PREFILL-FREE CROSS-FAMILY KV CACHE TRANSFER FOR HETEROGENEOUS MULTI-AGENT LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent multi-agent LLM systems increasingly combine heterogeneous models for specialized agent roles. However, text-based communication requires each receiver to prefill shared context already processed by the sender. Reusing the sender's key-value (KV) cache avoids this redundancy, but prefill-free transfer across model families must handle differences in tokenization, model depth, and KV representations. To address these issues, we propose *HeteroFold*, a prefill-free cross-family KV cache transfer method that keeps both the sender and receiver frozen. HeteroFold aligns model structures, maps the sender cache into the receiver space, and calibrates it to preserve receiver behavior. Across six transfer directions, HeteroFold achieves the best cache-transfer performance on all four long-context benchmarks and most short-context settings. It also matches text-based communication on the multi-agent benchmark. At 32K context length, Llama-3.1-8B→Ministral-3-14B transfer is 10.7× faster than Native Prefill and 1.18–1.47× faster than the state-of-the-art prefill-free baselines, Dense Latent and KV Ridge. These results show that HeteroFold enables efficient cross-family KV reuse without receiver prefill.

1 INTRODUCTION

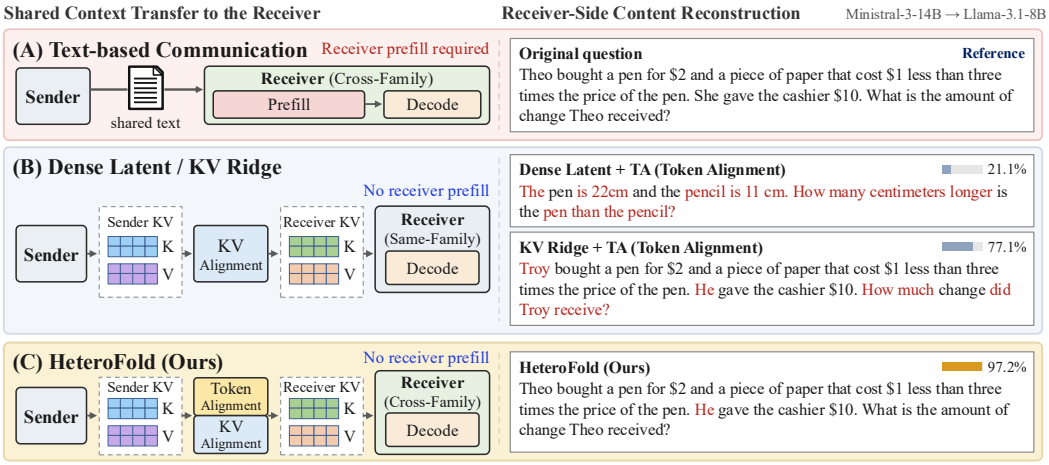


Figure 1: Comparison of communication settings and receiver-side question reconstruction. HeteroFold enables prefill-free cross-family KV cache transfer. For reconstruction, the sender prefills the question, transfers its KV cache, and the receiver is prompted to restate the question without the original text. Panel (B) shows the original same-family setting of prior cache-transfer methods; reconstruction uses TA for cross-family Ministral-3-14B→Llama-3.1-8B transfer on GSM8K. Red text marks differences from the reference. Percentages report ordered overlap of this prompt.

Large language models (LLMs) are increasingly used as collaborating agents in multi-agent systems (MAS), which decompose tasks across specialized roles (Hong et al., 2024), coordinate through

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

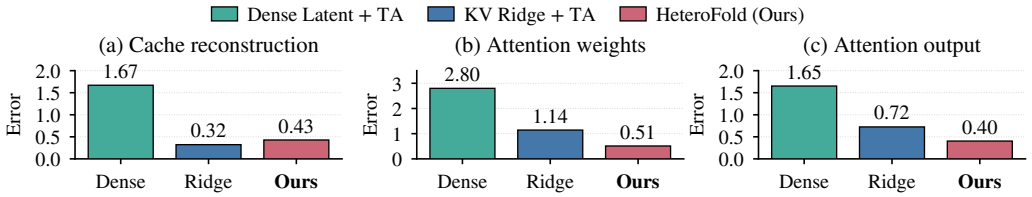


Figure 2: Cache and attention errors after transfer, averaged over all six transfer directions on Qasper. Panels (a,c) report normalized squared error for K/V reconstruction and attention output; (b) reports attention-weight KL divergence. Both baselines use Token Alignment (TA); lower is better.

conversation (Wu et al., 2024), and make decisions through consensus (Chen et al., 2023; Lee et al., 2026). We refer to systems that combine models from different families or scales as *heterogeneous multi-agent systems*. These systems can assign models to roles based on their capabilities and computational costs (Wang et al., 2024); for example, X-MAS (Ye et al., 2025) uses different model families for solver, evaluator, and aggregator roles. However, heterogeneous agents often exchange shared context. Text-based communication requires each receiver to prefill it again and rebuild its KV cache (Woo et al., 2026), adding redundant computation and latency as context length grows (Heo et al., 2026).

Cross-family KV reuse can remove repeated receiver prefill, but must handle mismatches in tokenization, model depth, and KV representations. Figure 1 illustrates how these mismatches can affect receiver-side text reconstruction, while Figure 2 shows that cache reconstruction error alone does not reflect receiver behavior: KV Ridge (Heo et al., 2026) achieves lower reconstruction error than HeteroFold, yet distorts receiver attention. Therefore, effective transfer must preserve not just the cache values, but the downstream receiver computation they induce.

We propose **HeteroFold**, to our knowledge the first receiver-prefill-free cross-family KV cache transfer method with explicit cross-tokenizer alignment. HeteroFold addresses **heterogeneity** across model families by aligning tokens and layers, mapping K/V features to receiver statistics, and calibrating the transferred cache against native receiver attention and outputs. The learned corrections are **folded** into fixed affine maps, keeping both models frozen and enabling direct decoding without receiver prefill or an additional inference module.

Across six transfer directions among Llama, Qwen, and Ministral, HeteroFold outperforms the evaluated cache-transfer baselines on all long-context benchmarks and in most short-context benchmarks. On HIDDENBENCH, it achieves group decision accuracy comparable to text-based communication without receiver prefill. At 32K context length, Llama-3.1-8B→Ministral-3-14B transfer is $10.7\times$ faster than Native Prefill and 1.18–$1.47\times$ faster than the state-of-the-art prefill-free baselines, KV Ridge and Dense Latent.

Our contributions are:

- We identify tokenizer mismatch as a key obstacle to prefill-free cross-family KV cache reuse and introduce *Token Alignment (TA)*, which establishes sender–receiver correspondence through shared character boundaries.
- We develop a cross-family K/V mapping that handles differences in model depth, KV structure, and feature distributions. It combines multi-layer sender features, cross-head mixing, and Recolor to construct receiver-compatible K/V states while keeping both language models frozen.
- We show that KV reconstruction alone does not preserve receiver behavior and introduce receiver-aware calibration that directly matches attention patterns and outputs. The learned corrections fold into fixed affine maps, enabling direct decoding without receiver prefill.

2 Related Work

Cross-family communication. Standard text communication, denoted TextMas following Zou et al. (2026b), supports cross-family exchange but requires each receiver to prefill the exchanged text and rebuild its KV cache. C2C (Fu et al., 2026) supports cross-model cache communication, but

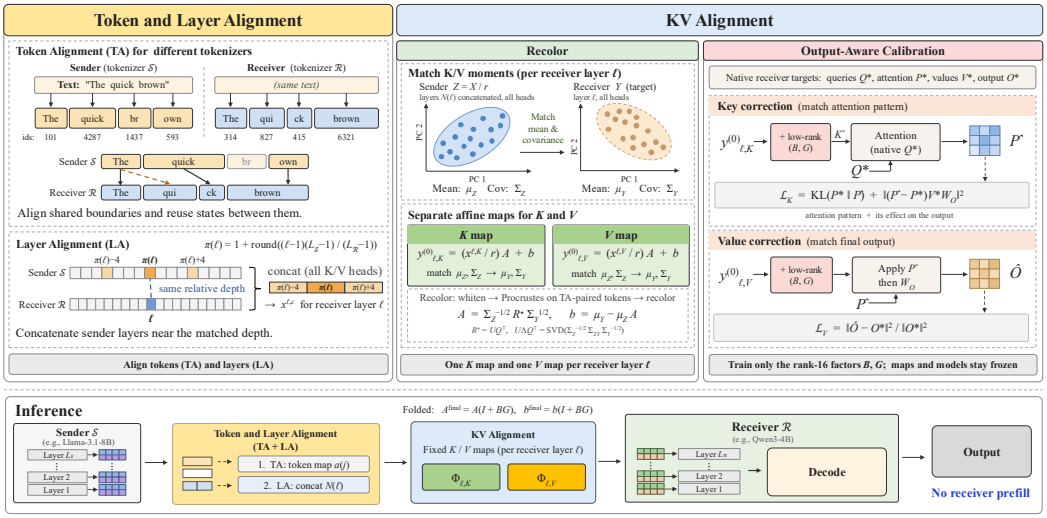


Figure 3: HeteroFold aligns tokens and layers, initializes separate K/V maps, and calibrates receiver attention and outputs. Corrections fold into affine maps; both models remain frozen.

requires a prompt cache generated by the receiver. Therefore, the receiver must process the shared prompt before cache transfer, so C2C does not eliminate receiver prefill. RECURSIVEMAS (Zou et al., 2026a) communicates across heterogeneous models through hidden representations, but the transferred information is still processed by the receiver's layers rather than provided as a decode-ready KV cache.

Prefill-free communication. LATENTMAS (Zou et al., 2026b) avoids standard text communication by sharing latent states and layer-wise caches, but assumes cache compatibility between the communicating models. This assumption does not directly cover models that differ in tokenization, depth, or KV structure. Dense Latent (Chen et al., 2026) and KV Ridge (Heo et al., 2026) map sender cache states into receiver cache states and avoid receiver prefill. However, their reported evaluations remain within model families with compatible tokenization. As a result, they do not address the token correspondence, layer alignment, and KV representation mismatches introduced by cross-family transfer.

Previous research has explored partial solutions for KV-cache transfer: cross-family communication still requires receiver-side processing, while prefill-free cache transfer is applicable when sender and receiver structures are compatible. In contrast, HeteroFold addresses both by aligning different tokenizers and model depths, mapping sender K/V features into the receiver space, and preserving receiver behavior. This enables direct cross-family KV cache transfer without receiver prefill.

3 METHOD: HETEROFOLD

Let $\mathcal{S}$ and $\mathcal{R}$ denote frozen sender and receiver models with $L_{\mathcal{S}}$ and $L_{\mathcal{R}}$ layers. Given context x, HeteroFold maps $\mathcal{C}_{\mathcal{S}}(x)$ to a receiver-compatible cache $\widehat{\mathcal{C}}_{\mathcal{R}}(x)$ without receiver prefill. We map keys before key normalization and RoPE (Su et al., 2023) and values after projection; the receiver then applies its native key normalization and RoPE. HeteroFold consists of token and layer alignment, Recolor KV mapping, and output-aware calibration (Figure 3).

3.1 TOKEN AND LAYER ALIGNMENT

Token Alignment (TA). TA aligns tokens through shared character-end boundaries. For an unmatched receiver boundary, we use the sender state at the latest preceding shared boundary, or the first sender state before the first match. In many-to-one cases, sender states are not averaged. After K/V mapping, the receiver applies RoPE using its own position indices. The following example illustrates different tokenizations of "The unbelievable result.":

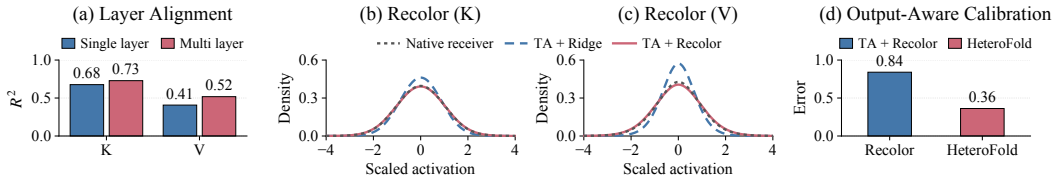


Figure 4: Motivation for HeteroFold components on 100 held-out HotpotQA prompts for Ministral-3-14B→Llama-3.1-8B. (a) Layer Alignment: multi-layer inputs yield higher R^2 for predicting native receiver K/V than a single layer. (b,c) Recolor: K/V activation densities at receiver layer 16, standardized using native per-feature statistics. (d) Output-Aware Calibration: calibration reduces the attention-output error after Recolor.

Model	**Tokens**	**End positions**
Qwen3	`The` | `unbelievable` | `result` | `.`	3, 16, 23, 24
Ministral-3	`The` | `unbel` | `iev` | `able` | `result` | `.`	3, 9, 12, 16, 23, 24

For Qwen3→Ministral-3, receiver boundaries at characters 9 and 12 reuse the Qwen state ending at character 3. In the reverse direction, the Qwen token ending at character 16 uses the Ministral state ending at the same boundary, matching the same causal prefix endpoint rather than averaging intermediate states.

Layer Alignment (LA). Across model families, receiver-relevant information may be distributed across multiple sender layers. Figure 4(a) shows that aggregating neighboring sender layers improves prediction of native receiver K/V representations over a single layer. We use proportional depth alignment with three sender layers at offsets $(-4, 0, +4)$ around the matched layer:

$$\pi(\ell) = 1 + \mathrm{round}\left((\ell - 1)\frac{L_{\mathcal{S}} - 1}{L_{\mathcal{R}} - 1}\right), \quad \mathcal{N}(\ell) = \mathrm{clip}_{[1, L_{\mathcal{S}}]}\big(\pi(\ell) - 4, \pi(\ell), \pi(\ell) + 4\big) \tag{1}$$

For each role $c \in \{K, V\}$, let $c_i^{\mathcal{M},\ell} \in \mathbb{R}^{d_{\mathcal{M}}^{\mathrm{KV}}}$ denote the flattened role-c KV feature at token i. For each aligned token pair (i_n, j_n), we concatenate the selected sender features:

$$x_n^{\ell,c} = [c_{i_n}^{\mathcal{S},t_1} \| c_{i_n}^{\mathcal{S},t_2} \| c_{i_n}^{\mathcal{S},t_3}], \quad y_n^{\ell,c} = c_{j_n}^{\mathcal{R},\ell}, \quad (t_1, t_2, t_3) = \mathcal{N}(\ell) \tag{2}$$

where $y_n^{\ell,c}$ is the receiver target. Flattening allows cross-head mapping even when head counts differ.

3.2 Moment-Matched Recoloring

Token-wise reconstruction objectives (Chen et al., 2026; Heo et al., 2026) can preserve low reconstruction error while altering receiver attention and outputs (Figure 2). In contrast, Figures 4(b,c) show that Recolor better matches native K/V distributions than Ridge (ℓ_2-regularized least squares). Therefore, we match receiver feature means and covariances through `Recolor`.

For each receiver layer and role $c \in \{K, V\}$, we collect N aligned token pairs across the calibration prompts and stack their features into $X \in \mathbb{R}^{N \times \tilde{d}_{\mathcal{S}}}$ and $Y \in \mathbb{R}^{N \times d_{\mathcal{R}}}$, where $\tilde{d}_{\mathcal{S}} = 3d_{\mathcal{S}}^{\mathrm{KV}}$ and $d_{\mathcal{R}} = d_{\mathcal{R}}^{\mathrm{KV}}$. We omit (ℓ, c) below when unambiguous and let x_n denote the nth row of X. For each layer and role, we compute one scalar RMS over all sender samples and feature dimensions:

$$r = \sqrt{\frac{\sum_{n=1}^{N} w_n \|x_n\|_2^2}{\tilde{d}_{\mathcal{S}} \sum_{n=1}^{N} w_n}}, \quad Z = X/r \tag{3}$$

Here, w_n is the sample weight used for moment estimation, and r remains fixed during inference. Let $\mu_Z, \mu_Y, \Sigma_Z, \Sigma_Y$, and Σ_{ZY} denote the corresponding weighted means, covariances, and cross-covariance.

We whiten both feature spaces to remove their original scales and correlations, and align the paired features using Procrustes alignment (Schönemann, 1966). The thin singular value decomposition gives:

$$\Sigma_Z^{-1/2} \Sigma_{ZY} \Sigma_Y^{-1/2} = U \Lambda Q^\top, \quad R^\star = U Q^\top \tag{4}$$

After alignment, we restore the receiver covariance and mean to obtain the *Recolor* initialization:

$$A = \Sigma_Z^{-1/2} R^\star \Sigma_Y^{1/2}, \quad b = \mu_Y - \mu_Z A, \quad y^{(0)} = (x/r)A + b \tag{5}$$

For $\tilde{d}_\mathcal{S} \geq d_\mathcal{R}$ and nonsingular covariances, the unregularized map satisfies $\mu_Z A + b = \mu_Y$ and $A^\top \Sigma_Z A = \Sigma_Y$. Numerical stabilization makes covariance matching approximate in practice.

3.3 Output-Aware Calibration

Recolor aligns the cache's global statistical structure but does not ensure that the transferred cache reproduces native receiver attention patterns and outputs. As illustrated in Figure 4(d), attention-output error persists even after applying Recolor. To reduce this residual attention-output error, we adapt output-aware KV calibration for KV quantization (Yun et al., 2026) to cross-family cache transfer. We calibrate K and V separately according to their effects on receiver computation.

For fixed receiver queries, let P^* and $\widehat{P}$ denote the native and transferred attention weights, V^* and $\widehat{V}$ the corresponding values, and W_O the receiver output projection. The native and transferred attention outputs are $O^* = (P^*V^*)W_O^\top$ and $\widehat{O} = (\widehat{P}\widehat{V})W_O^\top$. Their difference can be written as:

$$\widehat{O} - O^* = \underbrace{[(\widehat{P} - P^*)V^*]W_O^\top}_{\text{key-induced change}} + \underbrace{[\widehat{P}(\widehat{V} - V^*)]W_O^\top}_{\text{value-induced change}} \tag{6}$$

This separates changes in attention patterns from changes in the retrieved values, motivating distinct calibration objectives for K and V.

For each receiver layer ℓ and role $c \in \{K, V\}$, we add a rank-ρ_c correction to the Recolor output $y_{\ell,c}^{(0)}$ from Eq. (5):

$$\widehat{y}_{\ell,c} = y_{\ell,c}^{(0)} + (y_{\ell,c}^{(0)} B_{\ell,c})G_{\ell,c}, \quad B_{\ell,c} \in \mathbb{R}^{d_\mathcal{R} \times \rho_c}, \quad G_{\ell,c} \in \mathbb{R}^{\rho_c \times d_\mathcal{R}} \tag{7}$$

The key correction modifies $\widehat{P}$ in the first term of Eq. (6), while the value correction modifies $\widehat{V}$ in the second.

We optimize only B and G, with G initialized to zero; the initial maps and both models remain frozen. Calibration uses native receiver queries, attention weights, values, and outputs at tokens overlapping the prompt question span, without gold answers or generated continuations.

Key and Value Objectives. Corrected keys produce $\widehat{P}$ using native receiver queries, key normalization, RoPE, and attention scaling. For query head h and probe p, let $\Delta O_{\ell,h,p}^K = [(\widehat{P}_{\ell,h,p} - P_{\ell,h,p}^*)V_{\ell,g(h)}^*]W_{O,\ell,h}^\top$ denote the key-induced output change, where $g(h)$ is the corresponding KV head under grouped-query attention (Ainslie et al., 2023). Native values remain fixed for the key objective. For the value objective, we hold the mapped attention weights $\widehat{P}$ fixed and optimize only the mapped values, forming the full output $\widehat{O}_\ell(\widehat{P}, \widehat{V})$.

$$\mathcal{L}_{\ell,K} = \mathbb{E}_{x,h,p}\left[D_{\mathrm{KL}}(P_{\ell,h,p}^* \| \widehat{P}_{\ell,h,p}) + \frac{\|\Delta O_{\ell,h,p}^K\|_2^2}{d_\mathcal{R}^{\text{model}}}\right] \quad \mathcal{L}_{\ell,V} = \mathbb{E}_x\left[\frac{\|\widehat{O}_\ell - O_\ell^*\|_F^2}{\max(\|O_\ell^*\|_F^2, \epsilon)}\right] \tag{8}$$

Here, $d_\mathcal{R}^{\text{model}}$ is the receiver hidden dimension. The key objective matches attention patterns and their effect after the output projection, while the value objective refines the values under the fixed attention pattern without updating the keys. Both losses are averaged across layers, with equal weight assigned to each example. We optimize them jointly for four epochs and select the checkpoint with the lowest combined held-out loss.

Folding for inference. After calibration, we fold the learned corrections into the initial affine maps:

$$A^{\text{final}} = A(I + BG), \quad b^{\text{final}} = b(I + BG), \quad \Phi_{\ell,c}(x_j) = (x_j/r)A^{\text{final}} + b^{\text{final}} \tag{9}$$

Inference uses one fixed affine map per receiver layer and role, with no separate correction module. Mapped features are reshaped into receiver KV heads, with receiver key normalization and RoPE applied to K. Appendix B.2 gives the optimization settings.

Table 1: Cross-family transfer results across six directions. Dense Latent and KV Ridge use same-index token pairing, while +TA replaces only token correspondence with our shared character-boundary alignment. TextMas uses lossless text communication with native receiver prefill. Higher is better. Bold indicates the best cache-transfer result within each direction and benchmark; TextMas is excluded from emphasis.

Transfer direction	Method	Long Context QA				Short Context QA				
		Qasper (F1 ↑)	HotpotQA (F1 ↑)	LoCoMo (F1 ↑)	QuALITY (Acc. ↑)	ARC-C (Acc. ↑)	MMLU (Acc. ↑)	WinoGrande (Acc. ↑)	HellaSwag (Acc. ↑)	GSM8K (Acc. ↑)
Qwen3-4B → Llama-3.1-8B	TextMas	44.64	57.54	52.33	75.07	82.76	66.74	65.82	70.78	85.67
	Dense Latent	2.83	0.60	0.65	23.15	24.74	24.11	50.28	48.67	0.91
	Dense Latent + TA	3.17	1.37	1.68	26.46	27.65	26.56	50.75	50.17	4.09
	KV Ridge	2.78	0.74	1.41	25.17	24.40	30.36	50.36	51.40	0.53
	KV Ridge + TA	16.06	34.04	13.56	54.60	76.79	61.42	52.49	59.57	43.21
	HeteroFold (Ours)	**29.12**	**38.48**	**29.66**	**62.85**	**78.92**	**63.28**	**57.30**	**65.60**	**55.12**
Ministral-3-14B → Qwen3-4B	TextMas	43.07	56.05	38.92	70.61	87.80	67.95	55.09	41.54	90.14
	Dense Latent	2.56	0.22	0.22	24.98	25.00	27.99	51.22	44.66	1.82
	Dense Latent + TA	3.66	1.32	0.43	25.89	26.37	31.46	49.57	44.34	15.16
	KV Ridge	2.77	1.45	1.73	25.41	42.15	44.27	50.99	37.45	7.05
	KV Ridge + TA	39.02	48.24	14.71	73.20	88.14	71.73	**54.78**	**45.12**	87.11
	HeteroFold (Ours)	**42.00**	**52.74**	**35.59**	**75.55**	**88.99**	**72.14**	53.83	41.46	**89.23**
Llama-3.1-8B → Qwen3-4B	TextMas	43.07	56.05	38.92	70.61	87.80	67.95	55.09	41.54	90.14
	Dense Latent	2.72	1.40	0.67	25.31	24.49	27.00	50.67	42.96	0.76
	Dense Latent + TA	4.35	1.39	0.58	25.50	25.00	27.24	51.62	43.53	6.52
	KV Ridge	2.88	0.94	1.00	26.94	26.62	31.09	50.43	36.93	1.59
	KV Ridge + TA	31.63	25.12	21.91	59.59	74.32	61.25	51.38	**44.54**	70.66
	HeteroFold (Ours)	**35.99**	**49.44**	**34.83**	**62.90**	**77.05**	**62.39**	**52.88**	41.74	**70.74**
Qwen3-4B → Ministral-3-14B	TextMas	45.59	64.56	52.01	83.32	92.66	75.63	61.56	68.33	94.69
	Dense Latent	3.19	1.66	1.23	26.51	36.26	32.07	50.04	50.62	3.87
	Dense Latent + TA	5.07	4.26	3.37	38.97	42.66	41.31	50.20	50.14	28.81
	KV Ridge	2.17	0.44	1.43	27.85	26.45	26.78	51.14	52.47	1.74
	KV Ridge + TA	7.75	17.09	9.33	62.94	84.13	63.41	53.12	61.74	60.20
	HeteroFold (Ours)	**20.57**	**42.65**	**17.75**	**66.30**	**85.49**	**66.24**	**55.88**	**65.22**	**77.26**
Ministral-3-14B → Llama-3.1-8B	TextMas	44.64	57.54	52.33	75.07	82.76	66.74	65.82	70.78	85.67
	Dense Latent	2.72	0.79	0.78	24.83	31.14	28.34	51.62	51.36	1.74
	Dense Latent + TA	3.47	3.26	2.02	28.76	29.35	35.36	52.09	50.86	9.86
	KV Ridge	2.67	0.54	0.26	23.54	26.45	25.62	51.62	51.61	1.67
	KV Ridge + TA	19.09	37.48	39.07	74.74	88.31	68.56	52.64	65.61	53.30
	HeteroFold (Ours)	**33.72**	**50.29**	**43.15**	**79.00**	**90.36**	**71.76**	**63.22**	**69.16**	**63.46**
Llama-3.1-8B → Ministral-3-14B	TextMas	45.59	64.56	52.01	83.32	92.66	75.63	61.56	68.33	94.69
	Dense Latent	3.13	1.33	1.02	25.79	33.19	28.00	49.88	49.90	1.44
	Dense Latent + TA	6.71	4.88	4.43	29.72	38.57	29.58	50.51	50.41	18.65
	KV Ridge	1.24	0.25	0.80	23.78	22.70	26.16	51.38	48.34	0.99
	KV Ridge + TA	11.41	13.33	15.41	55.66	75.09	57.53	54.93	47.08	47.99
	HeteroFold (Ours)	**21.58**	**39.00**	**30.43**	**72.05**	**81.14**	**65.92**	**57.14**	**66.41**	**72.33**

4 Experimental Results

Baselines. We evaluate all six directed transfers among Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-4B (Yang et al., 2025), and Ministral-3-14B-Instruct (Mistral AI, 2025), with all models frozen in BF16. TextMas denotes standard text communication. In single-hop QA, the sender passes the original prompt unchanged, and the receiver performs native prefill to build its own KV cache, providing a text-based reference without cache transfer or mapping error.

For Dense Latent (Chen et al., 2026) and KV Ridge (Heo et al., 2026), we reimplement their methods with default mapping and layer-selection settings. Dense Latent uses proportional layer alignment, while KV Ridge selects $k = 8$ sender layers per receiver layer using calibration R^2 and fits separate per-head K/V maps. Since their original settings assume same-family models with a shared tokenizer, our direct cross-family variants pair sender token i with receiver token i. The +TA variants

replace only this token correspondence with our shared character-boundary alignment. Full settings are given in Appendix A.3.

Tasks. We use four long-context QA benchmarks and five short-context QA benchmarks:

- Long-context QA: Qasper (Dasigi et al., 2021), HotpotQA (Yang et al., 2018) [1], LoCoMo (Maharana et al., 2024), and QuALITY (Pang et al., 2022);
- Short-context QA: ARC-Challenge (Clark et al., 2018), MMLU (Hendrycks et al., 2021), WinoGrande (Sakaguchi et al., 2019), HellaSwag (Zellers et al., 2019), GSM8K (Cobbe et al., 2021).

We extend our evaluations to multi-round heterogeneous-agent communication on HIDDENBENCH (Li et al., 2026). Appendix A lists dataset sizes and decoding settings. All calibration and benchmark runs are conducted on NVIDIA H100 80GB GPUs.

Calibration setting. For all methods, we use same 1,600 training prompts (800 Open-R1 (Hugging Face, 2025) and 800 HotpotQA) and 400 held-out prompts (200 from each), excluding gold answers and solutions. The HotpotQA prompts used for calibration are drawn from the training split and do not overlap with the evaluation sets. HeteroFold fits Recolor maps on the training set and selects rank-16 correction checkpoint on the held-out set. KV Ridge uses the training set for R^2-based layer selection and ridge fitting, while Dense Latent uses it for K/V reconstruction and generates receiver traces for its second stage with a 512-token cap. Appendix B.3 reports sensitivity to these choices.

4.1 RESULTS ON CROSS-FAMILY TRANSFER

Table 1 compares HeteroFold with direct cross-family extensions of Dense Latent and KV Ridge, with and without TA. HeteroFold improves over the evaluated cache-transfer baselines in most settings, with particularly clear gains on long-context QA including unseen benchmarks during calibration, while avoiding receiver prefill. TextMas provides the native-prefill reference under lossless text communication.

Giving both baselines the same TA correspondence substantially improves KV Ridge in many settings, but HeteroFold remains stronger, especially on long-context tasks. This shows that its gains extend beyond token correspondence to the K/V mapping and receiver-aware calibration. Additional same-family comparisons are reported in Appendix C.

4.2 RESULTS ON HIDDENBENCH: MULTI-AGENT COMMUNICATION

Single-hop QA tests whether cache transfer preserves a fixed prompt, while multi-agent systems must also communicate newly generated information across agents. We therefore evaluate HeteroFold on HIDDENBENCH (Li et al., 2026), where agents must exchange private information to recover the correct answer. Each task uses three or four agents from Llama-3.1-8B, Qwen3-4B, and Ministral-3-14B. Agents first vote independently, communicate for 15 rounds, and then vote again.

Table 2: Group decisions on HIDDENBENCH. Initial accuracy precedes communication; the remaining metrics are measured after 15 rounds.

Communication	Init. Acc. (%)	Avg. Acc. (%)	Majority Acc. (%)	Invalid (%)
TextMas	11.5	29.2	27.7	4.0
Dense Latent + TA	11.5	18.5	7.7	3.2
KV Ridge + TA	11.5	21.8	16.9	20.9
HeteroFold (Ours)	11.5	**30.6**	**29.2**	3.2

Unlike single-hop QA, messages are generated during communication. TextMas sends them as text with native receiver prefill, while Dense Latent, KV Ridge, and HeteroFold transfer messages through mapped KV caches without receiver prefill. HeteroFold achieves performance comparable to TextMas while outperforming the evaluated cache-transfer baselines. Appendix A.2 gives the full evaluation settings.

4.3 COMPONENT ABLATION

Table 3 isolates the main components of HeteroFold for Ministral-3-14B→Llama-3.1-8B. Replacing TA with same-index pairing causes the largest degradation, highlighting the importance of

[1] We use 200-item LongBench subsets (Bai et al., 2024) for Qasper and HotpotQA.

Table 3: Component ablation for Ministral-3-14B→Llama-3.1-8B. ARC-C, GSM8K and QuALITY report accuracy; Qasper, HotpotQA and LoCoMo report F1. The first three rows replace token alignment, depth aggregation, and cross-head mixing, respectively. All variants use the same calibration data as the main results; Calibration refers to our output-aware calibration.

Map	ARC-C	GSM8K	Qasper	HotpotQA	LoCoMo	QuALITY
Same-index token pairing	69.20	7.13	3.19	0.72	1.47	24.59
Single-layer mapping	82.00	46.17	22.34	37.87	30.45	71.52
Head-local mapping	71.84	26.69	9.71	17.38	12.75	54.79
TA + Ridge (ℓ_2-regularized least squares)	87.03	4.55	3.27	6.82	2.13	47.41
+ key + value calibration	89.16	55.57	21.65	31.66	31.91	74.50
TA + Recolor	90.19	62.77	26.52	40.71	40.81	79.34
+ key calibration	90.19	62.35	34.99	48.71	42.85	78.41
+ value calibration	90.53	62.02	25.91	44.50	40.57	77.95
+ key + value calibration (HeteroFold)	90.36	63.46	33.72	50.29	43.15	79.00

Table 4: Transfer latency (in ms, ↓) and speedup over Native Prefill (×, ↑) at batch size 2. All methods use FlashAttention-2. Latency sums synchronized stage timings across two NVIDIA H100 80GB GPUs connected by NVLink, including inter-GPU payload transfer for cache-transfer methods. Native Prefill processes the original text on the receiver. Parentheses report speedup over Native Prefill. Median of three runs after warm-up.

	Llama-3.1-8B → Ministral-3-14B			Qwen3-4B → Ministral-3-14B		
Method	**4,096 tokens**	**16,384 tokens**	**32,768 tokens**	**4,096 tokens**	**16,384 tokens**	**32,768 tokens**
Native Prefill	443.1 (1.00×)	2091.2 (1.00×)	5167.4 (1.00×)	445.1 (1.00×)	2099.6 (1.00×)	5178.2 (1.00×)
Dense Latent + TA	155.4 (2.85×)	372.1 (5.62×)	705.7 (7.32×)	168.9 (2.64×)	389.3 (5.39×)	720.5 (7.19×)
KV Ridge + TA	135.6 (3.27×)	301.0 (6.95×)	569.9 (9.07×)	140.4 (3.17×)	316.5 (6.63×)	587.7 (8.81×)
HeteroFold (Ours)	**118.3** (3.75×)	**257.8** (8.11×)	**481.3** (10.74×)	**128.4** (3.47×)	**290.1** (7.24×)	**483.2** (10.72×)

cross-tokenizer alignment. Single-layer and head-local mappings also reduce performance, supporting multi-layer aggregation and cross-head mixing. Recolor outperforms Ridge (ℓ_2-regularized least squares), showing the benefit of preserving receiver feature statistics. While applying output-aware calibration substantially recovers Ridge's performance, it still falls short of TA + Recolor with output-aware calibration, where key correction contributes most of the gains on long-context tasks.

5 Analysis

5.1 Latency Measurement

We measure transfer latency for Llama-3.1-8B→Ministral-3-14B and Qwen3-4B→Ministral-3-14B on 4K, 16K, and 32K QuALITY contexts with batch size 2. The sender and receiver reside on two separate NVIDIA H100 80GB GPUs connected by NVLink. All methods use FlashAttention-2. Latency sums synchronized timings for tokenization, TA, inter-GPU payload transfer, K/V mapping, cache construction, receiver-side processing, and first-token computation. The transferred context is provided to the receiver as a mapped KV cache without receiver prefill. Sender prefill, model loading, and offline calibration are excluded. We report the median of three runs after warm-up. HeteroFold has the lowest latency in both directions. Compared with Dense Latent's two-layer MLP mapper, HeteroFold uses a single affine mapping stage. It also uses three sender layers per receiver layer, compared with eight in KV Ridge. Speedup over Native Prefill grows from about 3.5–3.8× at 4K to about 10.7× at 32K.

5.2 Cross-Family Transfer Analysis

We compare HeteroFold with Dense Latent and KV Ridge, both using TA, while *Native* denotes direct receiver prefill. Same-index pairing can match different text positions across tokenizers, and Figure 5 shows that this mismatch grows with context length, while TA maintains close align-

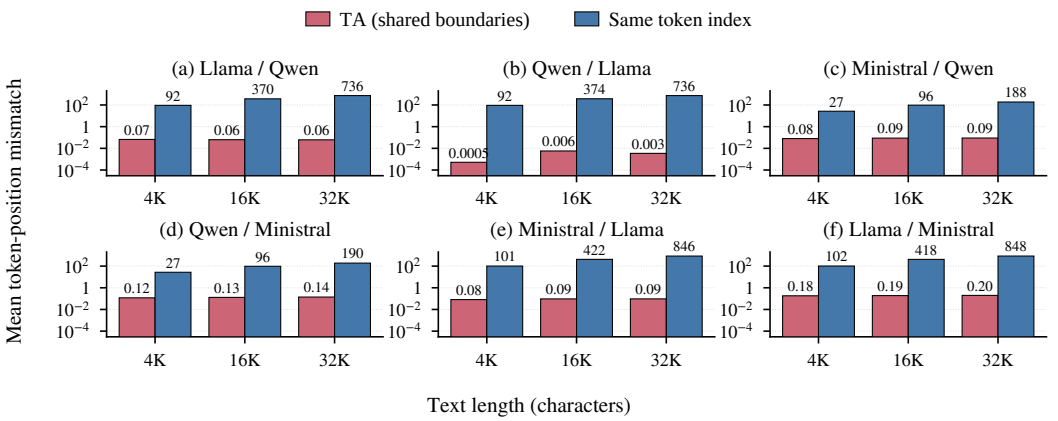


Figure 5: Token alignment across all six transfer directions. We compare TA with same-token-index pairing using the mean absolute difference in character position between matched sender and receiver tokens. Results use 48 HotpotQA documents at 4K, 16K, and 32K character lengths. Lower is better. Titles indicate sender / receiver.

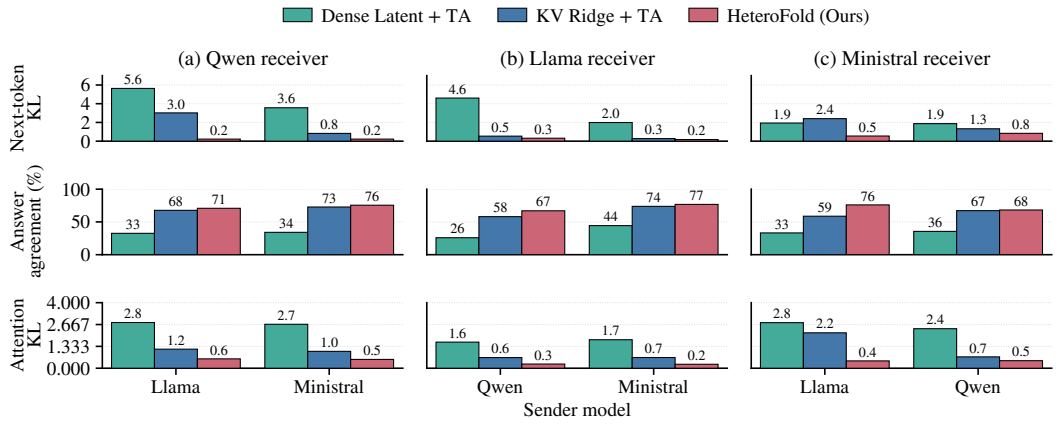


Figure 6: Receiver behavior preservation after cache transfer on QuALITY. Panels (a–c) group results by receiver model and report next-token KL, answer agreement, and attention KL for the corresponding sender models. Lower KL and higher agreement indicate closer native receiver behavior.

ment through shared character boundaries. On QuALITY, Figure 6 further shows that HeteroFold more closely preserves the native receiver's next-token distributions, attention weights, and answer choices than both baselines across all six transfer directions. With TA shared across methods, this comparison evaluates their K/V mapping designs, including HeteroFold's receiver-aware calibration.

6 Conclusion

HeteroFold enables prefill-free KV cache transfer across different model families while keeping both language models frozen. By resolving tokenizer and model-structure mismatches and calibrating the transferred cache against native receiver behavior, HeteroFold improves over the evaluated cache-transfer baselines across six transfer directions, including all four long-context benchmarks, while reducing receiver-side transfer latency. These results show that KV computation can be reused across model-family boundaries without requiring receiver prefill. HeteroFold provides a step toward efficient cache sharing among heterogeneous language-model agents and more general KV interfaces across model families.

AI USE STATEMENT

AI tools were used to improve the clarity, grammar, and readability of the manuscript. All AI-assisted edits were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, December 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Huaben Chen, Wenkang Ji, Lufeng Xu, and Shiyu Zhao. Multi-agent consensus seeking via large language models. 2023. URL https://arxiv.org/abs/2310.20151.

Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. See what i see, know what i think: Dense latent communication across heterogeneous agents, 2026. URL https://arxiv.org/abs/2606.13594.

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

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

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LeatkxrBCi.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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. 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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Hugging Face. Open R1: A fully open reproduction of DeepSeek-R1, January 2025. URL https://github.com/huggingface/open-r1.

Haejoon Lee, Vincent-Daniel Yun, Dimitra Panagou, and Sai Praneeth Karimireddy. Robust multi-agent llms under byzantine faults. 2026. URL https://arxiv.org/abs/2605.09076.

Yuxuan Li, Aoi Naito, and Hirokazu Shirado. Systematic failures in collective reasoning under distributed information in multi-agent LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=igHBKQaLLP.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents, 2024. URL https://arxiv.org/abs/2402.17753.

Mistral AI. Ministral-3-14B-Instruct-2512-BF16, 2025. URL https://huggingface.co/mistralai/Ministral-3-14B-Instruct-2512-BF16.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966. doi: 10.1007/BF02289451. URL https://doi.org/10.1007/BF02289451.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities, 2024. URL https://arxiv.org/abs/2406.04692.

Sunghyeon Woo, Hoseung Kim, Sunghwan Shim, Minjung Jo, Hyunjoon Jeong, Jeongtae Lee, Joonghoon Kim, Sungjae Lee, Baeseong Park, Se Jung Kwon, and Dongsoo Lee. PrefillShare: A shared prefill module for KV reuse in multi-LLM disaggregated serving. 2026. URL https://arxiv.org/abs/2602.12029.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent 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, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger

Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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/.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-MAS: Towards building multi-agent systems with heterogeneous LLMs, 2025. URL https://arxiv.org/abs/2505.16997.

Vincent-Daniel Yun, Woosang Lim, Minsoo Cheong, Sunwoo Lee, Murali Annavaram, Sai Praneeth Karimireddy, and Sungjoo Yoo. Output-aware rotation for INT2 KV-cache quantization. 2026. URL https://arxiv.org/abs/2608.02691.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Jiaru Zou, Rui Pan, Ruizhong Qiu, Pan Lu, Shizhe Diao, Jindong Jiang, Hanghang Tong, Tong Zhang, Markus J Buehler, Jingrui He, et al. Recursive multi-agent systems. *arXiv preprint arXiv:2604.25917*, 2026a.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=syG9I9ofd8.

APPENDIX: HETEROFOLD

This appendix provides the evaluation settings, implementation details, additional receiver analyses, and latency measurements used in the main paper.

- **Appendix A.** Experimental Details
  - A.1 Models and Evaluation
  - A.2 HiddenBench Evaluation
  - A.3 Calibration and Baseline Implementations
- **Appendix B.** Alignment and Calibration Details
  - B.1 TA Implementation
  - B.2 Output-Aware Calibration Details
  - B.3 Sensitivity to Calibration Choices
  - B.4 Mapper Size and Construction Cost
- **Appendix C.** Additional Experimental Results: Same-Family
- **Appendix D.** Additional Receiver Analysis
  - D.1 Measuring Receiver Behavior Preservation
  - D.2 Natural-Language Content Preservation
  - D.3 Stability under Autoregressive Decoding
- **Appendix E.** System and Latency Details
  - E.1 Measurement
  - E.2 Latency Breakdown
  - E.3 Latency across All Six Directions
  - E.4 Sender-Side Cost

A Experimental Details

A.1 Models and Evaluation

We use `meta-llama/Llama-3.1-8B-Instruct`, `Qwen/Qwen3-4B`, and `mistralai/Ministral-3-14B-Instruct-2512-BF16`. All model weights are frozen. Qwen uses non-thinking mode, while Ministral uses its language component. Each model tokenizes the same prompt with its own tokenizer, and native chat-template tokens are handled separately. We evaluate pairwise cache transfer on short- and long-context QA, together with multi-round communication on HiddenBench. For the controlled-length latency table, the sender and receiver are hosted on two separate H100 80GB GPUs connected by NVLink.

Table A: Evaluation sets and metrics. Qasper and HotpotQA use LongBench subsets. LoCoMo excludes unanswerable category 5.

Suite	Dataset	Examples	Metric
Short context	ARC-Challenge	1,172	Accuracy
	MMLU	14,042	Accuracy
	WinoGrande	1,267	Accuracy
	HellaSwag	10,042	Accuracy
	GSM8K	1,319	Accuracy
Long context	Qasper (LongBench)	200	Token F1
	HotpotQA (LongBench)	200	Token F1
	LoCoMo, categories 1–4	1,540	Token F1
	QuALITY, development	2,086	Accuracy
Multi-agent	HiddenBench	65	Accuracy / invalid rate

Qasper and HotpotQA use temperature 0.6, top-p 0.95, and top-k 20. LoCoMo uses greedy decoding with a 64-token generation limit. GSM8K uses greedy decoding with final-number exact match. HellaSwag and WinoGrande use length-normalized candidate likelihood, while ARC-Challenge and MMLU use answer-token likelihood. All methods use the same prompts and decoding settings within each task.

A.2 HiddenBench Evaluation

We evaluate all 65 Hidden Profile tasks from HiddenBench (Li et al., 2026). Each participant receives the public scenario and shared facts, plus one private fact. Three-agent tasks use one Llama, Qwen, and Ministral participant. Four-agent tasks add a second Qwen participant. Each task consists of an initial vote, 15 communication rounds, and a final vote. We use greedy BF16 decoding with limits of 160 tokens per message and 384 per vote. Invalid votes count as incorrect. We report mean participant accuracy, majority-vote accuracy, and the invalid-response rate.

TextMas sends generated messages as text for receiver prefill. For cross-family cache transfer, the sender encodes its message after the public context, and only the message-span K/V states are mapped to the receiver token grid. Dense Latent and KV Ridge use their TA variants. Qwen-to-Qwen messages remain text for every method. Private facts and conversation histories are transferred only when expressed in a generated message.

A.3 Calibration and Baseline Implementations

Calibration uses 1,600 training prompts and 400 held-out prompts, balanced between Open-R1 problem text and HotpotQA training passages and questions. Gold answers and solutions are excluded. HeteroFold estimates its moment statistics on the training split and selects the output-aware correction using held-out loss.

For single-hop evaluation, TextMas represents lossless text communication. The original prompt is provided directly to the receiver, which processes it through native prefill and constructs its own KV

cache. In HIDDENBENCH, the messages are generated during communication, so TextMas sends each generated message as text and the receiving agent performs native prefill.

Since there are no public implementations for Dense Latent and KV Ridge, we reimplement both following their published mapping and layer-selection settings. KV Ridge uses its default $k = 8$ setting, selects sender layers using training-set head-averaged K/V R^2, and fits separate ridge maps with regularization 0.01. Dense Latent uses proportional depth routing and separate two-layer GELU mappings. Its second training stage uses receiver-generated traces capped at 512 tokens.

Their original evaluations use models within the same family and compatible tokenization. For the direct cross-family control in Table 1, the sender and receiver tokenize the same prompt separately and sender token i is paired with receiver token i. Calibration stops at the shorter sequence. Receiver positions beyond the sender sequence reuse the final sender token. The +TA variants change only token correspondence while preserving each baseline's mapping and layer-selection design.

B ALIGNMENT AND CALIBRATION DETAILS

B.1 TOKEN ALIGNMENT (TA) IMPLEMENTATION

TA pairs nonempty sender and receiver tokens that end at the same character position in the original text. A receiver position without an exact match uses the sender token at the latest preceding shared boundary. Positions before the first match use the first sender token. In many-to-one cases, sender states are not averaged; the sender token at the matched character boundary is used. This forward filling also handles spans that the receiver tokenizes more finely than the sender. TA is applied to the shared raw-text span, while special and chat-template tokens are handled separately. Figure 5 measures the resulting position mismatch over long documents. TA changes only token correspondence. Layer alignment and map construction remain specific to each method.

Token-position mismatch. For each document prefix, we tokenize the same raw text without special tokens and record the character-end offsets $e_i^{\mathcal{S}}$ and $e_j^{\mathcal{R}}$ of every nonempty sender and receiver token. Let $a(j)$ be the sender index assigned to receiver token j. TA uses the sender token at the latest shared character boundary, with the first sender token used before the first shared boundary. Using one-based token indices, same-index pairing instead uses $a(j) = \min(j, n_{\mathcal{S}})$, where $n_{\mathcal{S}}$ is the sender sequence length. Receiver positions beyond the sender sequence reuse the final sender token. For the set J_d of nonempty receiver tokens in document prefix d, its mismatch is

$$m_d = \frac{1}{|J_d|} \sum_{j \in J_d} \left| e_{a(j)}^{\mathcal{S}} - e_j^{\mathcal{R}} \right| \tag{10}$$

Each bar in Figure 5 is the arithmetic mean of m_d over the same 48 documents. Thus every document receives equal weight, and the quantity is measured in characters. The calculation uses tokenizer offsets only and does not require a model forward pass.

The three-layer sender neighborhood is clipped at model boundaries. HeteroFold maps keys before key normalization and RoPE, and values after the value projection. The receiver then applies its native key normalization and RoPE once. Mapped K/V features use BF16, while RoPE arithmetic uses FP32.

B.2 OUTPUT-AWARE CALIBRATION DETAILS

Moment statistics are estimated independently for K and V at every receiver layer. The Open-R1 and HotpotQA calibration corpora receive equal total aligned-token weight. The Recolor initialization matches receiver feature means and covariances before output-aware calibration.

We evaluate the calibration losses at query positions in the question span, attending over the full causally accessible prompt. For the key objective, native receiver queries and values remain fixed. We match native attention weights through routing KL and compute the key-induced post-W_O output error separately for each query head, averaging over heads and question tokens. For the value objective, we recompute the mapped attention weights using the current key correction at each training forward pass and stop gradients through these weights. We then optimize the mapped values

against the native post-W_O output using normalized squared error. The key and value corrections are optimized jointly, with each objective updating only its corresponding correction.

The correction rank is $\rho_c = 16$. The B factor is initialized from a zero-mean normal distribution with standard deviation 0.02, while G starts at zero. We optimize for four epochs using AdamW with learning rate 3×10^{-4}, weight decay 10^{-4}, and gradient clipping at 1. We retain the checkpoint with the lowest combined held-out loss.

Figure A shows that the receiver-aware calibration objectives decrease across all six transfer directions. The reported values use the checkpoint selected by the lowest combined held-out loss. The component ablation in Table 3 separately measures the downstream contribution of Recolor, head mixing, and output-aware calibration.

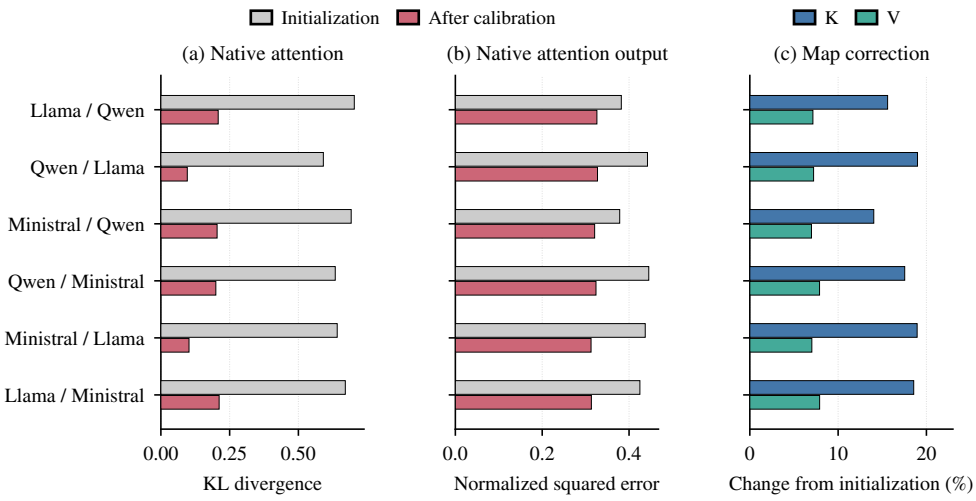


Figure A: Output-aware calibration on held-out prompts. Panels (a,b) compare the Recolor initialization with the checkpoint selected by the lowest combined held-out loss using attention KL and normalized attention-output error. Panel (c) shows the relative change of the affine map. Rows denote sender / receiver.

The learned low-rank correction is folded into the affine map after calibration. The inference-time map has the same form as the initial affine transformation and requires no separate correction module.

B.3 Sensitivity to Calibration Choices

Tables B–E examine the main calibration choices for Llama-3.1-8B→Qwen3-4B. Our default setting uses an equal mixture of Open-R1 and HotpotQA with 1,600 training and 400 held-out prompts, correction rank $\rho_c = 16$, and three sender layers per receiver layer. Each experiment varies one calibration choice while keeping the remaining settings fixed. The calibration-corpus comparison uses 800 training and 200 held-out prompts to keep the number of prompts equal across corpus choices. ARC-C and QuALITY report accuracy, while HotpotQA reports token F1.

Table B: Sensitivity to calibration-set size with an equal Open-R1/HotpotQA mixture, correction rank 16, and three sender layers.

Training / held-out prompts	**ARC-C**	**HotpotQA**	**QuALITY**
200 / 50	72.87	26.40	61.55
800 / 200	75.68	41.97	63.57
1,600 / 400 (paper)	77.05	49.44	62.90

Table B shows that increasing the calibration set from 200/50 to 1,600/400 improves ARC-C from 72.87 to 77.05 and HotpotQA from 26.40 to 49.44. QuALITY changes less between the two larger settings.

Table C: Sensitivity to correction rank with 1,600 training and 400 held-out prompts from the equal Open-R1/HotpotQA mixture and three sender layers. Rank 0 uses Recolor without output-aware calibration.

Correction rank ρ_c	**ARC-C**	**HotpotQA**	**QuALITY**
0 (Recolor only)	75.43	30.47	65.10
4	75.77	47.60	63.09
8	75.51	46.78	63.37
16 (paper)	77.05	49.44	62.90
32	76.71	44.45	64.57
64	76.11	43.09	64.05

Table C shows that all nonzero ranks improve HotpotQA over Recolor alone, while ARC-C and QuALITY vary within a narrower range. We fix $\rho_c = 16$ for all directions in advance.

Table D: Sensitivity to the sender-layer neighborhood with 1,600 training and 400 held-out prompts from the equal Open-R1/HotpotQA mixture and correction rank 16.

Sender-layer neighborhood	**ARC-C**	**HotpotQA**	**QuALITY**
1 layer	66.89	22.52	51.58
3 layers (paper)	77.05	49.44	62.90
5 layers	76.96	48.44	64.43

Table D shows that using three sender layers improves ARC-C, HotpotQA, and QuALITY from 66.89/22.52/51.58 to 77.05/49.44/62.90 compared with a single layer. The three- and five-layer settings give similar results, so we use three layers by default.

Table E: Sensitivity to the calibration corpus with 800 training and 200 held-out prompts in total, correction rank 16, and three sender layers. The equal mixture uses 400 training and 100 held-out prompts from each source.

Calibration corpus (800 / 200)	**ARC-C**	**HotpotQA**	**QuALITY**
Open-R1 only	73.21	12.69	47.70
HotpotQA only	74.32	36.87	60.98
Equal mixture	75.68	41.97	63.57

Table E shows that the equal Open-R1/HotpotQA mixture gives the highest result on all three benchmarks, reaching 75.68 on ARC-C, 41.97 on HotpotQA, and 63.57 on QuALITY. This indicates that combining reasoning and long-context calibration prompts is more effective than using either source alone, including on ARC-C and QuALITY, which are not used for calibration.

B.4 Mapper Size and Construction Cost

HeteroFold stores one folded affine map per receiver layer and role. With three sender-layer inputs, the mapper contains

$$2L_{\mathcal{R}}\left(3d_{\mathcal{S}}^{\mathrm{KV}}d_{\mathcal{R}}^{\mathrm{KV}} + d_{\mathcal{R}}^{\mathrm{KV}}\right)$$

parameters. This corresponds to approximately 0.40–0.50 GB in BF16 for the evaluated transfer directions. The rank-16 correction is folded into the affine maps and requires no separate inference

Table F: Stored mapper parameters per transfer direction, in millions, under the main experimental settings. HeteroFold uses three sender layers, KV Ridge uses eight, and Dense Latent uses its two-layer MLP mapper.

Direction	Dense Latent	KV Ridge	HeteroFold
Llama-3.1-8B → Qwen3-4B	303	604	227
Qwen3-4B → Llama-3.1-8B	270	537	201
Ministral-3-14B → Qwen3-4B	303	604	227
Qwen3-4B → Ministral-3-14B	337	671	252
Ministral-3-14B → Llama-3.1-8B	270	537	201
Llama-3.1-8B → Ministral-3-14B	337	671	252

Table G: Same-family transfer results across four Qwen3 directions. Dense Latent and KV Ridge follow the token correspondence and mapping settings of their original same-family formulations, while HeteroFold uses its standard Token Alignment (TA). All methods use the same calibration data and train/held-out split as the main experiments. TextMas uses lossless text communication with native receiver prefill. Higher is better. Bold indicates the best cache-transfer result within each direction and benchmark; TextMas is excluded from emphasis.

		Long Context QA				Short Context QA				
Transfer direction	**Method**	**Qasper** (F1 ↑)	**HotpotQA** (F1 ↑)	**LoCoMo** (F1 ↑)	**QuALITY** (Acc. ↑)	**ARC-C** (Acc. ↑)	**MMLU** (Acc. ↑)	**WinoGrande** (Acc. ↑)	**HellaSwag** (Acc. ↑)	**GSM8K** (Acc. ↑)
Qwen3-4B → Qwen3-8B	TextMas	47.81	60.34	48.18	77.09	91.47	72.35	58.72	50.79	93.40
	Dense Latent	9.27	7.93	7.48	61.74	80.03	39.77	51.85	53.01	90.30
	KV Ridge	35.21	50.09	34.57	69.03	85.92	67.38	54.54	**54.56**	90.22
	HeteroFold (Ours)	**40.27**	**55.56**	**42.02**	**69.27**	**86.86**	**67.89**	**56.99**	49.73	**90.98**
Qwen3-4B → Qwen3-14B	TextMas	45.70	62.50	50.42	80.63	92.32	76.56	57.93	43.23	94.01
	Dense Latent	7.10	4.43	4.55	47.99	61.60	36.17	50.43	**52.03**	38.67
	KV Ridge	29.02	45.16	26.05	68.46	**86.69**	**67.33**	**55.25**	46.42	90.22
	HeteroFold (Ours)	**34.13**	**54.23**	**37.61**	**69.70**	86.60	66.76	54.70	41.67	**90.98**
Qwen3-14B → Qwen3-4B	TextMas	43.07	56.05	38.92	70.61	87.80	67.95	55.09	41.54	90.14
	Dense Latent	4.44	1.71	0.42	25.31	25.60	37.46	50.91	43.07	35.86
	KV Ridge	39.78	44.35	35.41	72.87	90.61	**74.03**	**53.43**	**44.74**	89.69
	HeteroFold (Ours)	**41.20**	**56.06**	**36.95**	**75.12**	**91.21**	73.58	52.88	40.00	**91.28**
Qwen3-14B → Qwen3-8B	TextMas	47.81	60.34	48.18	77.09	91.47	72.35	58.72	50.79	93.40
	Dense Latent	4.06	2.35	1.26	30.44	48.29	43.53	50.04	48.44	34.95
	KV Ridge	36.26	53.99	39.78	76.32	89.42	**74.31**	55.33	**55.47**	91.96
	HeteroFold (Ours)	**40.08**	**55.87**	**40.63**	**76.46**	**90.61**	73.86	**56.59**	50.47	**92.19**

parameters. KV Ridge with $k = 8$ stores larger mappings because each receiver layer uses all KV heads from eight selected sender layers.

Table F shows that HeteroFold stores 201–252 million mapper parameters across the six transfer directions, approximately 25% fewer than Dense Latent and 62% fewer than KV Ridge. Constructing one HeteroFold transfer direction takes approximately two H100 GPU-hours under the default calibration setting of 1,600 training and 400 held-out prompts, three sender layers, and correction rank 16. This includes moment estimation, receiver-target collection, and four epochs of low-rank calibration.

C Additional Experimental Results: Same-Family

To complement the cross-family evaluation, we additionally evaluate HeteroFold on same-family transfers between Qwen3 models of different scales. For Dense Latent and KV Ridge, we follow the token correspondence, mapping, and fitting procedures of their original same-family formulations rather than applying our Token Alignment (TA). All methods use our calibration setting, including the same calibration data and train/held-out split as in the main experiments. These experiments

also provide a sanity check for the baseline implementations in their original same-family transfer setting.

Table G shows that HeteroFold also performs well in same-family Qwen3 transfers, with consistent gains over the evaluated cache-transfer baselines on long-context QA. Dense Latent retains reasonable performance on some short-context tasks but degrades sharply on long-context QA. KV Ridge remains competitive on several short-context tasks, which also provides evidence that the baseline implementation behaves as expected in the same-family setting. HeteroFold remains competitive on short-context QA while outperforming both baselines on Qasper, HotpotQA, LoCoMo, and QuALITY across all four transfer directions. These results show that HeteroFold's gains extend beyond cross-family tokenizer alignment to same-family KV cache transfer.

D Additional Receiver Analysis

D.1 Measuring Receiver Behavior Preservation

Figure 6 in the main paper uses all 2,086 QuALITY development questions for each transfer direction. The native receiver prefills the prompt and serves as the behavioral reference. Each transferred cache is built from the same sender capture and TA correspondence.

At the answer position, we record the native and transferred receiver distributions over the full vocabulary, denoted by p^{N} and p^{T}. For each question, next-token KL is

$$D_{\mathrm{KL}}(p^{\mathrm{N}} \| p^{\mathrm{T}}) = \sum_{v} p^{\mathrm{N}}(v) \log \frac{p^{\mathrm{N}}(v)}{p^{\mathrm{T}}(v)} \tag{11}$$

The top row of panels (a–c) reports the median next-token KL across questions for each direction and method. In the middle row, we restrict both distributions to the A/B/C/D answer tokens and record whether their highest-probability choices agree. The displayed value is the percentage of questions with agreement, measuring behavioral consistency with the native receiver rather than correctness against the gold answer.

The bottom row of panels (a–c) evaluates attention preservation on the same QuALITY questions. The native receiver processes the complete prompt, while the transferred arm processes the same receiver suffix over its mapped cache. For every suffix query, receiver head, and layer, let a^{N} and a^{T} denote the native and transferred attention distributions over the valid key positions. We compute $D_{\mathrm{KL}}(a^{\mathrm{N}} \| a^{\mathrm{T}})$ for each pair and report the mean KL averaged over suffix queries, heads, layers, and examples.

D.2 Natural-Language Content Preservation

To complement the example in Figure 1, we measure whether a receiver can reconstruct question text available only through the transferred cache. We sample 100 GSM8K test questions containing 20–60 words. As in Figure 1, the sender prefills the question, transfers its KV cache, and the receiver is prompted to restate the question without access to the original text. For the native reference, the receiver directly processes the original question tokens; for cache-transfer methods, the receiver accesses them only through the transferred cache. All methods use the same Ministral-3-14B→Llama-3.1-8B direction, BF16, greedy decoding, and a 192-token generation limit. Table H reports the reconstruction results.

Table H: Question reconstruction on 100 fixed GSM8K test questions. Ordered overlap is word-level longest-common-subsequence F1.

Method	Ordered overlap	Exact	Reached limit
Native receiver	100.0	100/100	0/100
HeteroFold	**83.04**	**16/100**	8/100
Dense Latent + TA	39.74	0/100	17/100
KV Ridge + TA	76.68	7/100	3/100

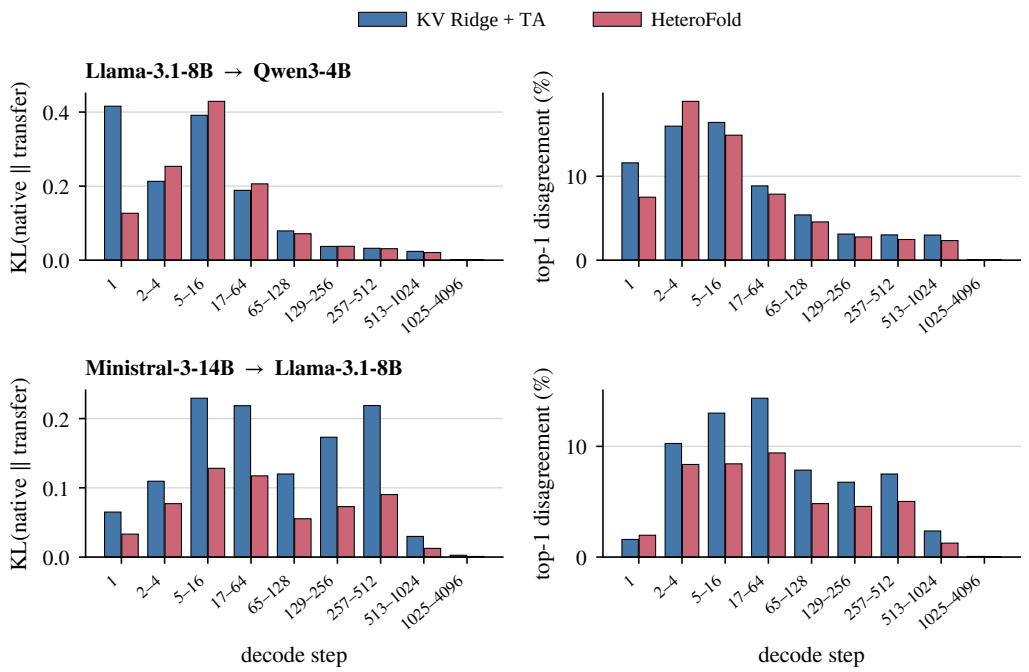


Figure B: Next-token divergence between native and transferred caches along teacher-forced GSM8K solutions. Left: mean full-vocabulary KL by decode-step bin. Right: fraction of positions whose top-1 token differs.

HeteroFold preserves more of the original wording than the evaluated cache-transfer baselines. Ordered overlap measures lexical reconstruction and does not directly measure downstream task accuracy.

D.3 Stability under Autoregressive Decoding

Our output-aware calibration uses only prompt tokens, with losses evaluated at query positions in the question span; no generated answers or reasoning traces are used for calibration. We test whether the resulting maps also preserve native receiver predictions at later decoding positions, beyond those used for calibration.

For every GSM8K test problem, we first obtain a greedy solution trace from the native receiver. Using the frozen calibrated maps, we replay this trace token by token with either the native prompt cache or a transferred cache. Both runs receive the same preceding trace tokens at every step. We compare their next-token distributions using full-vocabulary KL divergence and top-1 disagreement. We evaluate Llama-3.1-8B→Qwen3-4B and Ministral-3-14B→Llama-3.1-8B.

Figure B shows no sustained increase in prediction divergence with decoding length, with generally smaller differences at later positions. These results suggest that prompt-only calibration can preserve receiver behavior beyond the calibrated query positions.

E System and Latency Details

E.1 Measurement

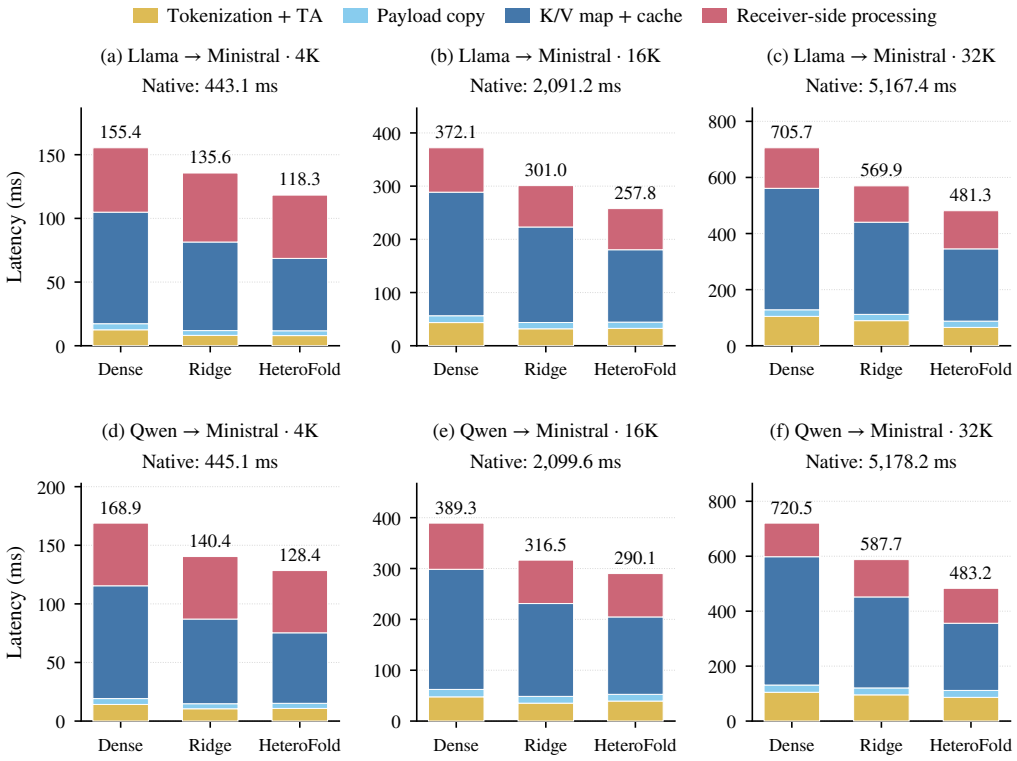


Figure C: Transfer-latency breakdown across two NVIDIA H100 80GB GPUs connected by NVLink at batch size 2, using FlashAttention-2. The payload stage represents inter-GPU transfer between the sender and receiver GPUs. The K/V map and cache stage includes receiver-side cache construction. The final receiver-side stage processes the mapped cache through the first output token; the transferred context itself is not re-prefilled by the receiver. Numbers above bars are total milliseconds. Panel titles give the corresponding Native Prefill time. Each bar uses the stages from the median-total run among three measurements.

Table 4 reports the sum of synchronized stage timings for batches of two QuALITY contexts at 4,096, 16,384, and 32,768 receiver article tokens, before question, options, and chat formatting. Sender prefill is excluded.

The sender and receiver reside on two separate NVIDIA H100 80GB GPUs in BF16, connected by NVLink. All methods use FlashAttention-2. Sender K/V payloads are transferred to the receiver GPU, where K/V mapping, cache construction, and receiver execution are measured.

For cache-transfer methods, the reported total includes receiver tokenization, TA, payload copying, K/V mapping, cache construction, and receiver processing through the first output token for each input. The transferred context is provided as a mapped KV cache without receiver prefill. Native Prefill tokenizes and prefills the original text. Each measured run is preceded by a matched-input warm-up, and we report the median of three runs. Subsequent decoding is excluded.

Table I summarizes the operations included in the measured interval.

Table I: Operations included in the transfer-latency measurement of Table 4.

Stage	Native Prefill	Cache transfer
Alignment / mapping	None	TA and K/V map
Payload	Original text available	H100-to-H100 K/V copy
Transferred context	Native receiver prefill	Mapped KV cache
First receiver token	Included	Included
Sender prefill	Not required in interval	Excluded
Model load / offline fit	Excluded	Excluded
Subsequent decoding	Excluded	Excluded

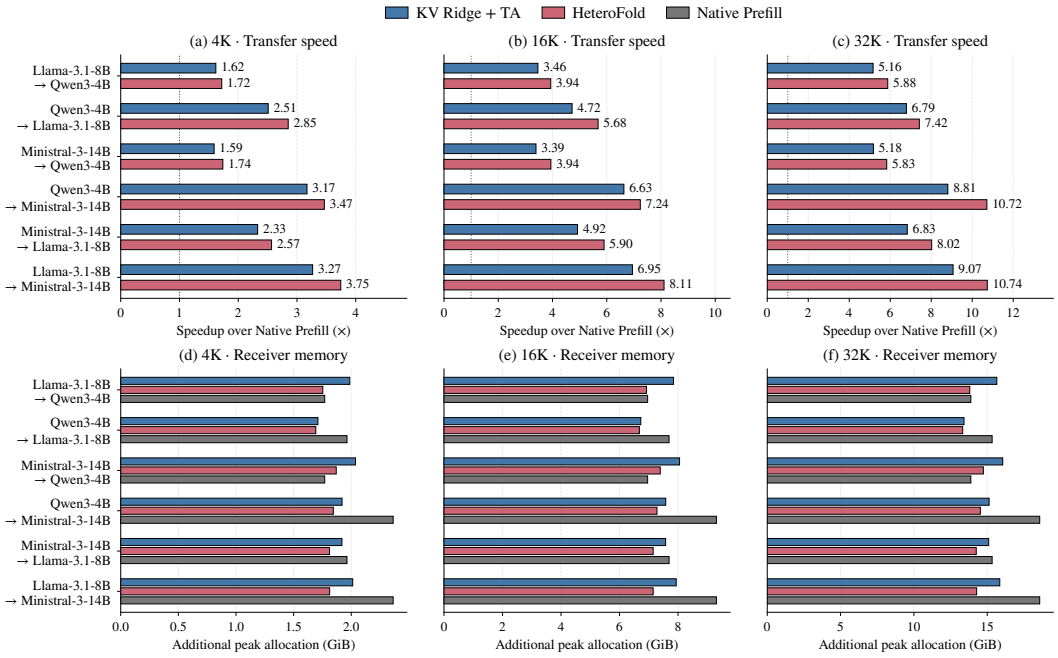


Figure D: Transfer latency and additional receiver memory across all six directions, using two NVIDIA H100 80GB GPUs connected by NVLink, BF16 models, FlashAttention-2, and batch size 2. Sender computation is excluded, while inter-device payload copying is included. Columns show 4K, 16K, and 32K contexts. Top: speedup over Native Prefill, computed as the ratio of median latencies over three runs. Bottom: median additional peak receiver GPU allocation through the first output token, relative to the allocation before hand-off. This excludes resident models, maps, and sender-GPU states, but includes incoming payload copies, temporary buffers, and receiver caches. Rows within each panel denote sender → receiver.

E.2 Latency Breakdown

Figure C decomposes the measurements in Table 4 for both transfer directions and all three context lengths. Each bar uses the stages from the run with the median total latency. Most of HeteroFold's latency advantage comes from its smaller K/V mapping and cache-construction stage.

E.3 Latency across All Six Directions

The main latency table focuses on two transfers into Ministral-3-14B at controlled context lengths and includes an NVLink payload copy. Figure D extends the comparison to all six directions at the same 4K, 16K, and 32K context lengths. Each direction uses two H100 GPUs with batch size 2 and FlashAttention-2. Sender computation is excluded, while inter-device payload copying is included. Each configuration is measured three times after matched-input warm-up, with method order rotated

across repetitions. For the two directions into Ministral-3-14B, the table and both figures use the same measured runs.

Both transfer methods are faster than Native Prefill in all six directions at all three context lengths. HeteroFold also has lower latency and uses less additional peak memory than KV Ridge in every configuration. With incoming payload copies included, cache transfer does not always reduce additional peak receiver memory relative to Native Prefill.

E.4 SENDER-SIDE COST

Table 4 assumes that the sender has already processed the shared context, as in the intended multi-agent setting. If the sender has not yet processed the context, sender prefill and payload capture must be added before cache transfer. For the two-context batches, these costs are 280, 1,328, and 3,381 ms for Llama and 235, 1,163, and 3,083 ms for Qwen at 4K, 16K, and 32K context lengths, respectively, when transferring to Ministral. These values include payload-capture overhead and are reported separately from transfer latency.