B, and 14B (Yang et al., 2025), and additionally evaluate Mistral-7B-Instruct-v0.3 → Qwen3-4B. Evaluation is performed on three in-domain tasks (GSM8K, MATH-500, ARC-C) and three held-out multiple-choice benchmarks (MMLU-Redux (Gema et al., 2025), MedQA (Jin et al., 2021), OpenBookQA (Mihaylov et al., 2018)). We compare three baselines against our method: **Receiver-only** (single-agent with the receiver model), **T2T** (text-based communication), and **C2C** (Fu et al., 2025) (learned cache projection and fusion). Both regimes from Section 3 are evaluated: *context-aware*, where the receiver also sees the question, and *context-unaware*, where the receiver only sees the transferred signal. We estimate inference FLOPs from model sizes and token counts (Appendix D.1). Appendix F describes the three-agent and cross-family protocols.

5.1 CONTEXT-AWARE RESULTS

Table 1 tests whether transferred representations help when the receiver already has the question. Our method improves over T2T on all three in-domain benchmarks across all six Qwen3 pairs, including gains of $+6.00$ to $+22.40$ pp on MATH-500. The mathematical reasoning gains suggest that dense

Pair	Method	In-domain			OOD			TFLOPs ↓
		GSM8K	MATH-500	ARC-C	MMLU-Redux	MedQA	OpenBookQA	
Within-family: Qwen3 sender → Qwen3 receiver								
4B → 8B	T2T-context-unaware	51.63	74.40	19.48	21.00	18.85	24.00	37.86
	C2C-context-unaware	1.90	3.00	22.52	21.91	8.96	27.00	14.94
	Ours-context-unaware	**91.43**	**78.80**	**91.38**	**74.59**	**61.82**	**88.80**	**9.42**
4B → 14B	T2T-context-unaware	56.79	**75.20**	23.39	23.06	27.65	27.20	60.83
	C2C-context-unaware	0.00	0.00	10.70	10.26	12.80	16.40	20.05
	Ours-context-unaware	**82.26**	70.60	**86.86**	**62.86**	**53.26**	**82.40**	**17.15**
8B → 4B	T2T-context-unaware	27.98	69.40	23.74	25.18	22.94	23.20	32.75
	C2C-context-unaware	0.38	2.40	10.35	8.20	7.23	10.00	14.76
	Ours-context-unaware	**91.36**	**81.60**	**93.60**	**77.06**	**64.57**	**90.00**	**6.56**
8B → 14B	T2T-context-unaware	30.86	**70.40**	22.96	22.80	27.81	27.40	72.06
	C2C-context-unaware	0.00	0.00	7.39	7.14	6.36	7.60	**9.84**
	Ours-context-unaware	**81.58**	65.00	**88.48**	**54.92**	**57.74**	**83.80**	15.85
14B → 4B	T2T-context-unaware	18.57	**66.60**	20.61	24.52	21.13	20.20	41.95
	C2C-context-unaware	0.68	0.40	4.96	6.11	0.63	6.60	17.93
	Ours-context-unaware	**82.49**	64.00	**88.48**	**67.15**	**59.54**	**85.40**	**8.90**
14B → 8B	T2T-context-unaware	18.80	66.40	22.87	21.41	25.22	26.20	53.80
	C2C-context-unaware	0.00	0.20	2.78	5.34	5.18	2.20	24.77
	Ours-context-unaware	**84.15**	**68.40**	**88.99**	**67.84**	**54.20**	**88.00**	**13.28**
Cross-family: Mistral-7B-Instruct-v0.3 → Qwen3-4B								
	T2T-context-unaware	62.24	34.60	75.43	**61.47**	**51.77**	**74.60**	14.41
	C2C-context-unaware	2.73	3.00	31.91	27.89	25.61	32.20	13.88
	Ours-context-unaware	**64.97**	**54.40**	**77.13**	55.74	41.01	73.00	**6.64**

Table 2: **Context-unaware accuracy (%)** with no receiver access to the source input. T2T sends text; C2C and Ours send KV caches. Cross-family T2T sends only the generated message. Checkpoint and compute conventions follow Table 1; Appendix F describes the cross-family evaluation protocol.

alignment can strengthen reasoning-oriented communication even when the receiver already has the task input.

Our method also outperforms C2C on all six benchmarks across all six Qwen3 pairs. C2C modulates the receiver's cache through projected sender states, head-level weighting, and gated residual fusion (Fu et al., 2025). Our approach instead reconstructs receiver-compatible states across KV groups, then refines their transformations and weights through generation supervision. The gains show that the reconstruction-and-generation interface provides useful representations even when the receiver already observes the input. Generalization remains task-dependent: our method improves over T2T on OpenBookQA across all six pairs, while T2T leads on MMLU-Redux and five of six MedQA comparisons. Mistral-to-Qwen transfer exceeds T2T on all six benchmarks and the Qwen3-4B receiver-only baseline on five, demonstrating that the shared-input benefit extends to Mistral-to-Qwen transfer. Our method uses fewer inference FLOPs than T2T across all six Qwen3 pairs, avoiding sender decoding and receiver re-encoding of generated messages. C2C uses fewer inference FLOPs, but with much lower accuracy.

5.2 Context-unaware Results

Table 2 reports the context-unaware regime, where only the sender observes the question and the receiver must produce the answer from the communicated signal alone. Under the context-unaware condition, the baselines degrade substantially: T2T-context-unaware achieves 18.57–56.79% on GSM8K and remains substantially below our method across the multiple-choice benchmarks, while C2C-context-unaware achieves only 0.00–3.00% on MATH-500 and similarly degrades across the remaining tasks.

In contrast, our learned transformation remains effective when the receiver has no access to the original question. Across the six Qwen3 model pairs, Ours-context-unaware achieves 81.58–91.43% on GSM8K, 64.00–81.60% on MATH-500, 86.86–93.60% on ARC-C, 54.92–77.06% on MMLU-Redux, 53.26–64.57% on MedQA, and 82.40–90.00% on OpenBookQA. The aligned cache therefore retains substantial source-side information even when the receiver must generate without the original question. Inference FLOPs are also lower than in the corresponding context-aware configurations. Removing the question reduces receiver prefill, while differences in generated output length also affect the totals. For example, 8B → 4B achieves $91.36/81.60/93.60\%$ on GSM8K/MATH-500/ARC-C at 6.56 inference TFLOPs, compared with $89.39/63.20/90.96\%$ for context-aware T2T at 33.91 inference TFLOPs. For Mistral-to-Qwen context-unaware transfer, Ours exceeds text communication on the three in-domain benchmarks but trails it on MMLU-Redux, MedQA, and OpenBookQA

A/B → C	Text communication	Latent (ours)	Receiver-only
8B→4B	23.84 / 30.45	**55.97 / 70.15**	35.88 / 47.32
4B→8B	47.53 / 61.53	**57.84 / 71.86**	53.49 / 68.10
14B→4B	23.84 / 30.45	**39.08 / 53.92**	35.88 / 47.32

Table 3: HotpotQA answer EM / F1 (%). Text communication forwards each sender's complete documents with source-agent labels; Latent (ours) transfers aligned caches; Receiver-only receives all documents directly. Bold marks the best EM and F1 within each row.

(Table 2). Cross-family transfer thus preserves the in-domain advantage over text communication, while the held-out results indicate weaker generalization beyond the training tasks.

5.3 Partial-Aware Results: Three-Agent Document Sharing

A user may share useful information with several agents and later ask another agent a question that draws on those earlier interactions. The later agent must combine information held by the others, whose memories were formed before the question was known. We instantiate the partial-aware regime (Section 2.2) with HotpotQA (Yang et al., 2018): agents A and B each observe one supporting document and separate distractors, without seeing the question. Agent C later receives the question and remaining distractors, then answers using information communicated by both agents. C thus has local context but relies on A and B for the supporting evidence.

A and B use the same sender model, and C uses a different model as receiver; A/B → C denotes sender and receiver sizes. We evaluate three configurations at the final checkpoint after task-specific two-memory training. We compare **Text communication**, which forwards each sender's complete documents with source-agent labels; **Latent (ours)**, which transfers independently encoded, aligned KV caches; and **Receiver-only**, a single instance of C given the question and all raw documents together. Evaluation excludes the development pool, leaving 7,241 examples; Appendix F gives additional implementation details.

Table 3 shows that latent communication improves both exact match and F1 over Text communication and Receiver-only in all three configurations. For 8B→ 4B, it achieves 55.97% exact match and 70.15% F1, compared with $23.84/30.45\%$ for text communication and $35.88/47.32\%$ for Receiver-only. The receiver can therefore combine independently constructed memories to answer a question neither sender saw during encoding. Its advantage over a receiver given all documents also suggests that how evidence is represented and integrated matters beyond simply making it available.

6 Conclusion

We presented dense KV-cache alignment for sharing internal representations across language-model agents. The retention study motivates broad information preservation with group-specific transformations and learned gates, refined through generation supervision. Across six Qwen3 pairs, the method improves in-domain accuracy over text communication with fewer inference FLOPs in the context-aware setting, and outperforms both communication baselines in 33 of 36 context-unaware task–pair comparisons in our evaluation. In the partial-aware setting, combining independently encoded document memories improves HotpotQA exact match and F1 over text communication and Receiver-only in all three configurations. Mistral-to-Qwen results further demonstrate transfer across model families. Together, these findings support dense alignment as an interface for reasoning-oriented communication and for sharing information beyond an agent's local observations.

Limitations. The current interface is trained to align the internal representation spaces of heterogeneous agents, meaning that supporting a new model pair requires training a new adapter. Dense caches also incur a larger memory footprint than text, resulting in higher memory usage and communication overhead for longer contexts. In addition, our experiments are limited to topologies with up to three agents and a small number of cross-family settings. Evaluating broader communication topologies and more diverse cross-family configurations will be important future work toward establishing heterogeneous latent communication in more practical scenarios.

AI use statement. We used Codex and Claude Code to assist with code development, result checking, and manuscript editing. Codex also assisted with literature and reference checks, interpretation of results, and feedback on experimental methodology. We used Qwen3 models to generate synthetic, gold-answer-guided training traces for generation supervision, as described in Appendices C and F. The authors take responsibility for the final manuscript, including all AI-assisted text, code, experimental results, and claims.

REFERENCES

Anna Bair, Yixuan Even Xu, Mingjie Sun, and J. Zico Kolter. Compressed Sensing for Capability Localization in Large Language Models. *arXiv preprint arXiv:2603.03335*, 2026. URL https://arxiv.org/abs/2603.03335.

Luís Brito and Carlos Baquero. When Latent Agents Lie: KV-Cache Integrity in Multi-Agent LLM Collaboration. *arXiv preprint arXiv:2606.28958*, 2026. URL https://arxiv.org/abs/2606.28958.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11534–11557, 2025. URL https://aclanthology.org/2025.findings-acl.601/.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 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. *arXiv preprint arXiv:2511.09149*, 2025. URL https://arxiv.org/abs/2511.09149.

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

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. *arXiv preprint arXiv:2510.03215*, 2025. URL https://arxiv.org/abs/2510.03215.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096, 2025. URL https://aclanthology.org/2025.naacl-long.262/.

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

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse. *arXiv preprint arXiv:2608.03893*, 2026. URL https://arxiv.org/abs/2608.03893.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What Disease Does This Patient Have? A Large-Scale Open Domain Question Answering Dataset from Medical Exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421. URL https://www.mdpi.com/2076-3417/11/14/6421.

Haibo Jin, Peng Kuang, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent Primitives: Reusable Latent Building Blocks for Multi-Agent Systems. *arXiv preprint arXiv:2602.03695*, 2026. URL https://arxiv.org/abs/2602.03695.

Yi Li, Dongming Jiang, Yi Zhao, and Bingzhe Li. A Universal Context-Reuse Layer for Cross-Model KV Sharing. *arXiv preprint arXiv:2608.30963*, 2026a. URL https://arxiv.org/abs/2608.30963.

Yiping Li, Zhiyu An, and Wan Du. When Less Latent Leads to Better Relay: Information-Preserving Compression for Latent Multi-Agent LLM Collaboration. *arXiv preprint arXiv:2604.13349*, 2026b. URL https://arxiv.org/abs/2604.13349.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, et al. DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving. *arXiv preprint arXiv:2411.02820*, 2024. URL https://arxiv.org/abs/2411.02820.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International conference on machine learning*, pp. 2681–2690. PMLR, 2017. URL https://proceedings.mlr.press/v70/omidshafiei17a.html.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sehRvaIPQQ.

Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. CacheBridge: Efficient Cross-Model KV Cache Transfer. *arXiv preprint arXiv:2609.00891*, 2026. URL https://arxiv.org/abs/2609.00891.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. *arXiv preprint arXiv:2501.14082*, 2025. URL https://arxiv.org/abs/2501.14082.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire, Jr., and Dejan Kostic. KVComm: Enabling Efficient LLM Communication through Selective KV Sharing. *arXiv preprint arXiv:2510.03346*, 2025. URL https://arxiv.org/abs/2510.03346.

Yichen Tang, Weihang Su, Yujia Zhou, Yiqun Liu, Min Zhang, Shaoping Ma, and Qingyao Ai. Augmenting Multi-Agent Communication with State Delta Trajectory. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10219–10240, 2025. doi: 10.18653/v1/2025.emnlp-main.518. URL https://aclanthology.org/2025.emnlp-main.518/.

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

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. ReMA: Learning to Meta-Think for LLMs with Multi-agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 126621–126667. Curran Associates, Inc., 2025. doi: 10.52202/085713-4225. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b83514e453ff03741a5fb3e5e49f709b-Paper-Conference.pdf.

Chenxu Wang, Yongkun Yang, Boyuan Du, Shiwei Lin, and Huaping Liu. LLM Agents for Deliberative Collaboration: A Study on Joint Decision Making Under Partial Observability. *arXiv preprint arXiv:2607.06157*, 2026. URL https://arxiv.org/abs/2607.06157.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, 2025. URL https://aclanthology.org/2025.acl-long.1170/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Ye Yu, Heming Liu, Haibo Jin, Xiaopeng Yuan, Peng Kuang, and Haohan Wang. Learning to Communicate: Toward End-to-End Optimization of Multi-Agent Language Systems. *arXiv preprint arXiv:2604.21794*, 2026. URL https://arxiv.org/abs/2604.21794.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=LkzuPorQ5L.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. SiriuS: Self-improving Multi-agent Systems via Bootstrapped Reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 124475–124504. Curran Associates, Inc., 2025. doi: 10.52202/085713-4148. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b45279ac82cb017a5f55ea7d3653193a-Paper-Conference.pdf.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought Communication in Multiagent Collaboration. In *Advances in Neural Information Processing Systems*, volume 38, pp. 123389–123418. Curran Associates, Inc., 2025. doi: 10.52202/085713-4113. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b2b502c3629beadda06311386d2c6f73-Paper-Conference.pdf.

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

Jiaru Zou, Rui Pan, Ruizhong Qiu, Pan Lu, Shizhe Diao, Jindong Jiang, Hanghang Tong, Tong Zhang, Markus J. Buehler, Jingrui He, and James Zou. Recursive multi-agent systems. *arXiv preprint arXiv:2604.25917*, 2026. URL https://arxiv.org/abs/2604.25917.

A Extended Related Work

Text-mediated multi-agent systems. Frameworks such as AutoGen (Wu et al., 2024) and MetaGPT (Hong et al., 2024) coordinate specialized language-model agents through dialogue and workflows. Other work improves coordination, agent selection, and communication policies (Chen et al., 2025; Wang et al., 2025; Zhang et al., 2025a; Wan et al., 2025; Zhang et al., 2025b; Zhao et al., 2025). Text is interpretable and model-agnostic, but generated messages require decoding and receiver-side re-encoding. Generated summaries can omit source details; this limitation should be distinguished from forwarding the original evidence verbatim.

Choices of latent interface. Latent communication exchanges internal representations instead of generated natural-language messages (Yu et al., 2026). Interfaces include embedding-level signals (Pham et al., 2024), hidden states or state trajectories (Du et al., 2025; Ramesh & Li, 2025; Tang et al., 2025; Fein-Ashley et al., 2025; Zou et al., 2026), and KV caches (Shi et al., 2025; Fu et al., 2025; Jin et al., 2026; Li et al., 2026b; Liu et al., 2024; Brito & Baquero, 2026). These interfaces can convey representations of observations and intermediate computations. KV caches retain contextual states that participate directly in attention, making them suitable for computation reuse. Selective transmission, compression, projection, and fusion offer different ways to adapt this interface (Shi et al., 2025; Fu et al., 2025; Li et al., 2026b); their quality and computational tradeoffs depend on the receiver's available information and the evaluated protocol.

Cross-model alignment and receiver observations. Partial observability is a longstanding concern in multi-agent learning (Omidshafiei et al., 2017). Copies of the same model share a representation space, but matching architecture alone does not ensure alignment between separately trained models. Cross-model latent interfaces must account for differences in both tensor structure and learned coordinates (Ramesh & Li, 2025; Fu et al., 2025; Liu et al., 2024). Cache-to-Cache uses learned projections and gating to fuse caches under shared input (Fu et al., 2025), while concurrent work studies cross-model KV-cache transfer primarily for context or prefill reuse across model sizes, architectures, and families (Heo et al., 2026; Li et al., 2026a; Qu et al., 2026). Our focus is direct cache alignment that also supports generation without receiver access to the source input. This observation distinction is separate from model heterogeneity: a homogeneous or heterogeneous pair may operate in any of the regimes in Section 2.2. The controlled retention analysis isolates observation availability using self-communication, while the learned adapter addresses transfer across different models.

B Transformer Attention and KV-Cache Preliminaries

Queries, keys, and values. For an input of N tokens, let $\boldsymbol{H}^{(l-1)} \in \mathbb{R}^{N \times d_{\text{model}}}$ denote the hidden states entering attention at layer l. With grouped-query attention (GQA), there are H^q query heads and G KV groups. Each query head h is assigned to a KV group $g(h)$. Before positional encoding, the projections are

$$\boldsymbol{Q}^{(l,h)} = \boldsymbol{H}^{(l-1)} \boldsymbol{W}_Q^{(l,h)}, \tag{16}$$

$$\boldsymbol{K}^{(l,g)} = \boldsymbol{H}^{(l-1)} \boldsymbol{W}_K^{(l,g)}, \qquad \boldsymbol{V}^{(l,g)} = \boldsymbol{H}^{(l-1)} \boldsymbol{W}_V^{(l,g)}, \tag{17}$$

where each projection matrix has shape $d_{\text{model}} \times d_{\text{head}}$, and each projected tensor has shape $N \times d_{\text{head}}$. Standard multi-head attention is the special case $G = H^q$; under GQA, multiple query heads share the same key and value tensors. After applying the model's positional operations to queries and keys, attention for query head h is

$$\text{Attention}^{(l,h)} = \text{softmax}\left(\frac{\boldsymbol{Q}^{(l,h)} (\boldsymbol{K}^{(l,g(h))})^\top}{\sqrt{d_{\text{head}}}} + \boldsymbol{M} \right) \boldsymbol{V}^{(l,g(h))}, \tag{18}$$

where $\boldsymbol{M}$ is the causal mask, with zero entries for allowed positions and $-\infty$ for masked positions. The notation in this equation denotes the position-encoded queries and keys; values do not receive RoPE in the models considered here.

Prefill and autoregressive caching. During prefill, the model processes the input and stores the key and value tensors at every layer. During subsequent decoding, it appends the new token's keys and values and reuses the stored tensors instead of recomputing projections for previous tokens. Attention still accesses the retained history; caching does not eliminate that attention cost. For model $m \in \{S, R\}$, the cache is

$$\mathcal{C}_m(\boldsymbol{X}) = \{(\boldsymbol{K}_m^{(l,g)}(\boldsymbol{X}), \boldsymbol{V}_m^{(l,g)}(\boldsymbol{X}))\}_{l,g}. \tag{19}$$

Including the batch axis, each layer's key or value cache has shape $[B, G, N, d_{\text{head}}]$. Query-head fanout is not an additional cache axis. These dimensions and the learned representation geometry may differ between sender and receiver, motivating the transformation in Section 4.

C Phase-II Trace Construction

For the six-pair Qwen3 study, Phase II trains the cache transformation with a supervised generation loss, so each source context must be paired with a receiver-side target output. We construct these targets as *receiver-self guided traces*: for each sender–receiver pair $\mathcal{A}_S \to \mathcal{A}_R$, the trace generator is the receiver model $\mathcal{A}_R$ itself. This choice avoids teaching the receiver to imitate a different model's wording or reasoning style after cache transfer; the adapter instead learns to produce receiver-native latent states that decode into outputs the receiver is already well suited to generate.

Trace-generation tasks. The Phase-II trace pool contains three in-domain training tasks: GSM8K, MATH-algebra, and ARC-Challenge. These are the same tasks used for multitask Phase-II training in the main experiments. MATH-500, MMLU-Redux, MedQA, and OpenBookQA are not used for Phase-II trace generation; they are held for evaluation, with MATH-500 replacing MATH-algebra as the reported mathematical reasoning benchmark.

Guided trace generation. For each training example $(\boldsymbol{X}, \boldsymbol{y})$, we prompt the receiver model with the source question and the ground-truth answer, then ask it to produce a step-by-step solution that arrives at that answer. The resulting JSONL record contains the question, the gold answer, the generated solution trace, and a trace-mode tag. For multiple-choice tasks such as ARC-Challenge, the trace is relabeled to end with the canonical answer format `Answer: X`, matching the receiver-side evaluation prompt. This relabeling prevents the training target from using a format that the evaluator later rejects or scores inconsistently.

Pair-specific receiver traces. Because the trace generator is the receiver, different cross-model directions use different trace files. For example, 14B→ 4B uses Qwen3-4B-self traces, while 4B→ 14B uses Qwen3-14B-self traces. These traces are generated on the training split with guided decoding and then matched back to training examples by question text before Phase-II optimization.

Mixed-regime receiver prompts. During Phase II, we train the same adapter on both context-aware and context-unaware receiver prompts. We set the receiver prompt to be context-aware with probability 0.5 and context-unaware with probability 0.5. Equivalently, the receiver-side context variable in Section 4 is sampled as $\boldsymbol{X}_R = \boldsymbol{X}$ half of the time and $\boldsymbol{X}_R = \emptyset$ half of the time. The target trace is unchanged across these two cases; only the receiver's direct access to the source context changes. This trains the cache to supplement an observed input or provide source information when that input is unavailable.

In the six-pair Qwen3 study, Phase II starts from the Phase-I reconstruction checkpoint and runs for 2000 optimization steps with cross-entropy on the receiver-self trace tokens.

Gold answers guide training traces only and are not supplied at inference. The three-agent and cross-family data, observation splits, and training budgets are documented separately in Appendix F.

D Estimated Inference Compute and Token Accounting

We estimate inference compute for the six-pair Qwen3 study and break the FLOPs column of Tables 1 and 2 into its four constituent per-side token counts: sender prefill ℓ^S_{pre}, sender decode ℓ^S_{dec}, receiver prefill ℓ^R_{pre}, and receiver decode ℓ^R_{dec}.

D.1 Estimation recipe

Sample composition. For each (pair, method, mode) cell we sample 17 examples from each of six benchmarks (GSM8K, MATH-500, ARC-Challenge, MMLU-Redux, OpenBookQA, MedQA) using a fixed seed, pooled to ≈102 samples per cell. We report mean token counts in the table.

Inference recipe. We record sender and receiver prefill and decode token counts during single-example inference and use these counts to estimate inference compute. Adapter training, trace generation, and communication transport costs are outside the estimate.

FLOPs estimate. We use the standard "2 parameters per token" rule of thumb for the forward pass:

$$\widehat{\mathrm{FLOPs}} \;\approx\; 2\,N_S\,(\ell^S_{\mathrm{pre}} + \ell^S_{\mathrm{dec}}) \;+\; 2\,N_R\,(\ell^R_{\mathrm{pre}} + \ell^R_{\mathrm{dec}}),$$

where N_S, N_R are nominal model-size proxies (4, 8, or 14 billion parameters), as used by the summarizer. The estimator includes only these sender and receiver parameter–token terms. It does not explicitly count adapter operations, attention over the retained history, serialization, or network transfer. These omissions limit the estimate to model computation; end-to-end latency also depends on cache processing and transport.

Token-count instrumentation. Token counts are exact, not estimated from rendered text. We record generated token IDs directly and compute per-sample prefill/decode lengths from model outputs.

Bandwidth disclosure. The saved payload estimates range from approximately 27.2 to 30.2 MB per example for Ours and 3.17 to 3.95 kB for generated text across the original pair/regime aggregates (decimal units). These are estimated message sizes, not measured network traffic. Transport and dense-cache storage can affect deployment cost and latency even when the parameter–token FLOPs estimate is lower.

D.2 Per-side breakdown

Table 4 reports the four per-side token counts plus the unweighted total `All tok` for every (pair, method, mode) cell. The structural mechanism behind each method's compute profile is then visible. Both Ours and C2C eliminate the sender's autoregressive reasoning ($\ell^S_{\mathrm{dec}} = 0$); the receiver prefill is small for both ($\ell^R_{\mathrm{pre}} = 220$ for Ours, 117 for C2C in w/ctx mode) – substantially below T2T's $\ell^R_{\mathrm{pre}} \sim$ 1,100–1,300 tokens of re-encoded sender text. The largest token-count difference between Ours and C2C is in the *receiver decode* length: Ours produces explicit step-by-step reasoning before the answer ($\ell^R_{\mathrm{dec}} \approx$ 400–500 in w/ctx mode), whereas C2C emits a more terse answer ($\ell^R_{\mathrm{dec}} \approx$ 80–170). The lower compute estimate accompanies lower accuracy on most benchmarks (see Tables 1 and 2).

D.3 Structural observations

Three observations follow from Table 4.

(i) Both Ours and C2C eliminate sender decoding. T2T averages approximately 900–1,080 sender-decode tokens under a 2,048-token generation budget. Ours instead encodes the question into a sender KV cache without autoregressive sender decoding ($\ell^S_{\mathrm{dec}} = 0$); C2C similarly transfers a cache without sender generation.

(ii) Receiver prefill is small for both cache-based methods, ∼6× smaller than T2T. Ours has $\ell^R_{\mathrm{pre}} = 220$ (w/ctx) / 63 (context-unaware); C2C has 117 / 14. Both stand in sharp contrast to T2T's $\ell^R_{\mathrm{pre}} \sim$ 1,100–1,300 tokens of re-encoded sender message: the receiver no longer needs to ingest the sender's natural-language reasoning through its embedding pipeline.

Pair	Method	Mode	Spre	Sdec	Rpre	Rdec	All tok	TFLOPs
4B → 8B	Receiver-only	w/ctx	0	0	154	1,087	1,241	19.85
	T2T	w/ctx	180	1,033	1,255	497	2,965	37.73
	T2T-context-unaware	context-unaware	180	1,077	1,180	558	2,995	37.86
	C2C	w/ctx	117	0	117	106	340	**4.50**
	C2C-context-unaware	context-unaware	117	0	14	861	993	14.94
	Ours	w/ctx	184	0	220	473	877	12.56
	Ours-context-unaware	context-unaware	184	0	63	433	681	9.42
4B → 14B	Receiver-only	w/ctx	0	0	154	959	1,113	31.16
	T2T	w/ctx	180	1,020	1,242	423	2,866	56.24
	T2T-context-unaware	context-unaware	180	1,052	1,155	666	3,053	60.83
	C2C	w/ctx	117	0	117	87	321	**6.64**
	C2C-context-unaware	context-unaware	117	0	14	668	800	20.05
	Ours	w/ctx	184	0	220	497	901	21.54
	Ours-context-unaware	context-unaware	184	0	63	497	744	17.15
8B → 4B	Receiver-only	w/ctx	0	0	154	994	1,147	9.18
	T2T	w/ctx	180	1,074	1,296	433	2,984	33.91
	T2T-context-unaware	context-unaware	180	1,037	1,139	520	2,877	32.75
	C2C	w/ctx	117	0	117	77	311	**3.43**
	C2C-context-unaware	context-unaware	117	0	14	1,597	1,728	14.76
	Ours	w/ctx	184	0	220	405	810	7.95
	Ours-context-unaware	context-unaware	184	0	63	388	636	6.56
8B → 14B	Receiver-only	w/ctx	0	0	154	959	1,113	31.16
	T2T	w/ctx	180	1,066	1,288	396	2,930	67.08
	T2T-context-unaware	context-unaware	180	1,073	1,175	682	3,111	72.06
	C2C	w/ctx	117	0	117	81	315	**7.42**
	C2C-context-unaware	context-unaware	117	0	14	270	402	9.84
	Ours	w/ctx	184	0	220	453	857	21.79
	Ours-context-unaware	context-unaware	184	0	63	398	645	15.85
14B → 4B	Receiver-only	w/ctx	0	0	154	994	1,147	9.18
	T2T	w/ctx	180	923	1,145	430	2,677	43.48
	T2T-context-unaware	context-unaware	180	886	988	525	2,579	41.95
	C2C	w/ctx	117	0	117	99	333	**5.01**
	C2C-context-unaware	context-unaware	117	0	14	1,817	1,948	17.93
	Ours	w/ctx	184	0	220	407	811	10.18
	Ours-context-unaware	context-unaware	184	0	63	405	652	8.90
14B → 8B	Receiver-only	w/ctx	0	0	154	1,087	1,241	19.85
	T2T	w/ctx	180	890	1,112	483	2,666	55.50
	T2T-context-unaware	context-unaware	180	887	989	505	2,562	53.80
	C2C	w/ctx	117	0	117	167	402	**7.83**
	C2C-context-unaware	context-unaware	117	0	14	1,329	1,461	24.77
	Ours	w/ctx	184	0	220	435	839	15.64
	Ours-context-unaware	context-unaware	184	0	63	444	692	13.28

Table 4: **Per-side efficiency breakdown.** Mean per-sample token counts on the sender (Spre / Sdec = prefill / decode) and receiver (Rpre / Rdec) sides, pooled across the six benchmarks of Tables 1 and 2. `All tok` = $\ell^S_{\text{pre}} + \ell^S_{\text{dec}} + \ell^R_{\text{pre}} + \ell^R_{\text{dec}}$ (every token that touches either model). TFLOPs computed via the formula in Appendix D.1. Bold marks the lowest TFLOPs cell within each pair (across all methods + modes).

(iii) Receiver decoding accounts for much of the compute difference between Ours and C2C. In w/ctx mode Ours produces explicit step-by-step reasoning before the answer ($\ell^R_{\text{dec}} \sim 400$–$500$); C2C emits a more terse output ($\ell^R_{\text{dec}} \sim 80$–$170$). The lower decode length translates to a $\sim$2–3$\times$ smaller TFLOPs total for C2C w/ctx – but at the cost of substantially lower accuracy on most benchmarks (Tables 1 and 2). In context-unaware mode C2C's receiver decode varies widely across pairs ($\ell^R_{\text{dec}} \in [270, 1{,}817]$); Ours keeps a stable decode profile across all six pairs ($\ell^R_{\text{dec}} \approx 400$), and Ours-context-unaware has the lowest estimated compute among the context-unaware methods in 5 of 6 pairs.

D.4 Compute tradeoffs

Receiver-only in 14B → 4B. For 14B→ 4B, one of two directions with a 4B receiver, the bare Receiver-only baseline (4B alone, $\sim$994 receiver-decode tokens) costs 9.18 TFLOPs. Adding our 14B sender prefill ($\sim$184 tokens) makes Ours marginally more expensive at 10.18 TFLOPs in w/ctx mode, although Ours-context-unaware drops back to 8.90. In every other direction Ours is cheaper than Receiver-only.

Scope of the compute comparison. The Qwen3 accuracy tables and efficiency summaries use the final Phase-II checkpoints. The reported task tradeoffs also differ: T2T is stronger on MMLU-Redux and most MedQA comparisons despite its higher estimated compute. Accuracy is evaluated on the benchmark sets, while compute is estimated on the smaller pooled sample. These measurements

characterize accuracy and model-compute tradeoffs; they do not measure latency or total deployment cost.

E Compressed-Sensing Analysis Across Regimes

Section 3 uses post-hoc compressed sensing (CS) (Bair et al., 2026) to rank sender query heads by their sensitivity to ablation, then evaluate accuracy as a function of retained KV groups. This section documents the setup and results behind Figure 4.

E.1 Self-communication setup

Sender and receiver are the same Qwen3-4B model (homogeneous, identity KV pass-through): the sender encodes the question into a KV cache with zero sender decoding, and the receiver consumes that cache directly. Qwen3-4B has 36 layers $\times$ 32 query heads $= 1152$ query heads, organized into $36 \times 8 = 288$ KV groups under GQA (each KV head shared across 4 query heads). The ranking stage ablates sender query-head outputs before the attention output projection; the retention stage instead zeros non-selected groups in the transmitted KV cache. Receiver attention is unchanged. These are distinct interventions: the head ranking is used as a heuristic for cache selection.

From query-head importance to KV-group importance. The CS measurements (Stage 1 below) perturb individual *query heads*, so the Lasso problem is naturally posed at the 1152-query-head granularity. The transmitted KV cache itself, however, has only 288 KV groups. To go from one to the other, we sum the 4 query-head Lasso coefficients within each GQA group:

$$\text{score}_{\text{KV}}^{(l,h_{\text{KV}})} \;=\; \sum_{h_Q \,\in\, \text{group}(h_{\text{KV}})} \hat{x}_{l,h_Q},$$

which gives a per-(layer, KV-head) score. More-negative scores indicate greater estimated importance under the ablation-mask convention below; Stage 2 retains the K groups with the smallest scores.

E.2 Stage 1: CS head ranking

We take $M = 200$ binary ablation masks $\Phi \in \{0,1\}^{M\times N}$ over the $N = 1152$ query heads, with each row ablating approximately 5% of heads. Here $\Phi_{mi} = 1$ means head i is ablated in measurement m, and zero means it is left intact. Ablation zeros the selected query-head output before the sender attention output projection. For each mask m, we run the full sender$\rightarrow$receiver pipeline on 100 evaluation samples and record the resulting metric y_m (accuracy on GSM8K / MATH-algebra / ARC-Challenge), then center it by the unablated baseline, $\tilde{y}_m = y_m - y_{\text{baseline}}$. We then solve a Lasso regression for the per-head importance vector $\mathbf{x} \in \mathbb{R}^N$:

$$(\hat{\mathbf{x}}, \hat{b}) \;=\; \arg\min_{\mathbf{x}\in\mathbb{R}^N,\, b\in\mathbb{R}} \;\frac{1}{2M}\left\|\tilde{\mathbf{y}} - \Phi\,\mathbf{x} - b\mathbf{1}\right\|_2^2 \;+\; \alpha\,\|\mathbf{x}\|_1, \qquad \alpha = 10^{-4},$$

where b is a fitted intercept. The additive surrogate is a ranking heuristic, not an assumption that attention heads act independently. We rank heads by $-\hat{x}_i$ (most-negative $\hat{x}_i$ indicates highest importance, since masking it most degrades accuracy).

E.3 Stage 2: KV-group retention sweeps

We evaluate the ranking by retaining selected groups in the transmitted cache and zeroing the remaining key and value entries. Receiver attention runs unmodified. The context-aware sweep includes $K \in \{5, 10, 20, 50, 200\}$; Figure 4 displays $K = 10, 20, 50$ together with the full-cache result at $K = 288$ and a receiver-only reference at $K = 0$. The context-unaware curve displays $K \in \{0, 10, 20, 50, 100, 150, 200, 250, 288\}$. The ranking is shared across regimes, but the displayed retention grids differ.

The context-aware receiver sees the question as well as the filtered cache; the context-unaware receiver is prompted to answer using the cache without seeing the question. The $K = 0$ context-aware points are separately recorded receiver-only references: GSM8K and ARC-Challenge use the original

paper's Qwen3-4B baseline, while MATH-algebra uses the corresponding historical single-agent result. The receiver-only references and retention scores come from separate evaluation runs, so differences from $K = 0$ do not provide a controlled estimate of communication benefit.

E.4 Random-filter baseline

For the random-filter markers in Figure 4, we replace the CS-Lasso ranking with a uniform random selection of K KV groups, repeat with 3 seeds, and report the mean. The comparison tests whether the fitted ranking improves on random selection at the same retention level.

E.5 Ranking limitations

The Lasso surrogate uses 200 measurements for 1,152 query-head coefficients. Many lower-ranked coefficients are zero or close to zero, so aggregation and tie-breaking can affect which KV groups are retained as K increases. A sparse fit alone does not establish statistical reliability or a recoverable number of task-relevant heads. We therefore interpret the curves as sensitivity to this selection procedure, rather than evidence of a minimal sufficient cache or independent causal head contributions.

E.6 Context-Aware and Context-Unaware Results

E.6.1 Context-aware regime

With $K = 10$ retained groups, accuracy is $0.883/0.873/0.699$ on GSM8K/ARC-Challenge/MATH-algebra, compared with full-cache values of $0.917/0.880/0.698$. Random selection at $K = 10$ gives similar results ($0.884/0.863/0.698$). The ranking has a larger effect at $K = 20$ and 50: at $K = 50$, CS selection gives $0.876/0.862/0.577$, versus $0.605/0.583/0.484$ for random selection. These results show that high task accuracy can survive removal of most transmitted groups when the receiver also observes the input.

E.6.2 Context-unaware regime

Without receiver access to the question, accuracy remains low through $K = 150$ and rises at higher retention. At $K = 250$, the recorded GSM8K/ARC-Challenge/MATH-algebra scores are $0.834/0.904/0.686$, compared with $0.880/0.901/0.694$ at $K = 288$. Each regime has its own full-cache reference. The comparison indicates greater sensitivity to KV-group removal when the cache supplies the source input.

E.6.3 Implication for alignment

The receiver's observations change what the transferred cache must provide. This motivates reconstructing receiver-compatible states across the cache, alongside generation supervision that trains the receiver to use them. KV-group transformations, gates, and positional handling implement this interface. The retention study motivates the overall design; component-level contributions require separate ablations.

F Additional Evaluation Protocols

F.1 Three-agent document memories

The HotpotQA distractor split assigns one supporting document and separate distractors to each of A and B; C receives the remaining distractors and, later, the question. Supporting annotations define the partition but are not shown in prompts. For Latent (ours), senders encode only their own documents, without the question, answer, or the other sender's evidence. Each cache is transformed independently with the shared pair-specific adapter; transformed content caches are concatenated before receiver positional encoding. This is a three-agent topology with two instances of the same sender model. The partition requires two supporting documents and at least three distractors, leaving 7,368 of 7,405 validation examples.

The adapters undergo 30,000-step OpenHermes-2.5 reconstruction with a 500,000-example source pool (effective batch 32, learning rate 10^{-3}, 1,000 warmup steps). Phase II uses 5,000 two-memory steps (effective batch 16, learning rate 10^{-4}), with 20,042 training examples retained from a 22,000-example candidate pool after evidence and heldout filtering; 128 training examples are held out. Training supervision is Qwen3-4B-generated, gold-guided, greedy text with a 256-token budget; it is shared across pairs, providing a common supervision source for all receiver sizes. Gold answers are used for training supervision only. The fixed step-5,000 checkpoint is evaluated with greedy decoding and a 512-token answer budget.

Text communication (Section 5.3) forwards A's and B's complete observed documents, labeled by source agent; no sender generation, evidence selection, or summarization is performed. Latent (ours) transfers the two independently encoded, aligned caches. Receiver-only is a single receiver given the question and all raw documents together, including both supporting documents, without sender agents. Text communication and Receiver-only therefore contain the same evidence in different presentations and are shared among pairs with the same receiver size. We report answer exact match and token F1, with F1 computed from token-count overlap. These answer-only metrics use the repository's text normalization.

F.2 Cross-family protocol

The sender is Mistral-7B-Instruct-v0.3 and the receiver is Qwen3-4B. Phase I compares caches at token positions ending at shared character boundaries on identical text, excluding unmatched positions. Shared character boundaries provide reconstruction targets at matching text prefixes despite different tokenizations. The run uses 30,000 reconstruction steps from a 500,000-example OpenHermes pool, followed by 2,000 generation-training steps on GSM8K, MATH-algebra, and ARC-Challenge. Generation training uses 10,335 examples matched to supervision traces by question text. Evaluation uses the final Phase-II checkpoint.

G Qualitative Latent Space Visualization

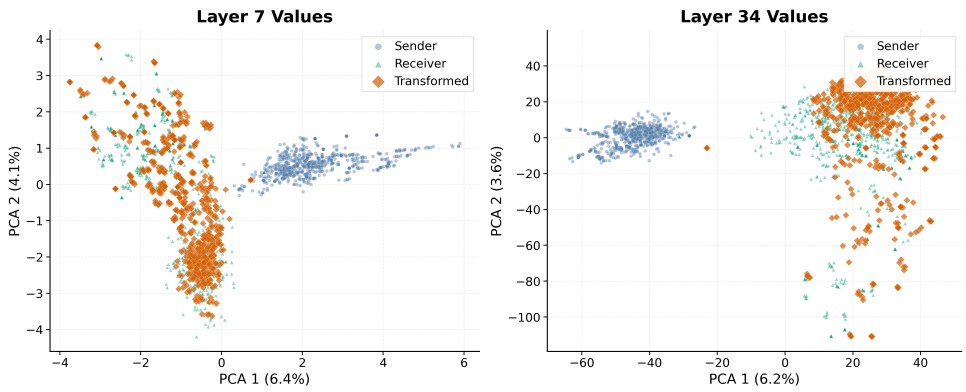

Figure 7: Two-dimensional PCA views of transformed sender and receiver-native value vectors for the same inputs at layers 7 and 34. The projections illustrate the geometry of aligned and receiver-native representations at two depths.

Figure 7 visualizes per-token value vectors in the receiver cache space. The projections suggest similarity between transformed and native representations at the displayed layers. As two-dimensional summaries, they complement the task-level evaluation but do not measure how much information the full cache preserves.